UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA AMINISTRACIÓN

ESCUELA DE INGENIERÍA DE SISTEMAS

“HERRAMIENTAS FRAMEWORK DE DESARROLLO OPEN SOURCE”

MONOGRAFÍA PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS

AUTORES:

CAROLINA FLORES
JOSÉ FRANCISCO MONTERO

DIRECTOR:

ING. PABLO PINTADO

CUENCA-ECUADOR
2007

DEDICATORIA

Dedico la culminación de esta monografía a mi querido padre osito, por todos sus sabios consejos, y todo su amor. A mi linda madre, por toda su dedicación y comprensión. A mi dulce hermana por toda su alegría y apoyo. Sin ellos no hubiera podido alcanzar esta gran meta, me han sabido guiar por el mejor camino, y por eso estoy orgullosa de ser parte de ellos.

Carolina Isabel Flores Coronel

Dedico la culminación de este trabajo a mis queridos padres, hermanos y sobrinos; ya que el amor y apoyo que ellos me brindan día a día es un factor clave para lograr alcanzar cada una de mis metas.

José Francisco Montero Flor

AGRADECIMIENTOS

Agradezco a Dios por haber puesto en mi camino a gente tan valiosa, que me ha guiado en varias facetas de mi vida. A mis queridos profesores, los más sinceros agradecimientos, con su dedicación han sabido compartir su conocimiento para hacernos unos verdaderos profesionales. Y a todos mis amigos y compañeros, que con su amistad y sinceridad han podido hacer que varias de mis mentas se hagan realidad.

Carolina Isabel Flores Coronel

Agradezco a Dios que me ha dado y me seguirá dando fortaleza para luchar, por brindarme la salud para poder entregar lo mejor de mi ante cada reto que se presenta, además le agradezco porque él es siempre el proveedor de los medios necesarios para que mi familia pueda salir adelante siempre.

José Francisco Montero Flor

INDICE DE CONTENIDOS

DEDICATORIA……..……ii
AGRADECIMIENTO……...iii
INDICE DE CONTENIDOS……..……iv-viii
INDICE DE ILUSTRACIONES……..……..ix-xiii
RESUMEN……...…….……xiv
ABSTRACT…….……...xv

INTRODUCCION……..1

CAPÍTULO 1: OPEN SOURCE Y FRAMEWORK……..2
1.1. Introducción…….2
1.1.1 Definición……..2
1. 1.2 Ventajas Software Libre……...4
1.2 Framework…….5
1.2.1 Introducción……..……5
1.2.2 Definición…….…5
1.2.3 Importancia de un Framework,……5
1.2.4 Ventajas…….……6
1.2.5 Conclusión…….……6

CAPITULO 2: JAVA

2.1 Introducción……..…….7
2.2 Distribuciones de la tecnología Java…….….8
2.2.1 J2SE-Java2 Standard Editíon…….8
2.2.2 J2EE - Java2 Enterprise Edition.....8
2.2.3 J2ME - Java Micro Edition.....8

2.3 La organización de las APIs……..…….9

2.3.1 JVM (Java Virtual Machine)……...……9

2.3.2 JRE (Java Runtime Enviroment)…….……9

2.3.3 SDK (Software Development Kit)….…….9

2.3.4 HotSpot……..….9

2.4 Tipos de programas Java.....9

2.4.1 Stand-alone.....9

2.4.2 Jawa applets……..……..9

2.4.3 Java servlets……...…9

2.4.4 Java Midlets……...…….9

2.4.5 JavaBeans.....10

2.5 Características fundamentales Java……..……10

2.6 Conceptos erróneos sobre Java…….10

2.7 La portabilidad de Java…….11

2.8 La aplicación Java stand-alone…….……11

2.9 Palabras reservadas del lenguaje Java…..……11

2.10 Variables…….……..12

2.11 El Entorno de Desarrollo de Java…….12

2.12 El compilador de Java……...14

2.13 La Java Virtual Machina…….……..14

2.14 Instalación del entorno java…….…….14

2.15 Conclusión……..……..14

CAPITULO 3: ECLIPSE

3.1 Introducción…….15

3.1.1 Características……...……15

3.2 Historia…….....16

3.2.1 Licencias…….…..16

3.2.2 Idiomas…….…….16

3.3 Descarga e Instalación……..17

3.4 Entorno de Proyectos de Eclipse……..18

3.4.1 Crear un nuevo Proyecto……...18

3.4.2 Proyectos de Eclipse……..…18

3.5 Crear Elementos de Java……...23

3.5.1 Java Class…….….23

3.5.2 File……25

3.5.3 Folder……25

3.5.4 Interface……26

3.5.5 Package…….27

3.5.6 Scrapbook page…….27

3.5.7 Source Fólder……28

3.6 Funciones Útiles de Programación……...28

3.6.1 Compilar y Detectar Errores…….28

3.6.2 Icono de Bombilla = Autocorregir……29

3.6.3 CTRL + Espacio = Autocompletar……...29

3.6 4 Menú "Source"…….32

3.6.5 Refactor Menu…….36

3.6.6 Consultar la documentación……...37

3.6.7 Importar Archivos JAR……..38

3.7.1 Vistas de Eclipse……39

3.7.2 Perspectivas……39

3.7.1.2 Tareas……..40

3.7.1.3 Navigator……41

3.7. 1.4 Package Explorer…….41

3.7. 1.5 Working Set…….42

3.7. 1.6 Outline View……42

3.7. 1.7 Hierarchy View……43

3.7.1.9 Search View…….44

3.7.2 Navegar Vistas y los Editores…….45

3.7.2.1 Maximizar una Vista o Editor……..45

3.7.2.2 Ir al Último Cambio……..45

3.7.2.3 Navegación de los Editores……...46

3.7.2.4 Revisar Problemas…….47

3.8 Ejecutar y Depurar……...….48

3.8.1 Ejecutar…….48

3.8.2 Depurar…….49

3.8.3 Gestión de Cambios…….54

3.9 Optimización Eclipse……55

3.9.1 Descarga e Instalación de Plugins.....55

3.9.2 Visual Editor…….……55

3.9.3 BIRT…….……68

3.10 Conclusión……74

CAPÍTULO 4: APLICACIÓN PRÁCTICA

4.1 Problema……..…75

4.2 Análisis de la aplicación……..76

4.2.1 Diagrama de Casos de Uso……...76

4.2.2. Diagrama de objetos…….77

4.3 Diseño de la Aplicación……77

4.3.1 Realización de la Aplicación restaurante..78

CAPÍTULO 5: CONCLUSIONES …….....84

CAPITULO 6: *RECOMENDACIONES*.…….....*85*

BIBLIOGRAFIA……..……..…86

GLOSARIO…….....88

INDICE DE ILUSTRACIONES

Figura1: Botón hacia la zona de descarga……..……17

Figura2: Link de descarga de Eclipse 3.2.1…….17

Figura3: Elección del Espacio de Trabajo (Workspace) del Proyecto……18

Figura4: Menú Principal……..……18

Figura5: Creación de un Proyecto de Java……..……19

Figura6: Creación de un proyecto en espacio de trabajo…….20

Figura7: Elección de una carpeta para el proyecto…….20

Figura8: Construir de forma automática la estructura de archivos deseada……21

Figura 9: Selección de una fuente de importación……..21

Figura10: Selección de un directorio destino……..22

Figura11: Menú contextual……..23

Figura12: Botón para crear una nueva clase……23

Figura13: Nueva clase de Java……24

Figura 14: Package Explorer, navigator……..25

Figura 15: Crear un archivo…….25

Figura16: Menú Contextual…….26

Figura17: Compilar y Detectar Errores……...26

Figura18 : Auto corregir……..30

Figura19: ctrl.+space……...30

Figura20: Marcas en variables y atributos……...30

Figura21: Métodos y Constructores……31

Figura 22: Auto completar……...31

Figura 23: Auto completar…….....31

Figura 24: Bucles…….32

Figura 25: Etiquetas de Javadoc……..32

Figura 26: Métodos de Implementación……..34

Figura 27: Niveles de Visibilidad……35

Figura28: Renombrar elementos…….36

Figura29: Modificar método marcado…….37

Figura 30: Consultar Documentación……..37

Figura31: Documentación de JavaDoc……....38

Figura 32: Propiedades……....38

Figura 33: Abrir Perspectiva…….....39

Figura 34: íconos de Perspectiva…….40

Figura 35: Carpeta Bin……41

Figura 36: Estructura lógica de paquetes…….41

Figura 37: Nombrar nuevo conjunto de trabajo……...42

Figura38: Métodos o atributos definidos dentro de una clase de java……42

Figura 39: Vista de Jerarquía……...43

Figura40: Vistas Rápidas…….43

Figura 41: Búsqueda……44

Figura42: Resultados búsqueda……...45

Figura 43: Última ubicación editada……...45

Figura 44: Regreso al proyecto……46

Figura 45: navegar hacia delante…….46

Figura 46: Revisar Problemas……..47

Figura 47: Ejecutar Depurar……48

Figura48: Seleccionar un tipo principal……...49

Figura 49: Pasar argumentos al método main…….49

Figura 50: Identificar posibles errores…….....50

Figura51 :Ruptura……....51

Figura52: Rupturas……..52

Figura53: Editor de Código……53

Figura 54: Variables……53

Figura55: Resumen de Modificaciones……..54

Figura 56: Preferencias……55

Figura57: Ruta de Acceso para descarga de actualizaciones …….56

Figura 58: Opción para descarga de actualizaciones……...56

Figura59: Sitio de donde se puede descargar las actualizaciones……...56

Figura 60: Pantalla que muestra el progreso de la conexión al sitio……...57

Figura 61: Pantalla con el listado de actualizaciones disponibles……...57

Figura 62: Opción para aceptar los términos legales……...57

Figura63: Barra de progreso de la descarga de actualizaciones……..57

Figura64: Pantalla de creación de un elemento “Visual Class”……..58

Figura65: Lista de selección del estilo del elemento “Visual Class”……..58

Figura 66: Barra de Herramientas del Visual Editor……...59

Figura 67: Elementos de la pantalla principal del Visual Editor de Eclipse……...60

Figura 68: Ejemplo de modificación de propiedades de un control……61

Figura69: Paleta de elección de controles de Visual Editor……61

Figura70: Modelo de Interfaz del ejemplo de Visual Editor……...62

Figura 71: Creación de un control en el Visual Editor……62

Figura 72: Añadir eventos a un control……...62

Figura73: Selección de tipo de evento en Visual Editor…….63

Figura74: Edición del código del evento…….63

Figura 75: Vista Java Beans de Visual Editor.....68

Figura76: Resultado Final de la Calculadora……..68

Figura 77: Creación de Nuevo Proyecto……..70

Figura 78: Open Perspectiva……70

Figura79: Creación de Reporte……71

Figura80: Perspectiva Report Design……..72

Figura 81: Configurar Conexión……..……73

Figura 82: Layout del Reporte…….73

Figura 83: Vista Preliminar del Reporte……..74

Figura 84: Casos de Uso……..76

Figura85: Diagrama de objetos……77

Figura86: Aplicación –MenúPrincipal Clientes……..77

Figura 87: Aplicación-Menú Pricipal Cuentas…….……...77

Figura88: Aplicación –Ingreso Clientes……..78

Figura89: Aplicación –Modificación Clientes……....……78

Figura90: Aplicación – Consulta de clientes……...78

Figura 91: Aplicación –Eliminar clientes……79

Figura 92: Aplicación – Abrir Cuenta…….79

Figura 93: Aplicación –Cerrar Cuenta…….79

Figura 94: Aplicación –Reabrir Cuentas……..……..80

Figura 95: Aplicación –Debitar Saldo……...……..80

Figura 96: Aplicación –Acreditar Saldo……...…...80

Figura 97:Aplicación-Menú Consultas……81

Figura 98: Consulta de Transacciones…….82

Figura 99: Aplicación –Consulta Cuentas……...……82

Figura 100: Aplicación – Layout Reporte……...83

Figura 101: Aplicación – Vista Preliminar Reporte……..……..……83

RESUMEN

Esta monografía está enfocada específicamente al estudio de “Eclipse”, una herramienta de código abierto (software distribuido y desarrollado libremente), que provee una plataforma de desarrollo y aplicaciones framework para la construcción de software. Eclipse es utilizado como editor de código para algunos lenguajes de programación, entre ellos Java.

Framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting, entre otros, para ayudar a desarrollar y unir los diferentes componentes de un proyecto, representando una arquitectura que modela las relaciones generales de las entidades del dominio.

ABSTRACT

This Project is about Eclipse, an open source (software developed and distributed as free) tool, focused on providing a development platform and application frameworks for building software. Eclipse is used as an open source editor, for writing code in some programming languages one of them Java.

Framework is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project. Frameworks are designed with the intent of facilitating software development, by allowing designers and programmers to spend more time on meeting software requirements rather than dealing with the more tedious low level details of providing a working system.

INTRODUCCION

En la época actual lo fundamental en el desarrollo de aplicaciones es manejar una Ingeniería de Software que brinde a los usuarios calidad, es por eso que los frameworks se han convertido en la base de la Ingeniería de Software. El uso de los mismos está ganando rápidamente la aceptación debido a que permite la reutilización del código del diseño y el código fuente.

El código abierto da la facilidad de aprender de enseñar de competir, dándonos libertad de expresión y de elección, representando en las empresas una disminución considerable en los costos de cualquiera de sus proyectos.
Los frameworks permiten la reutilización tanto del código del diseño como el código fuente.

Eclipse es una herramienta Framework OpenSource, es una estructura (*workbench*) sobre la que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados.

La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc.

CAPITULO 1

OPEN SOURCE AND FRAMEWORK

1.1. Introducción

OPEN SOURCE (código abierto), es todo software que se encuentra distribuido y desarrollado libremente.
Empezaron a utilizar este término algunos usuarios de la comunidad del software libre en 1998, se trató de utilizarlo en vez del nombre original que estaba en ingles del software libre (free software).

Al decir Free se puede pensar en un software por el que no hay que pagar, es decir software gratuito y, por otro lado se puede pensar en un software que posee ciertas libertades.

Traduciendo Literalmente el significado de open source es que se puede mirar el código fuente. Por eso se dice que un programa de código abierto puede ser software libre, semilibre o no libre. Pero casi siempre un programa de código abierto puede ser y de hecho es software libre, como igualmente un programa Software Libre es Open Source.

1.1.1 Definición

"Open Source" se refiere a tener acceso al código fuente, este ofrece la libertad de aprender, libertad de enseñar, libertad de competir, libertad de expresión y libertad de elección, además que estas representan una disminución considerable en los costos de cualquier proyecto empresarial.

A continuación aclaramos ciertos términos:

Software libre (free software)
Software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Software de dominio público
Es aquél por el que no es necesario solicitar ninguna licencia y cuyos derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquél cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado.

Software semilibre
Es una categoría de programas informáticos que no son libres, pero que vienen con autorización de uso, copia, modificación y redistribución (incluso de versiones modificadas) sin fines de lucro (PGP sería un ejemplo de un programa semilibre).

Software no libre
También llamado software propietario, software privativo, software privado, software con propietario o software de propiedad) se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido .

Licencia Open Source

Una licencia open-source es una licencia que se usa para programas de computadoras, con copyright, que siguen los principios del movimiento Open Source.

Una licencia es considerada Open Source cuando ha sido aprobada por la Open Source Initiative (OSI), donde el criterio lo da la definición de Open Source. El software de dominio público (esto significa sin licencia), cumple todos estos criterios siempre y cuando todo el código fuente esté disponible, y esté reconocido por la OSI y se le permita usar la marca de la misma.

La Open Source Initiative utiliza la Definición de Open Source para determinar si una licencia de software de computadora puede o no considerarse software abierto. Es similar pero no igual a la definición de licencia de software libre.

Bajo la Definición Open Source, las licencias deben cumplir diez condiciones para ser consideradas licencias de software abierto:

1. Libre redistribución: el software debe poder ser regalado o vendido libremente.
2. Código fuente: el código fuente debe estar incluido u obtenerse libremente.
3. Trabajos derivados: la redistribución de modificaciones debe estar permitida.
4. Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas solo como parches.
5. Sin discriminación de personas o grupos: nadie puede dejarse fuera.
6. Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.
7. Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa.

8. La licencia no debe ser específica de un producto: el programa no puede licenciarse solo como parte de una distribución mayor.
9. La licencia no debe restringir otro software: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
10. La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por click de ratón o de otra forma específica del medio de soporte del software.

1. 1.2 Ventajas Software Libre

- El término "Open Source" se refiere a tener acceso al código fuente. Pero el acceso al código fuente es apenas un pre-requisito para dos de las cuatro libertades que definen al Software Libre. Muchas personas no entienden que el acceso al código fuente no es suficiente.
- Desafortunadamente, muchas compañías han comenzado a llamar sus productos "Open Source" cuando algunas partes del código son visibles. Los usuarios compran este software creyendo que están adquiriendo algo "tan bueno como GNU/Linux" ya que se supone sigue los mismos principios.

  No debemos permitir que los vendedores de software propio abusen de las personas de esta forma.

El Software Libre ofrece libertad

El Software Libre proporciona la libertad de:

- Ejecutar el programa, para cualquier propósito;
- Estudiar el funcionamiento del programa, y adaptarlo a sus necesidades;
- Redistribuir copias;
- Mejorar el programa, y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.

La definición de software libre dada por la "Free Software Foundation", con sus cuatro libertades, es la definición más clara que existe actualmente.

Como consecuencia de estas 4 libertades, el Software Libre ofrece la libertad de aprender, libertad de enseñar, libertad de competir, libertad de expresión y libertad de elección. Además que estas representan una disminución considerable en los costos de cualquier proyecto empresarial.

1.2 FRAMEWORK

1.2.1 Introducción

En el desarrollo de software, un **framework** es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los lenguajes interpretados (o lenguajes de script) forman un subconjunto de los lenguajes de programación, que incluye a aquellos lenguajes cuyos programas son habitualmente ejecutados en un intérprete en vez de compilados.

1.2.2 Definición

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

En la época actual lo fundamental en el desarrollo de aplicaciones es manejar una Ingeniería de Software que brinde a los usuarios calidad, es por eso que los frameworks se han convertido en la base de la Ingeniería de Software. El uso de los mismos está ganando rápidamente la aceptación debido a que permite la reutilización del código del diseño y el código fuente.

1.2.3 Importancia de un Framework (Utilidades)

- Para no iniciar el proyecto desde cero, cuando lo que se quiere es ganar velocidad en el desarrollo.
- Da tranquilidad en el soporte de la base de la aplicación, se dispone de suficiente documentación.
- Existen patrones de programación, estructura, desarrollo y mantenimiento del código generado
- Disponer de lo mejor de cada plataforma de desarrollo da una sensación y seguridad de un proyecto prospero y bien hecho.

1.2.4 Ventajas

- Disponen de conectores a las bases de datos más conocidas
- La seguridad del proyecto resultante esta avalada por terceras personas que en realidad son miembros de la misma comunidad.
- Características multilenguaje y multicultura
- Disponen de mecanismos de autenticación variados y funcionales
- Son tolerantes a fallos propios o inducidos.
- Gracias a la documentación su curva de aprendizaje es muy buena.
- Disponen de excelentes herramientas para crear formularios haciendo fácil la ca ptura de datos y manejo de errores inducidos por ellos.

1.2.5 CONCLUSION

OpenSource nos da la facilidad de aprender de enseñar de competir, dándonos libertad de expresión y de elección, representando en las empresas una disminución considerable en los costos de cualquiera de sus proyectos.
Frameworks son la base de la Ingeniería de Software, lo que nos lleva a la rápida aceptación, porque permite la reutilización tanto del código del diseño como el código fuente. A la vez tiene la capacidad de separar La Funcionalidad, Los Datos, y La Presentación.

CAPITULO 2

JAVA

2.1 Introducción

Java hizo su aparición en 1991 cuando un grupo de Ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a computadores, cuya reducida potencia de cálculo y memoria los llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Debido a las diferentes arquitecturas existentes y los problemas de incompatibilidad que se presentan constantemente, fue importante conseguir una herramienta independiente del tipo de CPU que se utilice. Se desarrolla un código "neutro" que no es dependiente, se ejecuta sobre una "*máquina virtual*" denominada *Java Virtual Machine* (*JVM*), que es quien interpreta el código neutro convirtiéndolo a código particular de la CPU utilizada. Con esto se logró lo que luego se ha convertido en el principal lema del lenguaje: "*Write Once, Run Everywhere*". A pesar del gran esfuerzo que representó su desarrollo ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje.

Java se introdujo como lenguaje de programación para computadores a finales de 1995 cuando se incorporó un intérprete Java en el programa Netscape Navigator 2.0, produciendo una verdadera revolución en Internet. *Java 1.1* apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje.

Al programar en *Java* no se parte de cero. Cualquier aplicación que se desarrolle se apoya en un gran número de clases preexistentes. Algunas pueden ser hechas por el propio usuario, otras pueden ser comerciales, pero un gran número de ellas forman parte del propio lenguaje (el *API* o *Application Programming Interface*). *Java* incorpora muchas funciones importantes como ejecución remota, componentes, acceso a bases de datos, etc.

El principal objetivo de *Java* es llegar a ser el "nexo universal" que conecte a los usuarios con la información, ya esté situada en el ordenador local, en un servidor *Web*, en una base de datos o en cualquier otro lugar.

Java es un lenguaje muy completo, en cierta forma casi todo depende de casi todo; por ello debe aprenderse de modo iterativo primero de modo muy general y refinando en sucesivas iteraciones.

Sun Microsystems describe el lenguaje *Java* como "*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*".

Los programas desarrollados en *Java* muestran diversas ventajas frente a los desarrollados en otros lenguajes. La ejecución de los programas en *Java* tiene muchas posibilidades: ejecución como aplicación independiente, ejecución como *applet*, ejecución como *servlet*, etc. Un *applet* es una aplicación especial que se ejecuta dentro del browser al cargar una página HTML desde un servidor *Web*. El *applet* se descarga desde el servidor y no requiere instalación en el ordenador donde se encuentra el browser. Un *servlet* es una aplicación sin interface gráfica que se ejecuta en un servidor de Internet. La ejecución como aplicación independiente es similar a los programas desarrollados con otros lenguajes. Además de la ejecución como *Applet*, *Java* permite fácilmente el desarrollo tanto de arquitecturas cliente-servidor como de aplicaciones distribuidas, consistentes en crear aplicaciones capaces de conectarse a otros ordenadores y ejecutar tareas en varios ordenadores simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, *Java* incorpora en su *API* estas funcionalidades.

2.2 Distribuciones de la tecnología Java

Actualmente JavaSoft, empresa del grupo Sun Microsystems que guarda las inversiones y controla la compatibilidad y el desarrollo de las versiones sucesivas de Java, tiene disponibles tres distribuciones de esta tecnología, cada una para un entorno de ejecución. Estas distribuciones son paquetes de software que incluyen toda la plataforma para soportar la ejecución de los programas Java. Los elementos principales de este conjunto son las APIs y un entorno de ejecución o máquina virtual.

2.2.1 *J2SE-Java2 Standard Editíon*

- Provee las principales APIs y enfoca el desarrollo de aplicaciones Cliente/Servidor. No permite la distribución extendida de objetos ni ofrece soporte a tecnologías para Internet;
- Provee la Java Runtime Environment (JRE) o la JVM (Java Virtual Machine).

2.2.2 *J2EE - Java2 Enterprise Edition*

- Provee un conjunto de APIs para desarrollo corporativo y se enfoca a la integración entre sistemas. Permite alta distribución de objetos y ofrece total soporte a tecnologías para Internet;
-Depende de J2SE y de JRE.

2.2.3 *J2ME - Java Micro Edition*

- Provee las APIs necesarias para el desarrollo de aplicaciones para computación móvil, en pequeños dispositivos o tecnologías portátiles;
- Provee un Java Runtime Environment (JRE) de capacidad reducida.

2.3 La organización de las APIs y la plataforma de J2SE.

2.3.1 *JVM (Java Virtual Machine)*

Software que emula la CPU *y* memoria para la ejecución de programas en Java.

2.3.2 *JRE (Java Runtime Enviroment)*

Entorno obligatorio para la ejecución de programas en Java. El JRE es compuesto por JVM y por el conjunto de APIs del J2SE (JVM + APIs = JRE].

2.3.3 *SDK (Software Development Kit)*

Conjunto de herramientas para compilación, documentación y depuración de errores de aplicativos Java. El SDK está compuesto por JRE y por sus herramientas de desarrollo.

2.3.4 HotSpot

Componente del JRE, realiza una compilación previa de fragmentos del código que agiliza la ejecución de los programas.

Para ejecutar cualquier aplicación java, es necesario tener una JRE, que contiene el JVM más las APIs del J2SE.

2.4 Tipos de programas Java

El lenguaje Java es muy versátil, pudiendo ser utilizado para construir varios tipos de aplicaciones, para diferentes medios *y* finalidades. Las diferencias entre los tipos de programas que pueden ser desarrollados es el conjunto de APIs utilizado.

2.4.1 *Stand-alone*: Aplicación basada en el J2SE que tiene total acceso a los recursos del sistema, memoria, disco, redes, dispositivos, etc. Es posible ejecutar una aplicación stand-alone, por ejemplo, en un servidor Web. También es común ejecutar aplicaciones stand-alone de automatización comercial en estaciones de trabajo.

2.4.2 *Java applets*: Pequeñas aplicaciones que no tienen acceso a los recursos de hardware, que necesitan de algún navegador con soporte para J2SE para que sean ejecutadas. Son generalmente usadas en juegos, animaciones, teclados virtuales, etc.

2.4.3 *Java servlets:* Programas desarrollados para su ejecución en servidores Web basados en el J2EE, comúnmente usados para generar contenido dinámico para sitios Web.

2.4.4. *Java midlets*: Pequeñas aplicaciones, extremadamente seguras y construidas para ser ejecutadas dentro del J2ME. Se usan generalmente en teléfonos móviles, ordenadores de mano, controladores electrónicos, ordenadores de a bordo, tarjetas inteligentes, tecnología integrada en vehículos, etc.’

2.4.5 *JavaBeans*: Pequeños programas con patrón bastante rígido de codificación. Pueden ser reaprovechadas en cualquier tipo de programa en Java, pudiendo ser llamados a partir de aplicaciones stand-alone, applets, servlets y midlets.

2.5 Características fundamentales de la tecnología Java

Las características fundamentales (key features) de la tecnología Java son:

• Simplicidad: Contiene solamente 48 palabras reservadas.
• Orientación a objetos: Permite la programación orientada a objetos.
• Capacidad de distribución: Permite la distribución de aplicaciones en varias JREs (en estaciones de trabajo y/o en el servidor) por medio de red.
• Robustez: Utiliza compilación e interpretación de código, lo que aumenta su fiabilidad.
• Portabilidad: «Write once, run anywhere» - una vez escrita *y* compilada, una aplicación en Java se puede ejecutar en cualquier plataforma operativa en que esté instalada un JRE.
• Multithreading: Permite el desarrollo de aplicaciones para que funcionen en servidores que atiendan a muchos usuarios simultáneamente, o de aplicaciones que necesiten ejecutar más de una tarea al mismo tiempo.

2.6 Conceptos erróneos sobre Java

A continuación, los principales mitos sobre Java:

• Se trata de una extensión del HTML.
Falso. En realidad, el Java es un lenguaje completo, derivado del SmallTalk y del C++.

• Es sólo un lenguaje como otro cualquiera.
Falso. Java posee un lenguaje único, que permite construir componentes para todos los entornos.

• Todos los programas Java funcionan en páginas Web.
Falso. Existen tres entornos distintos (J2SE, J2EE y J2ME) de ejecución para programas Java.

• El JavaScript es una versión light de Java.
Falso. Netscape aprovechó la ola de marketing y bautizó su tecnología LiveScript, como JavaScript.

• El lenguaje Java es interpretado, por lo que es muy lento para aplicaciones serías.
Falso. El lenguaje Java realmente exige interpretación, pero incluye también compilación. La forma como la pareja compilador/intérprete trata los programas garantiza un rendimiento muchas veces equivalente al del C++ (con la ventaja de ser un lenguaje mucho más simple que este).

• Es difícil programar en Java.

Falso. La programación en Java en sí es relativamente simple. La única posible dificultad inicial es la asimilación de los conceptos de Orientación a objetos.

2.7 La portabilidad de Java

Java usa las dos formas computacionales de ejecución del software (compilación e interpretación), y reúne las ventajas de ambas. Los lenguajes que utilizan sólo compilación necesitan de instrucciones adicionales de la plataforma operativa, lo que compromete la portabilidad. Por otro lado, los lenguajes que se apoyan sólo sobre la interpretación generalmente son muy lentos. La coordinación de los procesos de compilación e interpretación permite la agilización del proceso de desarrollo del software al mismo tiempo que garantiza la portabilidad de las aplicaciones creadas.

Combinación de las formas de ejecución en Java: en un primer momento, el código se compila, o sea, se transforma en instrucciones comprensibles por la Java Virtual Machine (bytecode).

Ese bytecode, a su vez, puede ser interpretado en varias plataformas operativas, basta con que tenga un intérprete de Java (incluido en un JRE) instalado. Los fabricantes de hardware y sistemas operativos, en conjunto con JavaSoft, desarrollaron JREs para varios entornos operativos. Por eso, las aplicaciones totalmente desarrolladas en Java son 100% portables.

2.8 La aplicación Java stand-alone

Para que una aplicación Java pueda ser considerada del tipo stand-alone y ejecutada directa mente por el intérprete, debe poseer el método main, cuya sintaxis es:

public static void main(String args[])

A partir de esa línea de código el programa Java comienza a ejecutarse. Por lo tanto, el método main desempeña la función de punto de acceso de una aplicación stand-alone. Cuando el proceso de ejecución llega al fin de ese método, la aplicación termina.

Lenguaje de programación Java

Vamos a presentar las principales características específicas del lenguaje Java: palabras reservadas y tipos de variables.

2.9 Lista de las palabras reservadas del lenguaje Java

abstract	double	int	static
boolean	else	interface	super
break	extends	long	switch
byte	final	native	syncronized

case	finally	new	this
catch	float	null	throw
char	for	package	throws
class	goto*	private	transient
const*	if	protected	try
continue	implements	public	void
default	import	return	volatile
do	instanceof	shot	while

"*" Palabras reservadas, pero no usadas por no representar instrucciones de JRE.

2.10 Variables

En lenguajes de programación orientada a objetos, para crear un elemento que represente datos como un atributo de un objeto o un parámetro de un método, debemos hacer uso de variables. Sirven para almacenar determinado tipo de valor dentro de un algoritmo. Ese valor puede variar – por eso el nombre «variable».

Se pueden usar variables de tipo primitivo como atributos de objeto, parámetros de método o variables locales de método.

En el lenguaje Java, existen dos tipos de variables:

• **Tipos primitivos:** capaces de almacenar valores simples, como números enteros *y* de toma flotante, caracteres y booleanos. Se usan para ejecutar cálculos *y* comparaciones;

• **Tipos de referencia:** capaces de almacenar referencias de objetos. Se acepta también una definición por exclusión para los tipos de referencia: serían de referencia todos aquellos que no son primitivos. Se usan para navegar entre los objetos de un sistema.

2.11 Entorno de Desarrollo de Java

Existen distintos programas comerciales que permiten desarrollar código *Java*. *Sun* distribuye gratuitamente el *Java Development Kit* (*JDK*), que es un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en *Java*. Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (Depurador o *Debugger*). Por experiencia sabemos que usualmente la mayor parte del tiempo de desarrollo se destina a la *detección y corrección de errores*. Existe también una versión reducida del *JDK*, denominada *JRE* (*Java Runtime Environment*) que solo ejecuta código *Java*, no permite compilar.

Existen también los *IDE*s (*Integrated Development Environment*) o entornos de desarrollo integrados, que en el mismo programa se puede escribir el código, compilarlo y ejecutarlo sin tener que cambiar de aplicación.

Algunos incluyen una herramienta para depurar gráficamente mucho más sencilla frente a la del *JDK que* utiliza una consola de MS-DOS. Estos entornos integrados permiten desarrollar aplicaciones mucho más rápido, incorporando en muchos casos librerías con *componentes* ya desarrollados que se incorporan al proyecto. Como inconvenientes se pueden señalar algunos fallos de compatibilidad entre plataformas y ficheros resultantes de mayor tamaño que los basados en clases estándar.

2.12 El compilador de Java

Es una de las herramientas de desarrollo incluidas en el *JDK*. Este analiza la sintaxis del código escrito en los ficheros fuente de *Java* (**.java*). Si no hay errores genera los ficheros compilados (**.class*). Sino muestra la línea o líneas erróneas.

2.13 La Java Virtual Machine

La existencia de distintos tipos de procesadores y ordenadores llevó a los ingenieros de *Sun* a la conclusión de que era muy importante conseguir un software que no dependiera del tipo de procesador utilizado. Se plantea la necesidad de conseguir un código capaz de ejecutarse en cualquier tipo de máquina. Una vez compilado no debería ser necesaria ninguna modificación por el hecho de cambiar de procesador o de ejecutarlo en otra máquina. La clave consistió en desarrollar un código "neutro" el cual estuviera preparado para ser ejecutado sobre una "*máquina virtual*", denominada *Java Virtual Machine* (*JVM*). Es esta *JVM* quien *interpreta* este código neutro convirtiéndolo a código particular de la CPU o chip utilizada. Así se evita tener que realizar un programa diferente para cada CPU o plataforma. Esta *JVM* es el intérprete de *Java*. Ejecuta los "*bytecodes*" (ficheros compilados **.class*) creados por el compilador de *Java*.

2.14 Instalación del Entorno Java

Los pasos que detallamos a continuación son los pasos a seguir para instalar el J2SE de Java en Windows XP:

1. Instalamos el J2SE 6 de Java (jdk1.6.0) que es la versión que funciona correctamente con la versión de Eclipse que utilizamos.

2. Se debe cambiar la variable de entorno PATH que se encuentra en:

   Inicio → Panel de Control → Sistema → Opciones Avanzadas

   Aquí damos click en Variables de Entorno.

   En la parte de Variables del Sistema seleccionamos la variable PATH y damos un click en Modificar, y agregamos al inicio de esta:

   C:\Program Files\Java\jdk1.6.0\bin;

3. En caso de que se vaya a llamar a paquetes contenidos en otras carpetas modificamos la variable CLASSPATH de igual forma ingresamos a:

   Inicio → Panel de Control → Sistema → Opciones Avanzadas
   Damos click en Variables de Entorno.

   Aquí en lugar de Variables del Sistema, buscamos en la parte de variables de usuario, si ya existe CLASSPATH la Modificamos sino seleccionamos Nueva, se agrega al final de esta la carpeta desde donde van a ser accedidos los paquetes:

4. Los plugins necesarios se instalan automáticamente al instalar el J2SE.

2.15 CONCLUSION

Java tiene como meta llegar a ser el "nexo universal", conectando a los usuarios con la información, ya esté situada en el ordenador local, en un servidor *Web*, en una base de datos o en cualquier otro lugar.
Este es un lenguaje muy completo, en cierta forma casi todo depende de casi todo, a la vez es: simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico".

CAPITULO 3

3.1 INTRODUCCION

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Desarrollador:	Eclipse Foundation
Última versión:	3.2.1 / 21 de Septiembre de 2006
S.O.:	Multiplataforma
Género:	Java SDK
Licencia:	Eclipse Public License

3.1.1 Características

La versión actual de **Eclipse** dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (*wizards*): para creación de proyectos, clases, tests, etc.
- Refactorización

3.2 Historia

Eclipse comenzó como un proyecto de IBM Canadá. Fue desarrollado por OTI (Object Technology International) como reemplazo de VisualAge también desarrollado por OTI. En Noviembre del 2001, se formo un consorcio para el desarrollo futuro de Eclipse como código abierto. En 2003, la fundación independiente de IBM fue creada.

Eclipse 3.0 (2003) seleccionó las especificaciones de la plataforma OSGi como la arquitectura de tiempo de ejecución.

En 2006 la fundación Eclipse coordino sus 10 proyectos de código abierto, incluyendo la Plataforma 3.2, para que sean liberados el mismo día. Esta liberación simultánea fue conocida como la liberación Callisto.

3.2.1 Licencias

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre.

3.2.2 Idiomas

Alemán, Árabe, Checo, Chino Simplificado, Chino Tradicional, Coreano, Español, Francés, Hungaro, Inglés, Italiano, Japonés, Polaco, Portugués (Brazil) y Ruso

3.3 DESCARGA E INSTALACIÓN

Para poder empezar a trabajar con la herramienta “Eclipse”, esta puede ser descargada de su sitio Web www.eclipse.org, esta es la página principal, después de esto debes ingresar a la zona de descargas dando un click en el siguiente botón.

Figura1: Botón hacia la zona de descarga

Ya una vez dentro de la zona de descargas se puede ver un link para descargar la última versión de “Eclipse”, que es la que nosotros elegimos:

Figura2: Link de descarga de Eclipse 3.2.1

Al dar click aquí se envía a otra página donde se muestra todos los links de donde se puede descargar este software, se indica el lugar más cercano a la ubicación que tienes para lograr que la descarga se realice más rápidamente.

Después de dar click en uno de estos sitios se puede proceder con la descarga.

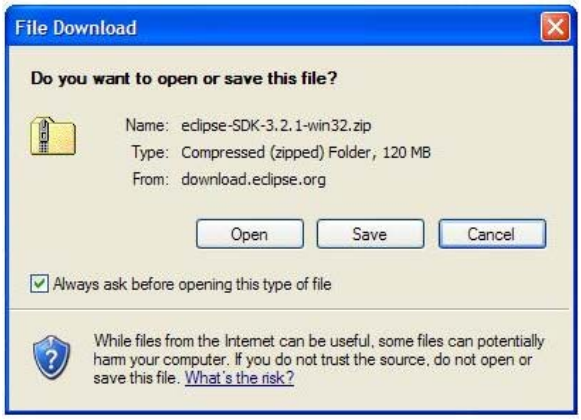

Figura: Cuadro de descarga de Eclipse

Una vez descargado el archivo eclipse-SDK-3.2.1-win32.zip lo único que queda para poder utilizarlo es descomprimirlo en la carpeta de Archivos de Programa y listo, dentro de esta se encuentra el archivo eclipse.exe con el que se puede correr el programa.

Figura: Ejecutable de la Aplicación Eclipse

Al ejecutar el programa lo primero que nos pide es establecer el espacio de trabajo o “Workspace” del proyecto, se puede abrir uno ya existente o sino crear un proyecto nuevo para empezar a trabajar en este.

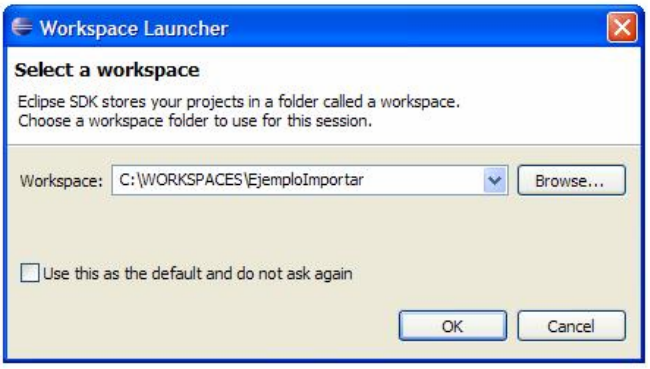

Figura3: Elección del Espacio de Trabajo (Workspace) del Proyecto

3.4 ENTORNO DE PROYECTOS DE ECLIPSE

Todo archivo se almacena dentro de un proyecto. Es decir que todo documento, carpeta, archivo de código fuente (.java) y código compilado (.class) tiene que estar contenido dentro de un proyecto.

Hay que tener completamente en claro la estructura de proyectos de Eclipse antes de usarlo para programar en Java, es preciso establecer un nuevo proyecto no sólo para desarrollar un nuevo programa de Java, sino para editar también archivos ya existentes.

3.4.1 Crear un Nuevo Proyecto

En la línea de menú principal tenemos que seleccionar:

"File > New > Project...". Es posible también seleccionar "New > Project..." dando un click derecho en cualquier parte de Eclipse (Package Explorer, Resource Navigator).

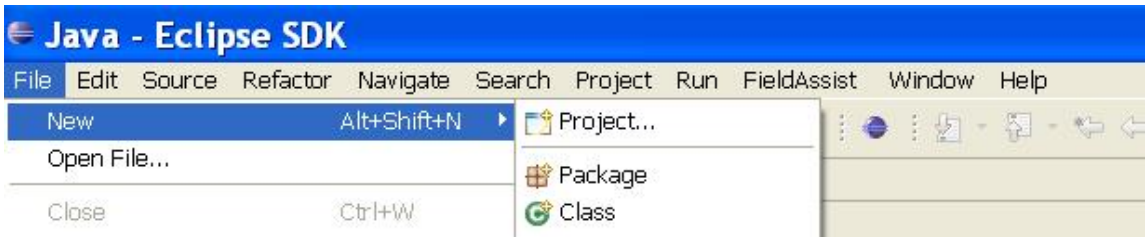

Figura4: Menú Principal

3.4.2 Tipos de proyectos de Eclipse que pueden crearse:

- Siempre un "Java Project" debería ser creado cuando se quiere crear o editar programas Java. En un proyecto de Java también puede recopilarse toda la información relacionada con el proyecto (no tiene que ser solo código fuente, también puede llevarse documentación y otros archivos que tenga una relación).
- Siempre que se creen archivos ".java" sería recomendable crear un "Java Project". Los "Simple Project" sólo deberían crearse para almacenar documentos y otros archivos, pero no código Java que se desee compilar.

- Los "Plug-in Development Project" se utilizan para sumar nuevos módulos y funciones al entorno Eclipse.
- Los proyectos de "EMF" se utilizan para crear modelos de análisis y diseño.

Para crear un nuevo proyecto:

"Java Project" seleccionando "Java > Java Project" y haciendo click en el botón "Next >" del asistente de creación.

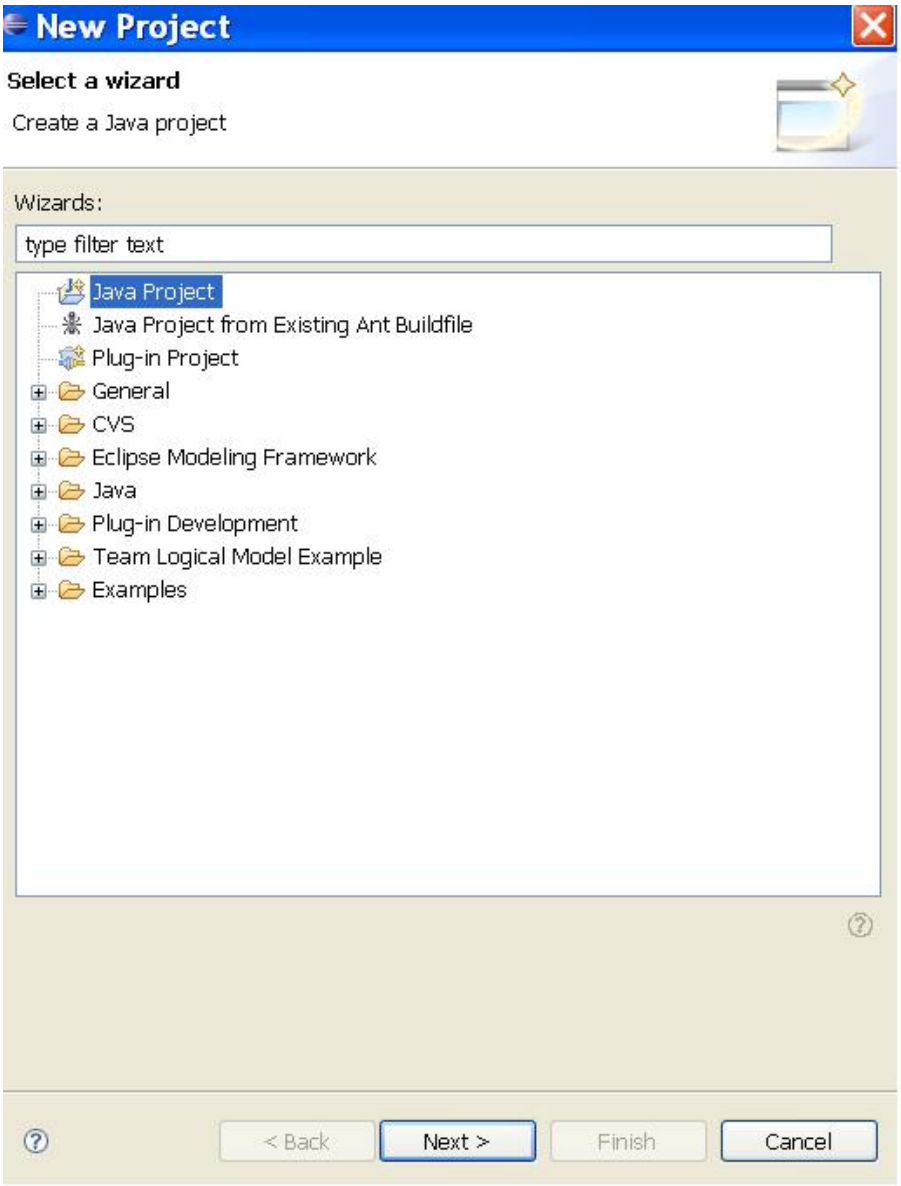

Figura 5: Creación de un Proyecto de Java.

Se debe dar un nuevo nombre para nuestro proyecto lo llamaremos “Manual”.

Si se usa el directorio por defecto, los archivos del proyecto se almacenarán en el directorio "C:MANUAL \manual". Es posible también especificar un directorio diferente en el que guardar dichos contenidos.

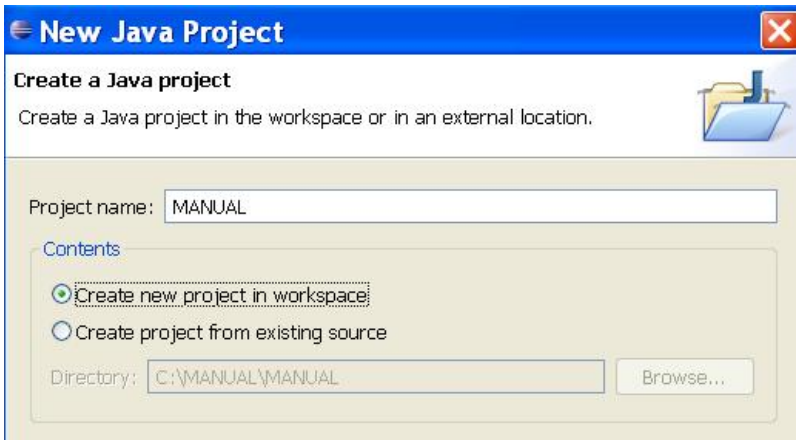

Figura6: Creación de un proyecto en espacio de trabajo

Aquí se puede terminar la creación del nuevo proyecto haciendo click en el botón "Finish", pero se recomienda pulsar el botón "Next >" para poder concretar una carpeta fuente para los archivos ".java" desde el principio.

Estas carpetas fuente ("Source Folders") acumulan los archivos de código fuente de Java (.java), de manera que Eclipse sepa donde encontrarlos y pueda realizar la compilación automática de los mismos cada vez que un archivo sea guardado.

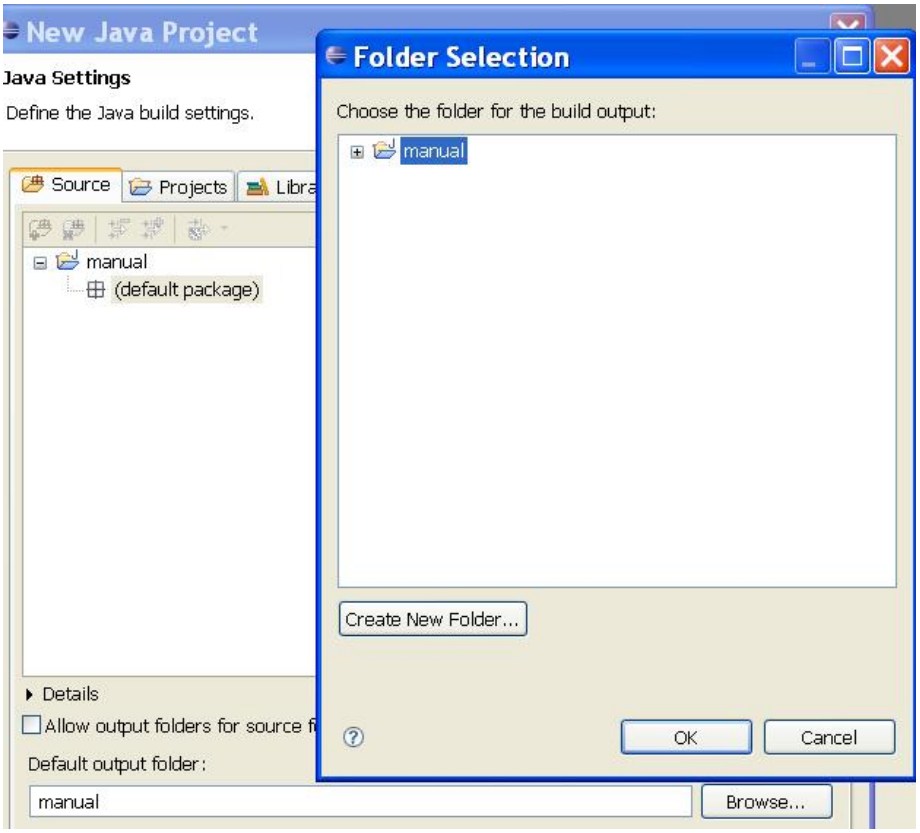

Figura7: Elección de una carpeta para el proyecto

Cuando vayamos a crear la carpeta fuente seleccionamos la pestaña de "Source" y pulsamos el botón "Browse/folder selection.". Seleccione el proyecto recientemente creado y pulse el botón "Create New Folder". Así, los archivos ".class" que resulten de la compilación de los ".java" almacenados en la carpeta fuente irán a parar a la carpeta "\bin". Pulsamos el botón "Finish" para terminar el proceso de creación del nuevo proyecto.

La Opción "Create separate source and output folders" sirve para construir de forma automática la estructura de archivos deseada.

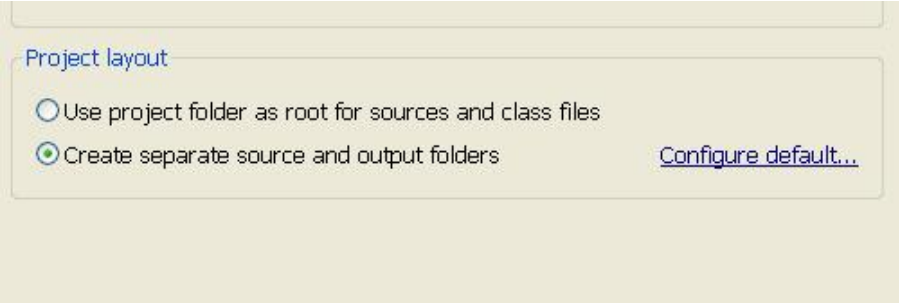

Figura8: Construir de forma automática la estructura de archivos deseada.

Si queremos editar un elemento existente debería ser primero importado dentro de un proyecto de Eclipse. Podemos hacerlo desde el menú "File > Import..." o bien dando un click con el botón derecho en cualquier punto de la vista del "Package Explorer" o del "Resource Navigator". Si queremos seleccionar un archivo o directorio hay que seleccionar "File system" en el sub menú de importación. Entonces sólo habrá que recorrer los directorios marcando los archivos que se deseen importar así como el proyecto y la carpeta destino.

Figura 9: Selección de una fuente de importación.

Todo elemento que se importa en Eclipse se duplica.

Es decir que borrar la copia que Eclipse esté manejando no borrará el archivo original, de manera que se seguirá teniendo una copia de reserva. Pero, si se usa la opción de importar un proyecto de Eclipse ya existente, los contenidos de dicho proyecto serán duplicados. Por esto hay que ser muy cuidadosos al borrar proyectos importados del entorno de trabajo de Eclipse, ya que es posible que otras copias de backup de dicho proyecto no existan.

Cualquier archivo creado mediante la utilización de Eclipse puede exportarse como un archivo normal (seleccionando Export... > File System), como ".jar" hasta archivos comprimidos en ".zip". El proceso que hay que seguir es similar de importación.

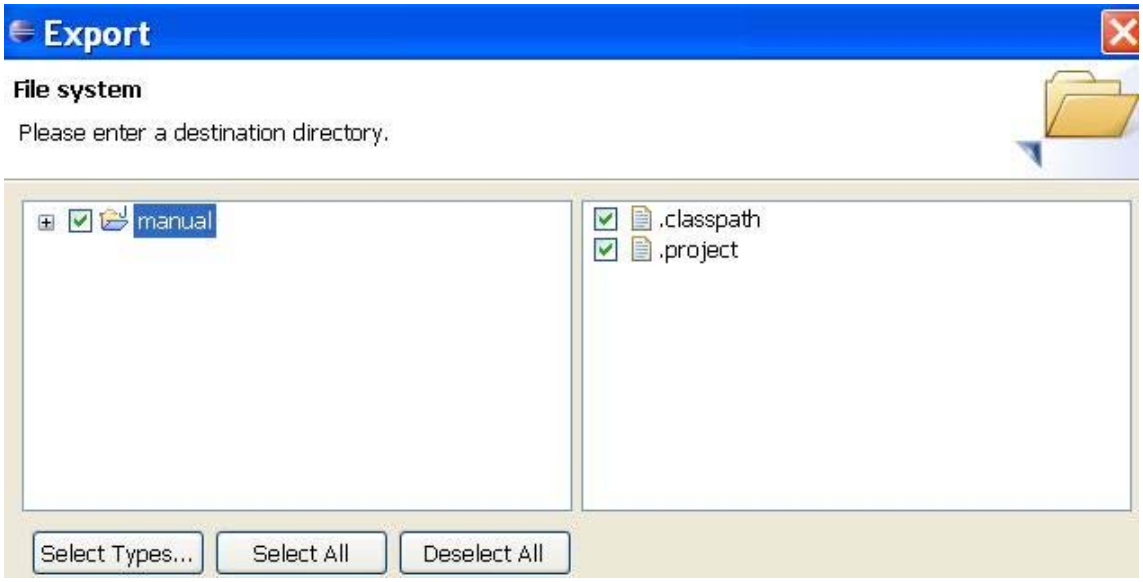

Figura10: Selección de un directorio destino.

3.5 CREAR ELEMENTOS DE JAVA

Luego de crear un nuevo proyecto, se procede a la creación de elementos de java. Para continuar con los siguientes pasos es necesario cambiar a la "Perspectiva Java. Seleccionamos "Window >Open Perspectives > Java".Contiene las vistas y editores más útiles a la hora de crear nuevos programas en Java.

Dando un click derecho en la carpeta fuente recientemente creada (por ejemplo, "src") dentro de la vista del Package Explorer, aparecerá un menú contextual. Seleccionando "New >" se mostrará una lista con los diversos elementos de Java que pueden ser creados.

Figura11: Menú contextual

Pueden ser creados también dando click sobre los iconos del menú de la parte superior de la pantalla.

Figura12: Botón para crear una nueva clase

3.5.1 Java Class

Son los archivos ".java" (código fuente) y que luego son compilados en archivos ".class". Estos archivos Java tienen que ser almacenados dentro de la carpeta fuente recientemente creada (por ejemplo, "src"). Si pulsamos "New > Class" abrimos la ventana de creación de clases.

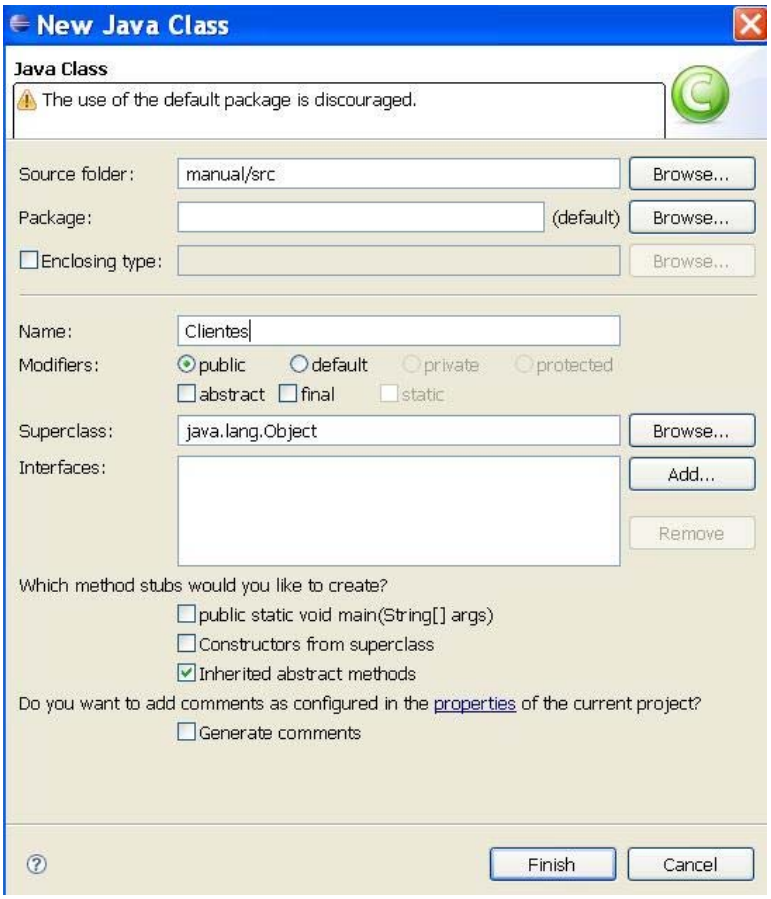

Figura13 : Nueva clase de Java

Si no se especifica ningún paquete para contener las clases Java, se guardarán dentro de un paquete por defecto. El nombre que se le da a la clase es el último campo obligatorio. El nombre de una clase debe comenzar con mayúscula. Nosotros crearemos la clase clientes.

También existen otros elementos que pueden ser fácilmente añadidos a una clase desde el mismo momento de su creación. Estos también podrían ser incrementados manualmente en otras etapas más avanzadas del proceso de desarrollo.

Cuando se quiere que la nueva clase herede de otra clase existente, se debería describir la clase "padre" dentro del campo "Superclass".

El botón "Browse..." es de gran utilidad a la hora de encontrar clases que sean posibles candidatas para ser extendidas. A pesar de que Java sólo soporta herencia única (puede heredarse solo de una única clase) es posible que una clase implemente más de una interfaz. "Browse..." simplifica la tarea de seleccionar interfaces implementadas.

Si queremos que la nueva clase contenga un método "main" (punto inicial de ejecución del programa), puede sumarse dicho método automáticamente marcando la casilla con la opción apropiada. También pueden implementarse de esta manera los constructores de la superclase y todos los métodos abstractos heredados. Esta última opción es muy práctica si se desea instanciar la clase, puesto que para esto todo método abstracto debería estar implementado.

Los archivos compilados ".class" sólo son visibles en la ventana "Navigator", abierta por defecto dentro de la perspectiva "Resource". Debido a que la perspectiva de Java no abre esa ventana por defecto, los ficheros .class no serán visibles en la vista del Package Explorer. Sin embargo, basta con escribir y guardar un archivo ".java" para que se cree un archivo ".class" resultante de compilar el archivo fuente anterior.

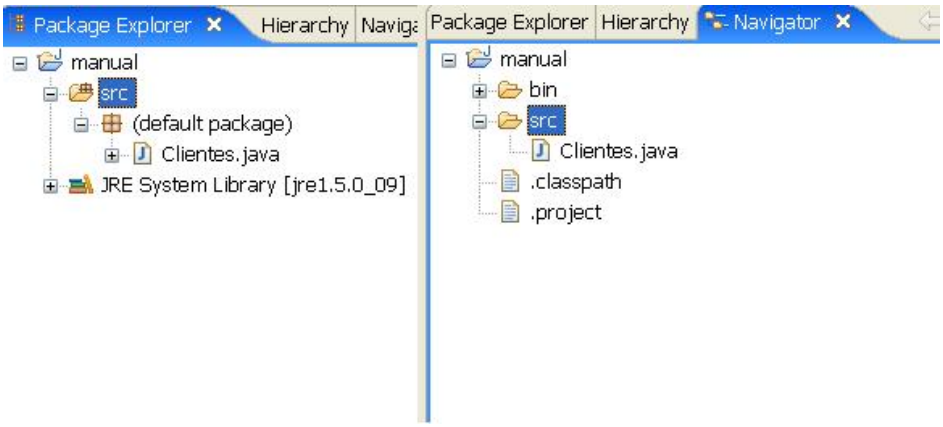

Figura 14: Package Explorer, navigator

3.5.2 File

Cuando se crean los archivos a través de este menú suelen almacenar notas e información general. Para crear un nuevo archivo pulsamos "New > File", seleccionar el proyecto y carpeta adecuados en que se desea crear el archivo, dar nombre al nuevo archivo y pulsar el botón "Finish". Por defecto, los archivos genéricos se abren en el editor de texto.

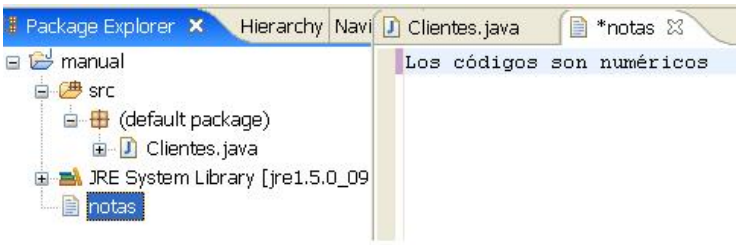

Figura 15: Crear un archivo.

3.5.3 Folder

Las carpetas se utilizan para almacenar y organizar archivos. Una carpeta normal no es exactamente lo mismo que una carpeta fuente. Si ponemos un ejemplo, en una carpeta fuente se almacenan los archivos .java con el código fuente (src), en una carpeta de salida código compilado (bin) y en otra carpeta se guarda toda la documentación relacionada (docs). Para crear una nueva carpeta basta con especificar el nombre de la nueva carpeta y la carpeta que la contiene.

Figura16: Crear una nueva carpeta

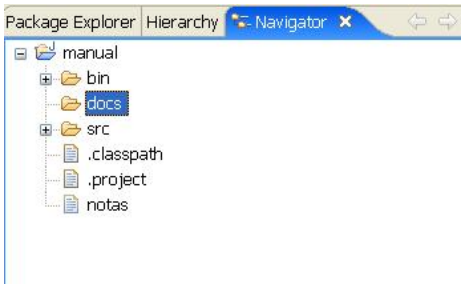

Figura: navigator

3.5.4 Interface

Decimos que las interfaces son casos específicos de las clases de Java, que carecen de implementación y que se espera que otras clases implementen. Indicando lo que la clase implementada debería hacer, mientras que los detalles de más bajo nivel corresponderían al implementador.

El procedimiento de estas es similar al de la creación de nuevas clases. Una interfaz no puede implementar ninguna interfaz, pero si puede extender otra interfaz mediante una relación de herencia.

3.5.5 Package

Los paquetes se declaran para almacenar y organizar los archivos de Java. El nombre de un paquete consta usualmente de varias partes separadas por puntos. Cada una de estas partes será un directorio nuevo dentro del sistema de archivos. Las clases que se creen dentro de un paquete determinado en Eclipse llevarán añadida automáticamente la declaración "package" en su código fuente.

3.5.6 Scrapbook Page

"Hojas de sucio" Forma sencilla de probar fragmentos de código antes de añadirlos al programa final. Sólo hay que crear una "Scrapbook Page" dentro de la carpeta deseada y escribir el código dentro de ella. No hace falta meter el código dentro de un método main para ejecutarlo.

Estas "scrapbook pages" no se muestran directamente en el menú contextual. Para crear una nueva hoja de sucio seleccionamos "New > Other > Java > Java Run/Debug > Scrapbook Page".

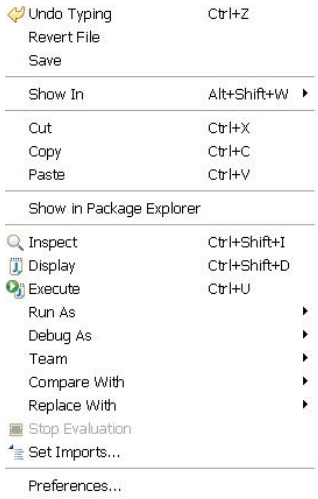

Figura16: Menú Contextual

Es necesario importar las clases usadas antes de intentar ejecutar el código añadido. Para ello basta con realizar click derecho sobre cualquier parte de la hoja y seleccionar "Set Imports" del menú contextual. Ahí es donde hay que especificar los tipos y paquetes que es necesario añadir.

Si queremos ejecutar el código recién creado es necesario seleccionarlo dando click con el botón izquierdo del ratón y arrastrando hasta tener todo el código seleccionado. Luego pulsamos el botón derecho del ratón sobre este código seleccionado y ejecutamos la opción "Execute" del menú contextual. La salida estándar de dicho proceso se mostrará dentro de la vista "Console", y otros mensajes de error se mostrarán dentro de la misma hoja de sucio.

Una vez que se ha completado la prueba pulsamos el botón "Stop Evaluation" del menú contextual.

3.5.7 Source Folder

Estas carpetas fuente son un tipo especial de carpetas destinadas a almacenar los archivos fuentes de Java (".java"). Estos archivos de código serán automáticamente compilados en archivos ".class". Todo proyecto de Java debería tener una carpeta fuente.

3.6 FUNCIONES ÚTILES DE PROGRAMACIÓN

A continuación presentaremos las funciones de ayuda a la programación de Java en Eclipse. Nos daremos cuenta de cómo usar Eclipse para programar en Java esto ahorra gran cantidad de tiempo y esfuerzo.

3.6.1 *Compilar y Detectar Errores*

Los errores de compilación se muestran en Eclipse en tiempo real subrayando el fragmento de código adecuado con una línea roja. Y además el entorno automáticamente compila los archivos salvados. Entonces no es necesario pasar por el lento proceso de compilar - observar los errores - corregir los errores.

Todos los errores pueden encontrarse fácilmente porque se muestran además como marcas rojas en el margen derecho del editor de código Java. También los errores y advertencias presentes en archivos ya guardados se muestran dentro de la vista de tareas (Tasks View), como se detallará posteriormente.

Dando un click en cualquiera de los dos tipos de marcadores de error llevará automáticamente hasta la línea en que el error está presente. Las advertencias (warnings) se muestran de la misma manera, pero con marcas amarillas.

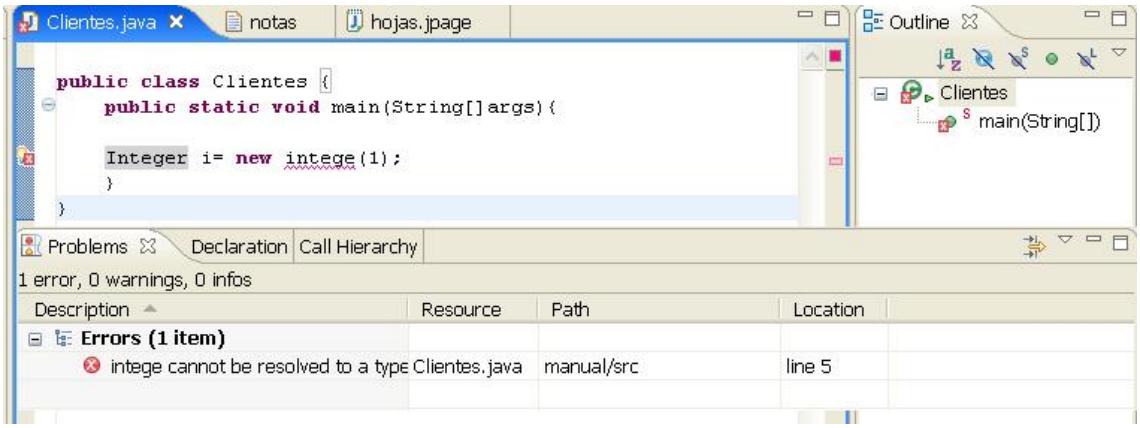

Figura17: Compilar y Detectar Errores

3.6.2 Icono de Bombilla = Auto corregir

Eclipse detecta y marca todo error y advertencia de compilación. Permite auto corregir los posibles errores haciendo click en el icono de bombilla presente en el margen izquierdo del editor de código. Aparecerá una ventana mostrando todas las opciones. Seleccionamos una opción mediante los cursores del teclado o dejamos la punta del ratón sobre dicha opción esto abrirá una nueva ventana mostrando detalladamente las modificaciones de código que la auto corrección efectuaría. Basta con pulsar la opción seleccionada (o pulsar ENTER) para hacer que Eclipse lleve a cabo la corrección automatizada.

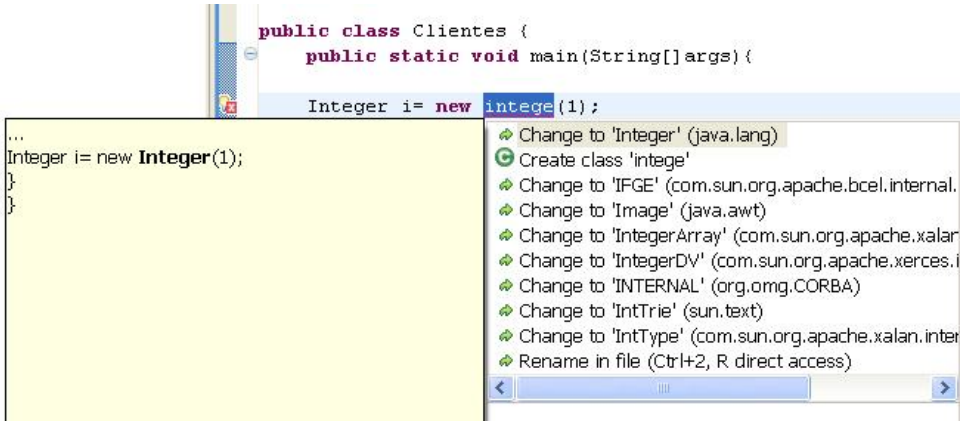

Figura18 : Auto corregir

3.6.3 *CTRL + Espacio = Auto completar*

Esta función de ayuda a la programación es muy útil.

- Nombres de Clases

Cuando se crea referencias a otras clases dentro de la clase actual es visto esto como una tarea de programación habitual. Pero algunas clases de Java tienen nombres muy largos que son difíciles de recordar. Además, es necesario añadir declaraciones de importación para poder resolver dichas referencias a clases a la hora de compilar.

"CTRL + Espacio" tras escribir los primeros caracteres del nombre de una clase Java mostrará las posibles alternativas. Puede seleccionar cualquiera de ellas simplemente realizando click izquierdo del ratón. La sentencia de importación correspondiente se añadirá de forma automática. Las clases se marcan con una "C" verde mientras que las interfaces se marcan con una "I" morada. El paquete al que pertenece la clase se muestra también, permitiendo de este modo evitar posibles confusiones.

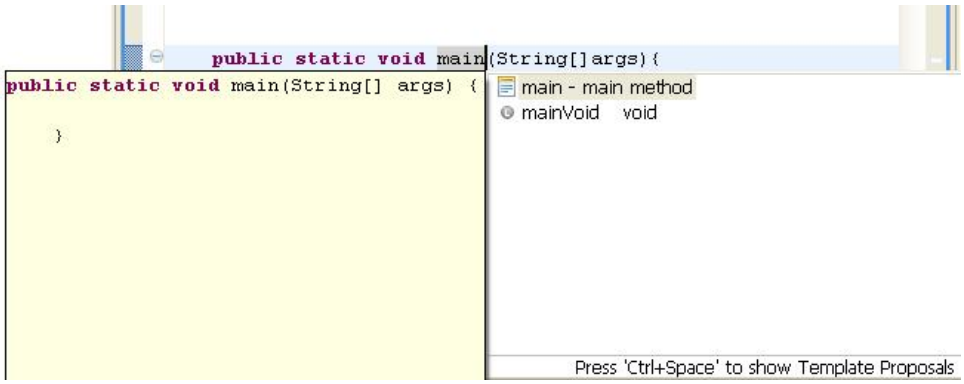

Figura19: ctrl.+space

- Atributos y Variables Locales

Si definimos una clase es normal dar nombres inventados a sus atributos y a las variables internas de los métodos. Pero en ocasiones resulta difícil recordar el nombre exacto. Tras escribir los primeros caracteres del atributo o de la variable local, pulsar "CTRL + Espacio" mostrará las posibles alternativas. Este proceso es muy similar al de auto completar el nombre de las clases recientemente expuesto. Las variables locales se marcan con el icono de una "L" gris, mientras que los atributos se marcan con un icono que puede variar según la visibilidad del atributo.

Figura20: Marcas en variables y atributos

- Métodos y Constructores

Cuando se haya creado un objeto Java pueden llamarse los métodos correspondientes a su clase. Sin embargo, es bastante habitual olvidar el nombre de un método en concreto, o incluso los tipos de sus parámetros y su orden. Este problema puede solucionarse fácilmente pulsando "CTRL + Espacio" tras escribir el nombre del objeto seguido de un punto, lo cual mostrará una ventana con las posibles alternativas. Pulsar sobre la alternativa escogida añadirá la signatura del método al objeto.

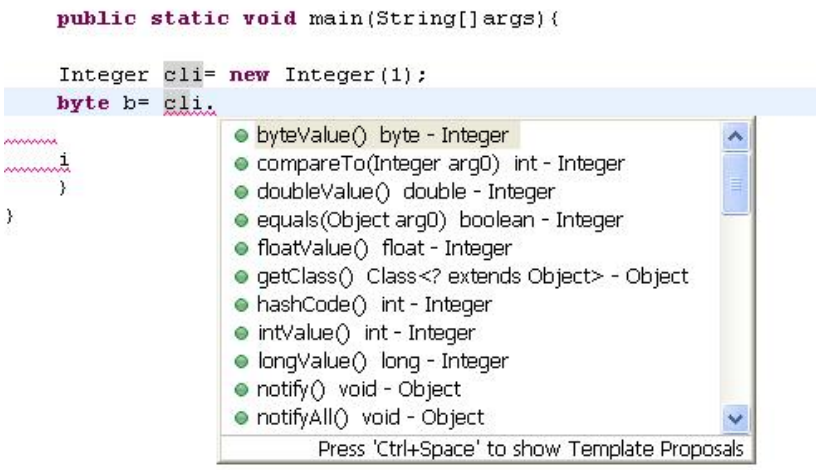

Figura21: Métodos y Constructores

Se puede también auto completar la marca de los constructores pulsando "CTRL + Espacio" tras escribir (o auto completar) el nombre de la clase seguido de un signo de apertura de paréntesis, "(".

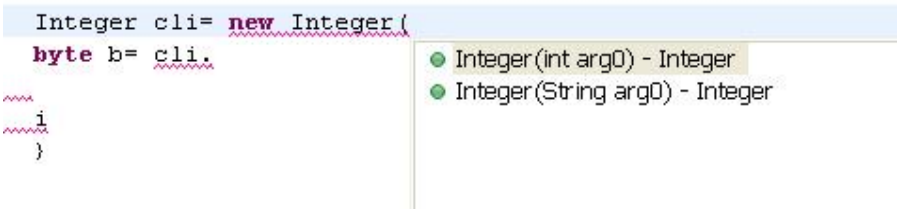

Figura 22: Auto completar

Cuando escribamos las primeras letras del modificador de un método tal como "public" o "private", "CTRL + Espacio" le permitirá crear automáticamente una plantilla del método. Pulsar el tabulador permite saltar de un campo de la plantilla a otro, de manera que se pueda completar el tipo de retorno, el nombre del método y sus parámetros.

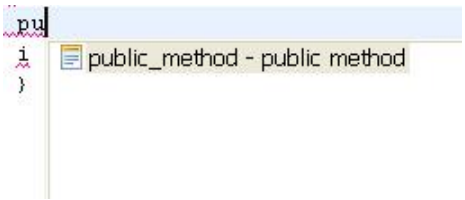

Figura 23: Auto completar

- Bucles

Estos están presentes en todos los programas. A pesar de que crear un bucle no es una tarea compleja, Eclipse proporciona algunas funciones de auto completado que pueden acelerar considerablemente el proceso. Basta con escribir "do", "Chile" o "Foz" y pulsar "CTRL + Espacio" para mostrar las posibles opciones. Si el bucle ha sido creado con el propósito de recalcar sobre un arría de elementos, seleccionar esta opción intentará auto completar incluso el nombre del array.

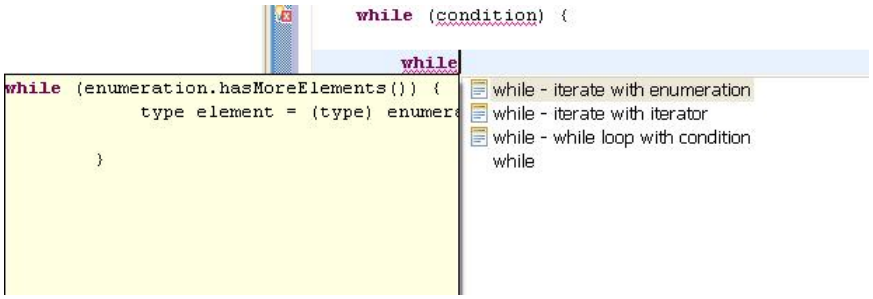

Figura 24: Bucles

- Etiquetas de Javadoc

Los comentarios internos del programador se indican con una "//", los comentarios de Javadoc se inician con un "/**". Tras crear un método, añadir "/** + ENTER" sobre la signatura del método auto completará información de. Pulsar "CTRL + Espacio" dentro de un bloque "/** ... */" mostrará toda la lista de etiquetas Javadoc posibles.

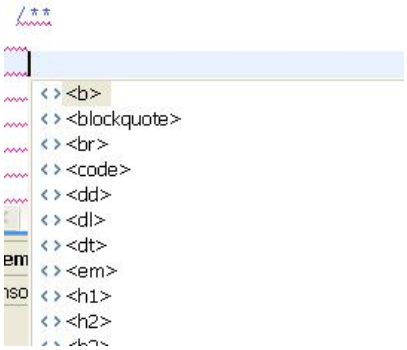

Figura 25: Etiquetas de Javadoc

3.6. 4 *Menú "Source"*

Si damos un click derecho en el editor de código se mostrará un menú contextual. Las funciones más importantes de su submenú "Source >" son las siguientes:

- Toggle Comment

Con esto podemos cambiar el estado del código seleccionado, pasándolo de comentario a código o viceversa.

- Add Block Comment

Esto se encargará de marcar el código como comentario rodeándolo con los símbolos "/*" y "*/". Mientras que seleccionar un bloque de código comprendido entre esos símbolos de comentario hará que en aparezca la opción "Remove Block Comment", la cual eliminaría los símbolos de comentario.

- Format

Esta función de formateado automático de código automáticamente posiciona el código de la forma adecuada, además de llevar a cabo otras funciones de representación. Es una forma rápida de conseguir tener un código ordenado y comprensible.

Permitiendo especificar opciones avanzadas de formateo de código. La página de preferencias que permite configurar el formateo de código se ha trasladado a "Window > Preferences > Java > Code Style > Code Formatter". Pese a que las opciones se pueden configurar de acuerdo a las preferencias personales, ya vienen configuradas por defecto para cumplir las convenciones de Java.

- Organise and Add Imports

A pesar de que las sentencias de importación adecuadas se muestran siempre cuando se usan las funciones de auto completar código para completar el nombre de una clase Java, nuevas sentencias de importación pueden añadirse en cualquier momento usando la función "Add Import". Si pulsamos "Organise Imports" eliminamos automáticamente todas las declaraciones de importación no utilizadas, incrementando la eficiencia del código. El método abreviado del teclado es "CTRL + Mayúsculas + O".

Figura: Organizar y Añadir Importes

- Override and Implement Methods

Seleccionando esta opción de sombrear o implementar métodos abrirá una ventana de menú en la cual se podrán marcar los métodos de la superclase cuya signatura se desea que se añada.

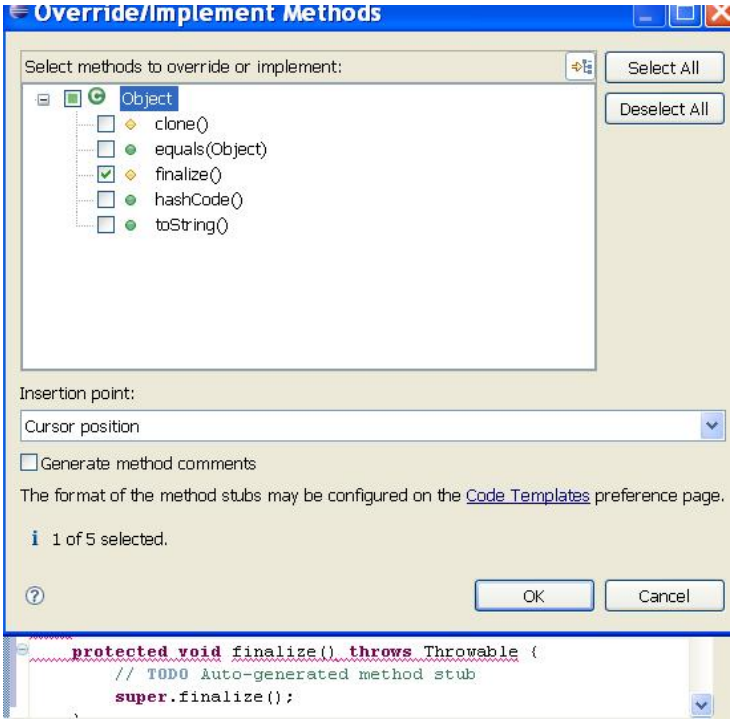

Figura 26: Métodos de Implementación

La opción "Add Constructors from Superclass" permitirá especializar todos los constructores usados.

- Generate Getter and Setter

Java permite especificar diferentes niveles de visibilidad de atributos. A pesar de esto, en programación orientada a objetos, los atributos internos de una clase deberían ser siempre privados. Esto quiere decir que no debería poder realizarse ningún tipo de modificación directa del atributo de una clase desde otra clase externa. A causa de esto, la única manera de acceder a un atributo privado para leer su valor o bien para darle un nuevo valor sería utilizando un método modificador que sea público. Seleccionando "Generate Getter and Setter" una ventana mostrando los posibles métodos que podrían crearse de acuerdo con los atributos definidos aparecerá. Entonces los métodos necesarios podrían crearse simplemente seleccionándolos y pulsando "Ok".

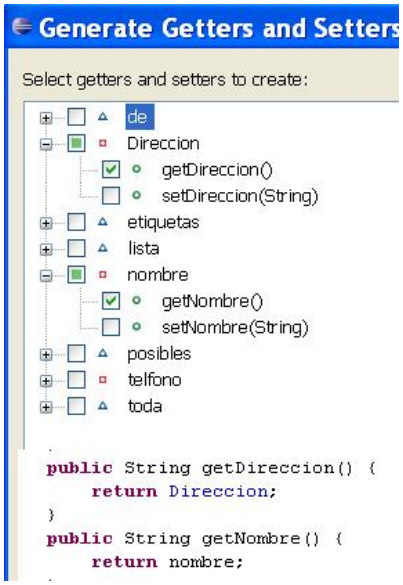

Figura 27: Niveles de Visibilidad

- Surround with try/catch block

Para el uso adecuado de esta opción es necesario tener seleccionado de antemano un fragmento de código dando un click con el botón izquierdo del ratón (o pulsando Mayúsculas) y arrastrando. Activar esta opción creará un bloque "try" alrededor del código seleccionado. Tras este bloque se añadirán automáticamente los bloques "catch" adecuados, los cuales atraparán toda posible excepción que el código rodeado pueda lanzar. Por defecto se añade una sentencia de traza dentro de esos bloques "catch" de manera que sea posible identificar inmediatamente dónde se lanzó la excepción.

socket = new Socket (“localhost”, 100);

Cuando una excepción no es atrapada, aparecerá como texto en rojo (de la salida de error estándar) en la vista "Console". Pulsar la línea de código se muestra en qué línea tuvo lugar la excepción llevará directamente a ese punto del programa en el editor de código.

```
try {
        socket= new Socket("localhost",100);

} catch (Exception e) {
    // TODO: handle exception
}
```

3.6.5 *Refactor Menu*

Dando un click derecho en el editor de código mostrará el menú contextual. A continuación se muestran las funciones más interesantes del sub menú "Refactor >".

- Rename

Para invocar la función de renombrado hay que tener previamente seleccionado un elemento.

"Update references" actualizará toda referencia al nuevo elemento renombrado. Usando esta opción de "Refactor > Rename..." es como deberían renombrarse todos los elementos incluyendo los archivos ".java". Así se actualizan todas las referencias no aparecerán problemas a la hora de compilar. Al renombrar determinados elementos será posible actualizar también referencias en comentario de Javadoc, comentarios normales y cadenas de caracteres entrecomilladas, lo cual también puede resultar bastante útil. La opción de "Preview" permite asegurarse de que no habrá ningún tipo de error durante la operación de renombrado.

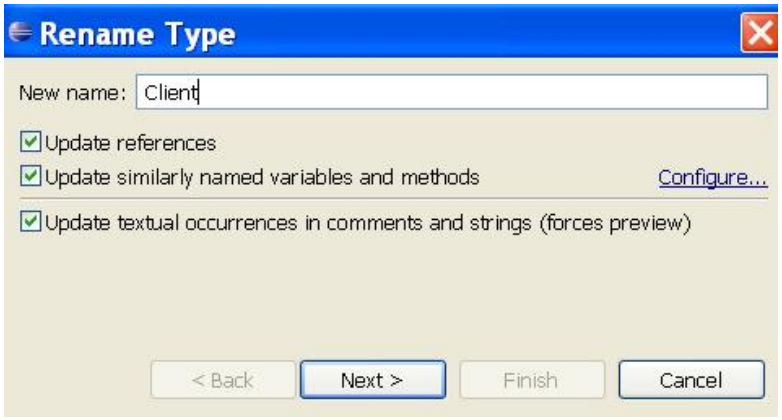

Figura28: Renombrar elementos

- Move

Antes de seleccionar "Refactor > Move...", el archivo fuente o elemento que se desea mover deberá haber sido seleccionado. Entonces será sólo necesario seleccionar el destino de manera que se lleve a cabo la operación de mover. Esta es la forma correcta de mover archivos ya que evita problemas futuros con referencias y rutas de compilación.

- Change Method Signature

Si se desea modificar un método marcado es posible usar esta opción en lugar de hacerlo manualmente. Hay que colocar el cursor dentro del método cuya marca se desea cambiar. Esta es una forma rápida de cambiar la visibilidad, el tipo de retorno, los parámetros y su orden. Los nuevos parámetros se añaden pulsando el botón "Add" y se modifican pulsando en botón "Edit".

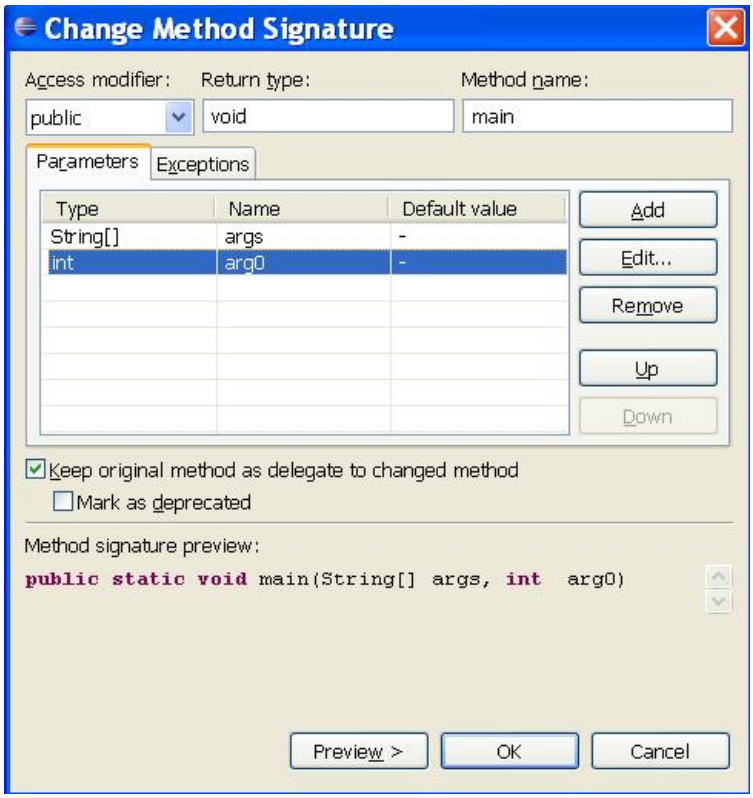

Figura29: Modificar método marcado

- Pull Up and Push Down

Si la clase actual extiende o es extendida por otra clase, puede ser interesante mover algunos elementos a la superclase (pull up) o a la subclase (push down) respectivamente. Seleccionar el elemento y la opción adecuada llevará a cabo esta operación de forma automatizada.

3.6.6 *Consultar la Documentación*

La documentación Javadoc del código que se esté actualmente programando puede ser consultada en tiempo real simplemente colocando el cursor o el puntero del ratón sobre el elemento elegido. Para expandir la ventana con esta documentación basta con pulsar la tecla de función F2.

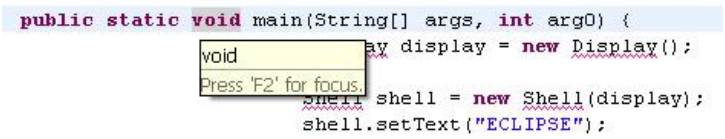

Figura 30: Consultar Documentación

También se dispone de una nueva vista de Javadoc ("Window > Show View... > Java > Javadoc"). Dicha vista muestra la documentación Javadoc asociada al elemento sobre el que está situado el cursor.

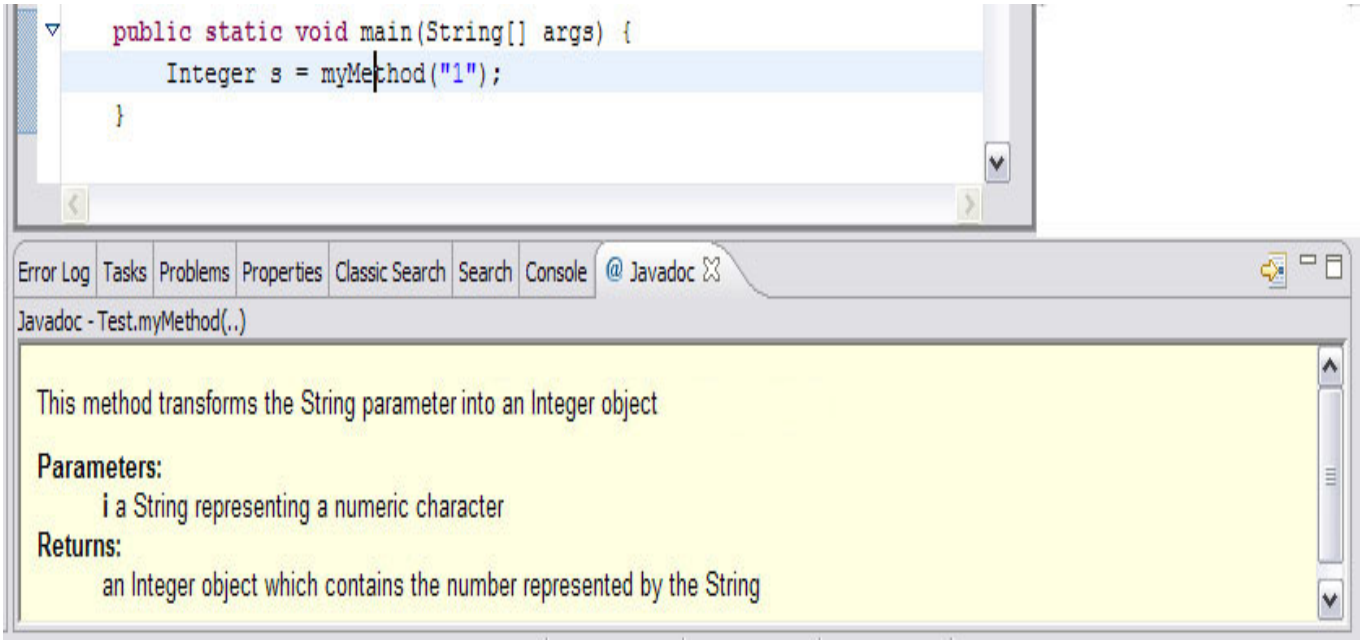

Figura31: Documentación de JavaDoc

3.6.7 *Importar Archivos JAR*

Puede ser necesario importar algunos archivos Jar no incluidos por defecto en el JRE estándar para que el proyecto pueda compilar. Basta con pulsar el botón derecho del ratón sobre la carpeta adecuada, elegir "Properties > Java Build Path", seleccionar la pestaña "Libraries", pulsar el botón "Add External Jars" y seleccionar el archivo ".jar" o ".zip". El nuevo Jar añadido será visible en la ventana Package Explorer como un pequeño frasco.

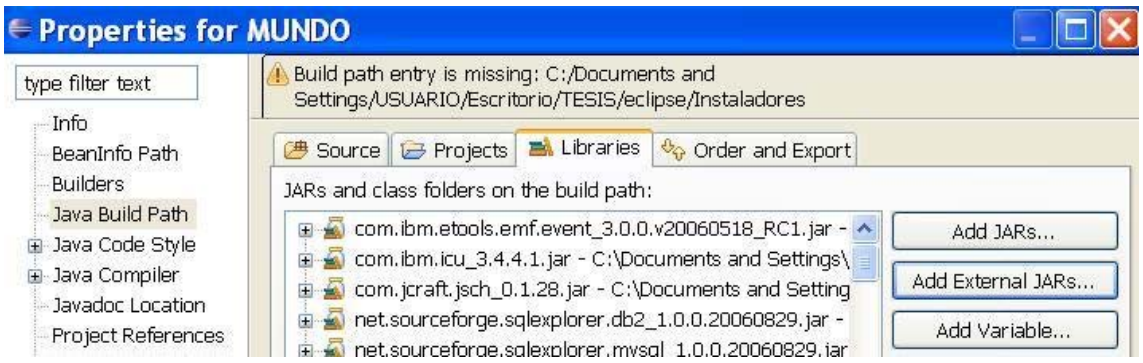

Figura 32: Propiedades

3.7.1 VISTAS DE ECLIPSE

Dos tipos de elementos integran lo que es la interfaz de usuario: vistas y editores. Los editores permiten realizar una tarea completa, las vistas proporcionan funciones de apoyo.

Describiremos las vistas más interesantes de Eclipse a la vez también describiremos algunos consejos de cómo navegar a través de los editores.

3.7.2 *Perspectivas*

En Eclipse una perspectiva es una agrupación de vistas y editores de manera que den apoyo a una actividad completa del proceso de desarrollo software.

Es posible crear perspectivas propias añadiendo nuevas vistas y cambiando su distribución en la pantalla. Las perspectivas pueden seleccionarse eligiendo "Window > Open Perspective" del menú. Las perspectivas son:

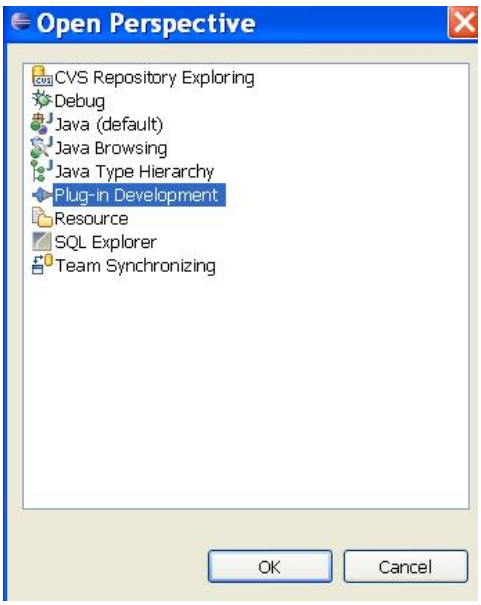

Figura 33: Abrir Perspectiva

Resource

Relacionada estrechamente con el sistema de archivos puesto que representa la localización física de los recursos almacenados dentro de los proyectos

Java

Centrada en tareas de programación, mostrando paquetes, clases, métodos y atributos en sus vistas asociadas.

Plug-in development

Permite a los desarrolladores añadir nuevos módulos de Eclipse.

Install/Update

Permite gestión de la configuración. Muestra los componentes instalados así como sus versiones y conflictos.

Debug

Relacionada con la tarea de depuración. Se centra en los procesos ejecutados, puntos de ruptura, variables, salida, etc.

Java Browsing

Permite ojear rápidamente código, proyectos, paquetes y jerarquías.

Los iconos de perspectiva se encuentran en la esquina superior derecha.

Existe un botón etiquetado como "Open a Perspective" que permite acceder rápidamente a otras perspectivas. Otro cambio es que la perspectiva "Install/Update" ha sido eliminada, y puede accederse a sus funciones seleccionando "Help > Software Updates".

Figura 34: íconos de Perspectiva.

3.7.1.2 *Tareas*

La vista de tareas ("Tasks View") permite una rápida gestión de tareas pendientes. Seleccionando "Window > Show View > Tasks" se muestra esta vista. Pueden añadirse nuevas tareas haciendo click en el botón "Add task".

La prioridad de la tarea y su estado también pueden modificarse sin más que hacer click en dichos campos. También los errores y las advertencias de los archivos con código guardados se muestran en esta vista. Haciendo click en la descripción de un error llevará hasta el punto exacto del código en que se encuentra dicho error.

Dando un click derecho en cualquier punto de la vista de tareas se mostrará un menú contextual que permitirá realizar de forma rápida cualquier actividad relacionada con la gestión de las tareas definidas.

3.7.1.3 *Navigator*

Esta ventana del navegador de recursos permite echar un vistazo a la estructura de archivos de los proyectos definidos.

Esta vista es la única que muestra la carpeta de salida ("bin") así como los archivos Java compilados (".class").

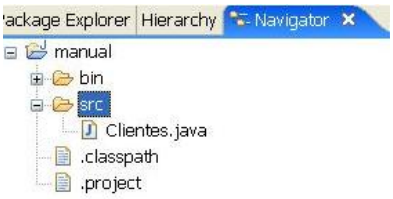

Figura 35: Carpeta Bin

3.7. 1.4 *Package Explorer*

La vista del explorador de paquetes muestra la estructura lógica de paquetes y clases Java almacenados en los distintos proyectos.

Las carpetas fuente (que deben almacenar los archivos fuente ".java") se muestran decoradas con el icono de un paquete contenido. Los archivos Java también pueden ser expandidos de modo que muestren sus métodos y atributos internos al pulsar el botón "+".

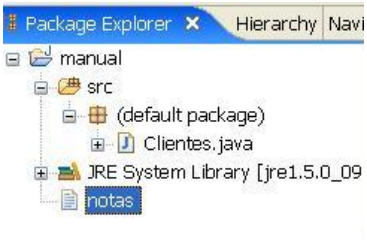

Figura 36: Estructura lógica de paquetes

3.7. 1.5 Working Set

Un conjunto de trabajo es un grupo de elementos que se muestran en las distintas vistas de eclipse. Estos conjuntos de trabajo se usan como filtros que permiten separar claramente los diferentes proyectos en que se está trabajando.

Esto es muy útil cuando se está trabajando simultáneamente en varios proyectos no directamente relacionados entre sí. Organizar los distintos proyectos en grupos de trabajo acelerará el proceso de buscar los elementos deseados y reducirá la confusión visual.

Para definir un conjunto de trabajo, basta con pulsar en el icono de menú del Package Explorer (el icono de un triángulo invertido) y seleccionar "Select Working Set". Aquí se permitirá nombrar un nuevo conjunto de trabajo, así como seleccionar sus recursos relacionados y editar o quitar otros conjuntos de trabajo existentes. Todos los conjuntos de trabajo disponibles se muestran directamente la próxima vez que se pulse el icono triangular de menú.

Figura 37: Nombrar nuevo conjunto de trabajo

Si creamos un nuevo proyecto cuando un conjunto de trabajo está siendo usado hará que el nuevo proyecto no se muestre dentro de las vistas de Eclipse. Para poder ver el proyecto recién creado, será necesario editar el conjunto de trabajo actual ("Menú de la vista > Select Working Set > Edit" o directamente "Edit Current Working Set") y seleccionar el nuevo proyecto para que se muestre.

3.7. 1.6 *Outline View*

La vista de resumen es una forma rápida de ver qué métodos o atributos se encuentran definidos dentro de una clase de Java. Los iconos asociados proporcionan información adicional de acuerdo con la visibilidad del atributo o método en cuestión. Y sólo con hacer click en cualquiera de estos iconos conducirá a la línea de código exacta en que dicho atributo o método está definido. La vista de resumen es una herramienta esencial para entender y navegar archivos Java voluminosos.

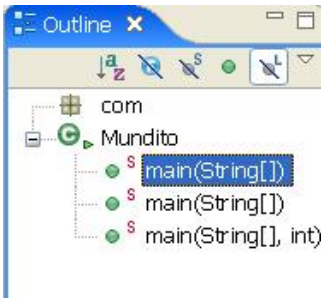

Figura38: Métodos o atributos definidos dentro de una clase de java.

3.7. 1.7 Hierarchy View

La vista de jerarquía muestra las relaciones de herencia presentes entre distintos elementos de Java. Haciendo click derecho en el nombre de una clase Java en el editor de código y seleccionando "Open Type Hierarchy" abrirá esta vista de jerarquía. La tecla rápida asociada es "F4"

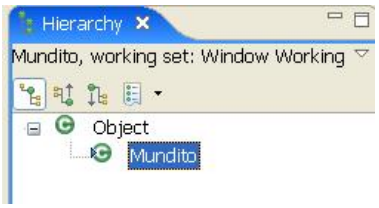

Figura 39: Vista de Jerarquía

Existe la opción "Open Call Hierarchy" al menú contextual del editor de código. Tras seleccionar un método, al hacer click en esta opción se abrirá una vista que mostrará dónde es usado dicho método. Las teclas rápidas asociadas son "CTRL + ALT + H".

Fast Views

Arrastrar una vista hasta el margen izquierdo (hasta que aparezca un icono de carpetas apiladas) convierte esta vista en una "vista rápida". Pulsar el icono de la vista rápida hará que dicha vista se muestre, mientras que volver a pulsarlo (o pulsar en cualquier otro punto de la pantalla) hará que se oculte. Mediante un click derecho en el icono de la vista rápida y seleccionando "Fast View" restaurará la vista a su posición original.

Es un pequeño rectángulo situado en la esquina inferior izquierda de la pantalla. Así pues, las vistas rápidas se crean ahora arrastrando la vista dentro del rectángulo hasta que aparece un icono de una flecha dentro de un cuadrado. No obstante, la zona en la que se almacenan las vistas rápidas puede cambiarse de sitio colocando el cursor sobre ella hasta que se transforma en un cursor con cuatro flechas, arrastrando y depositando la zona en el lugar deseado.

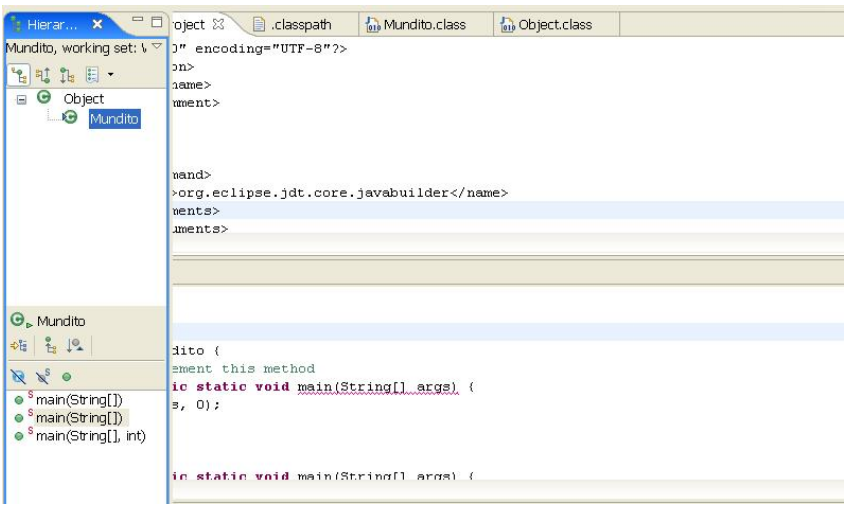

Figura40: Vistas Rápidas

3.7.1.9 Search View

Para realizar una búsqueda dentro de Eclipse, el menú "Search" de la barra superior de menús debería ser seleccionado. También se pueden lanzar búsquedas pulsando el icono de linterna.

Hay varios tipos de búsquedas dentro de Eclipse.

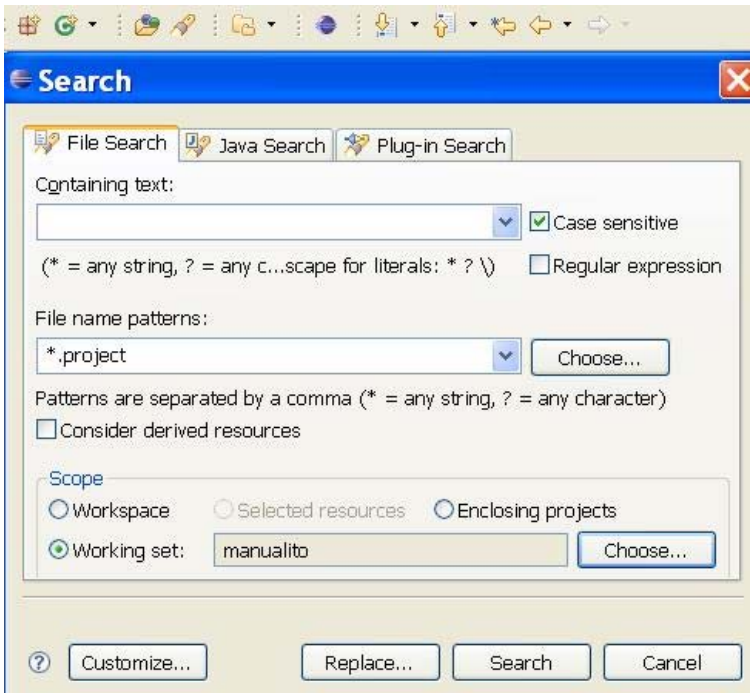

Figura 41: Búsqueda

- La búsqueda de archivos "File Search" es una búsqueda textual que puede ser ejecutada sobre archivos de todo tipo. Es equivalente a una búsqueda tradicional.
- La búsqueda de ayuda "Help Search" efectúa búsquedas dentro de la ayuda de Eclipse.
- La búsqueda de Java "Java Search" es similar a la búsqueda de archivos pero proporciona funciones adicionales para buscar en archivos Java. Permite buscar explícitamente por tipos, métodos, paquetes, constructores y campos, usando restricciones de búsqueda adicionales (como por ejemplo, buscar sólo el punto del código en que se declararon los elementos coincidentes).

Es importante comprobar que la búsqueda se efectúa sobre los ficheros apropiados. Esto puede definirse usando el campo "scope". "Workspace" hace referencia al entorno de trabajo completo. "Selected Resources" son sólo los archivos seleccionados (es posible seleccionar más de un archivo haciendo click izquierdo en ellos mientras se mantiene pulsada la tecla CTRL). "Working Set" es un conjunto de trabajo previamente definido.

Los resultados de búsqueda se muestran como un árbol jerárquico.

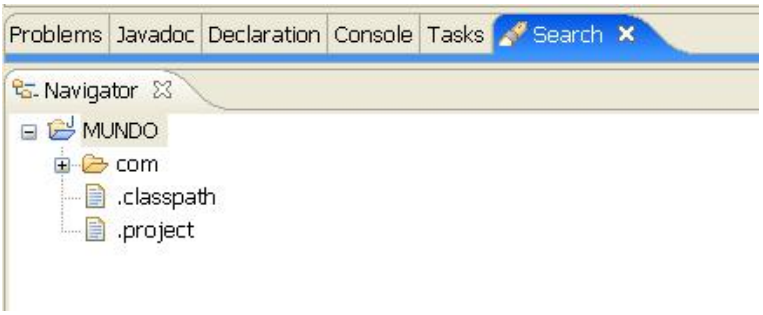

Figura42: Resultados búsqueda

3.7.2 *Navegar por las Vistas y los Editores*

Esto fue una introducción de cómo utilizar las vistas de Eclipse y cómo dichas vistas ayudan a manejar la información.

A continuación explicaremos algunas funciones de navegación adicionales que serán útiles para encontrar rápidamente la información deseada y que permitirán presentarla adecuadamente en los diversos editores y vistas.

3.7.2. 1 Maximizar una Vista o Editor

Haciendo doble click en el título de una ventana para maximizarla. Doble click en el título de nuevo hará que las dimensiones y posición de la ventana sean restauradas a las que tenía originalmente.

"CTRL + M" como tecla rápida asociada a maximizar o restaurar la ventana del editor actual.

3.7.2.2 Ir al Último Cambio

El icono del menú representado como "una flecha con un asterisco" sirve para colocar el cursor en el último punto del código que fue modificado dentro del editor activo. Es habitual que tras cambiar algo de código (por ejemplo, tras escribir algunas instrucciones nuevas) movamos el cursor a otra línea para revisar otra parte del programa. Si deseáramos volver al punto en que añadimos el último cambio (que suele ser el lugar por el que "íbamos programando") tendríamos el problema solucionado con sólo pulsar este icono de "ir al último lugar editado".

Las teclas rápidas asociadas son "CTRL + Q".

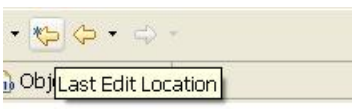

Figura 43: Última ubicación editada

3.7.2.3 Acciones de Navegación de los Editores

Qué se debería hacer si queremos volver a un punto del programa en el que no introdujimos ningún cambio (es decir, en el que situamos el cursor pero en que no escribimos o borramos ningún carácter)?

Y si quisiéramos regresar al lugar en que estuvimos justo *antes* de editar algo en otro lugar? Las flechas de navegación del menú resolverán estos problemas. Basta con pulsar la flecha de "navegar hacia atrás" para regresar a puntos previamente visitados del programa.

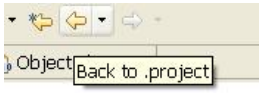

Figura 44: Regreso al proyecto

Y pulsando la flecha de "navegar hacia delante" recorreremos el historial de lugares visitados hacia los puntos más recientes.

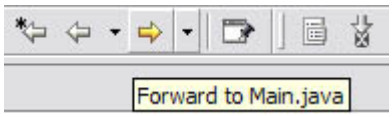

Figura 45: navegar hacia adelante

De hecho, estas útiles acciones funcionan de forma muy similar a como lo hacen los botones de "atrás" y "adelante" de un navegador web.

La opción de "atrás" sólo se activa si existen puntos de programas que se visitaron previamente.

La opción de "adelante" se activa tras haber pulsado el botón de "atrás". También hay que tener en cuenta que pulsando en el pequeño triángulo negro que se encuentra junto a las flechas de navegación desplegaremos un menú que muestra otros archivos (distintos del abierto en la ventana activa del editor) en los que se encuentran otros puntos visitados accesibles.

Para cambiar la ventana activa del editor a otras ventanas abiertas existe un método abreviado: "ALT + F6" (comando de "siguiente editor").

Las útiles teclas rápidas asociadas a estas acciones de navegación son "ALT + IZQUIERDA" para navegar hacia atrás y "ALT + DERECHA" para navegar hacia delante.

3.7.2.4 Revisar Problemas

Los botones de "Ir al siguiente/anterior problema" permiten recorrer uno tras otro los problemas pendientes que aparecen en el editor actual.

Estos botones de "ir a problema" se han sustituido por botones de "ir a anotación". Haciendo click en el pequeño triángulo negro cercano a estas flechas de navegación por anotaciones se abrirá una lista editable con los tipos de anotaciones que serán recorridas.

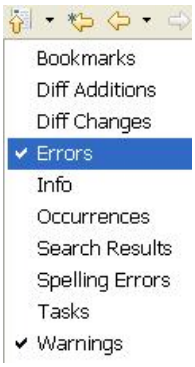

Figura 46: Revisar Problemas

3.8 EJECUTAR Y DEPURAR

Si el programa de Java se encuentra completo será hora de ejecutarlo y probarlo. Quizás aparezcan algunos "bugs" a la hora de ejecutarlo. Entonces será hora de depurar el programa. Eclipse proporciona ayuda a las tareas de ejecutar y depurar código.

3.8.1 *Ejecutar*

Para ejecutar un programa dentro de Eclipse hay que seleccionar "Run > Run..." del menú principal. Dentro de "Configurations" se almacenan diferentes configuraciones de ejecución. Hay cuatro tipos de configuraciones de ejecución:

- Java Applet (para applets web)
- Java Application (para programas normales de Java)
- JUnit (casos de prueba)
- Run-Time Workbench (otras instancias de Eclipse que permiten probar nuevos módulos de Eclipse).

Figura 47: Ejecutar Depurar

Para ejecutar un programa de Java normal debería seleccionarse "Java Application" y pulsar el botón "New" para crear una nueva configuración. Dentro de la pestaña "Main" hay que dar nombre a la nueva configuración seleccionar el proyecto que contiene la clase con el método main y seleccionar dicha clase. El método "main" es el punto de ejecución de un programa Java, y se representa como un pequeño icono de un hombre corriendo al lado del icono de la clase.

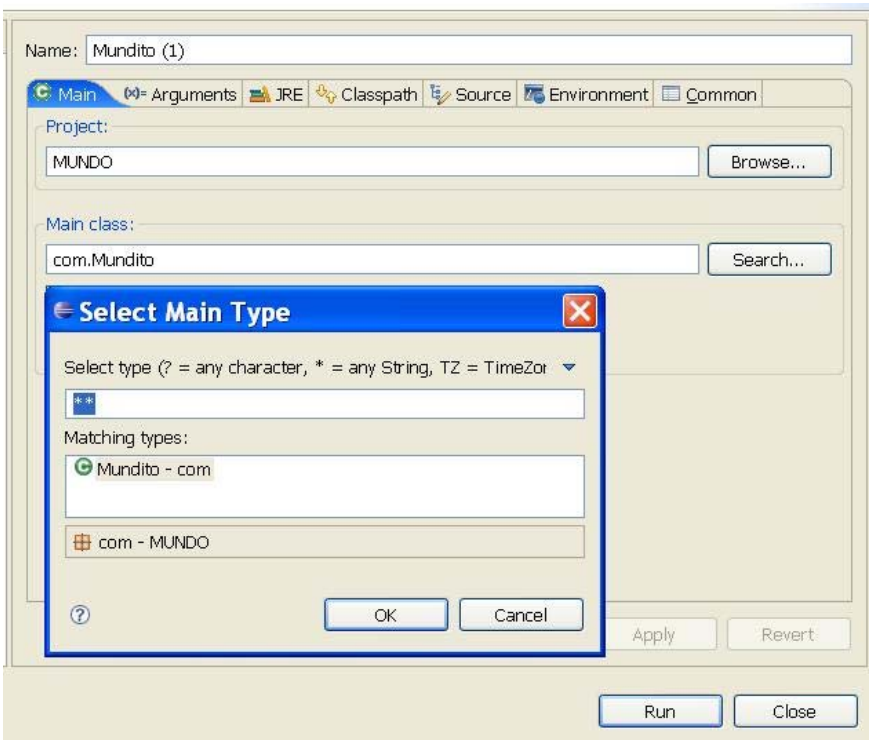

Figura48: Seleccionar un tipo principal

Cuando se desea pasar argumentos al método main (en la forma de "String[] args"), no hay más que hacer click en la solapa de "Arguments" y escribir esos argumentos separados por espacio dentro de la zona en blanco de "Program Arguments".

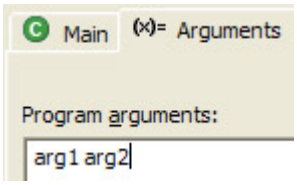

Figura 49: Pasar argumentos al método main

Y para terminar, hacer click en el botón "Run" lanzará la ejecución del programa seleccionado. Una forma más rápida de arrancar la ejecución del programa recientemente ejecutado es pulsar en el icono de una flecha blanca dentro de un círculo verde.

3.8.2 *Depurar*

A pesar de que Java no es tan difícil de depurar como otros lenguajes de programación, también es posible que surjan complejos problemas de ejecución. Eclipse da apoyo completo a la tarea de depuración a través de su perspectiva "Debug" ("Window > Open Perspective > Debug" o seleccionando el icono del "bicho" en el margen).

Dentro de esta perspectiva de depuración, haciendo click en el margen izquierdo del editor de código aparecerá un menú contextual. Seleccionando "Add/Remove Breakpoint" añadirá o quitará un punto de ruptura.

"Toggle Breakpoint" cambiará el estado de activación del punto de ruptura. Los puntos de ruptura marcan líneas en que la ejecución del programa se detendrá de manera que sea posible comprobar el valor de las variables en ese instante, identificando así posibles errores.

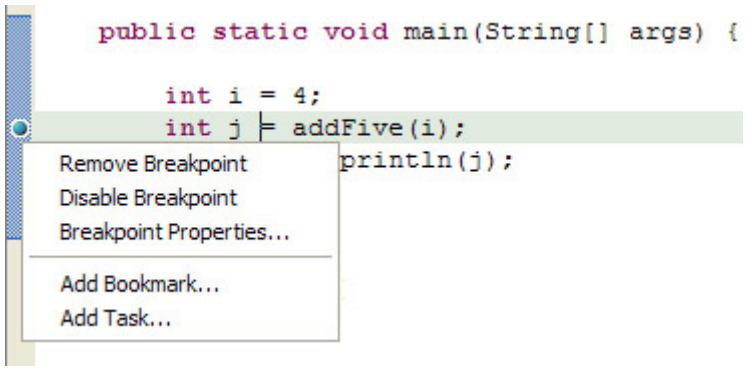

Figura 50: Identificar posibles errores

Haciendo click derecho en un punto de ruptura y seleccionando "Breakpoint Properties..." permitirá especificar opciones avanzadas del punto de ruptura.

"Hit Count" especifica que la ejecución del programa se detendrá cuando se pase por el punto de ruptura el número especificado de veces. Las condiciones de activación detendrán la ejecución cuando la condición sea cierta o bien cuando el valor de la condición cambie.

Especificar una variable como una condición de activación y seleccionar "suspend when value of condition changes" es una forma de "detener la ejecución en el punto de ruptura cuando dicha variable sea modificada".

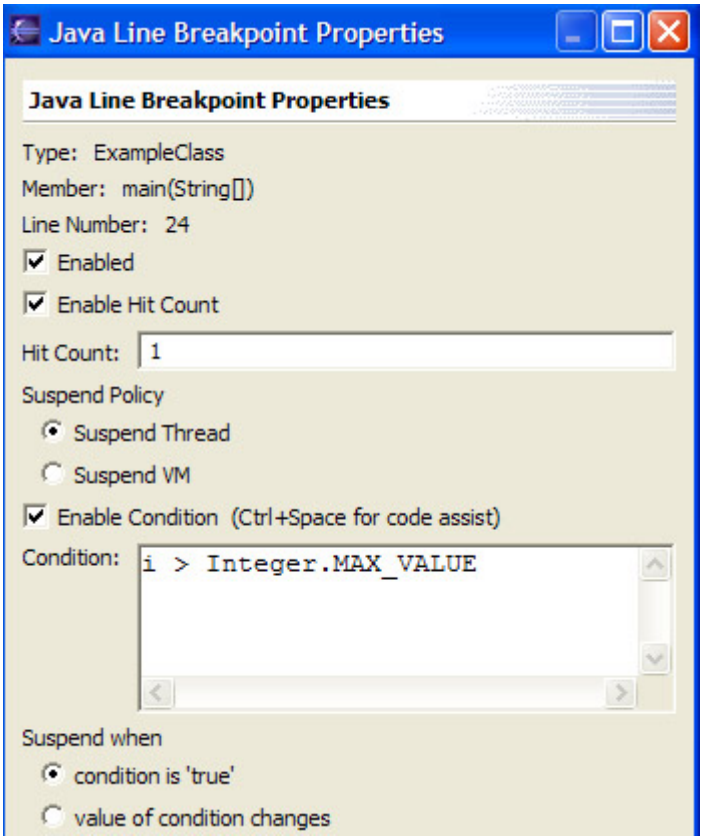

Figura51 :Ruptura

Las excepciones son uno de los síntomas más evidentes de errores de ejecución.

Los "Java Exception Breakpoints" detienen la ejecución cuando salta una excepción del tipo seleccionado. Estos puntos de ruptura se activan haciendo click en el icono "J!" de la vista de "Breakpoints" o desde el menú principal "Run".

La ejecución puede detenerse cuando la excepción sea capturada, no capturada o ambas. Añadir siempre los puntos de ruptura de excepciones Java de "ArrayIndexOutOfBoundsException" (lanzada cuando el índice de una matriz se sale de sus dimensiones)

"NullPointerException" (lanzada cuando se intenta acceder a una referencia que apunta a null) es una práctica de depuración recomendada.

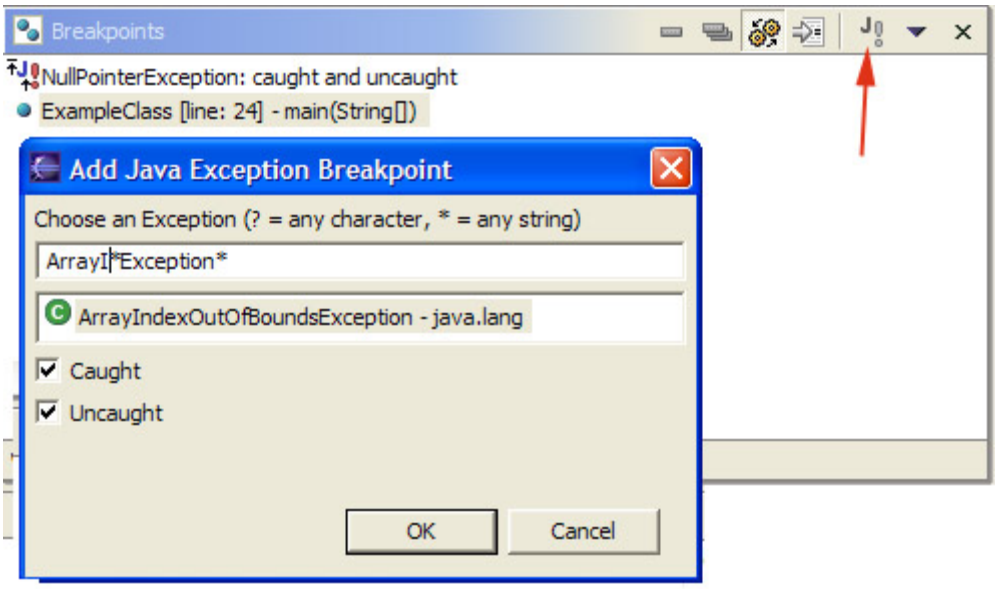

Figura52: Rupturas

Si se desea que el programa se detenga en los puntos de ruptura definidos deberá ser ejecutado en modo depuración ("Run > Debug..."). Tras detenerse en un punto de ruptura la ejecución del programa puede continuar de diversas maneras.

Dando un click derecho en el editor de código dentro de la perspectiva de depuración aparecerá un menú contextual con estas opciones.

"Run to line" reanuda la ejecución del programa hasta que se alcanza la línea en que está el cursor.

"Step into selection" continuará la ejecución dentro del método seleccionado siempre y cuando el código fuente del método esté disponible.

La ejecución también puede reanudarse mediante un click derecho en la ventana de "Debug" y seleccionando las opciones adecuadas, o directamente pulsando los iconos de dicha ventana. "Step over" parará en la línea siguiente a la invocación de un método.

"Resume" reanudará la ejecución normal del programa y sólo se interrumpirá en los puntos de ruptura si sus condiciones de activación se satisfacen.

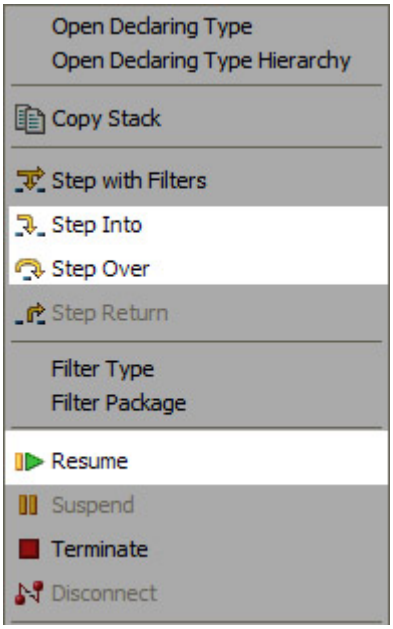

Figura53: Editor de Código

La vista "Variables" proporciona información verdaderamente útil ya que muestra los valores que tienen actualmente las variables cuando la ejecución se detiene en un punto de ruptura.

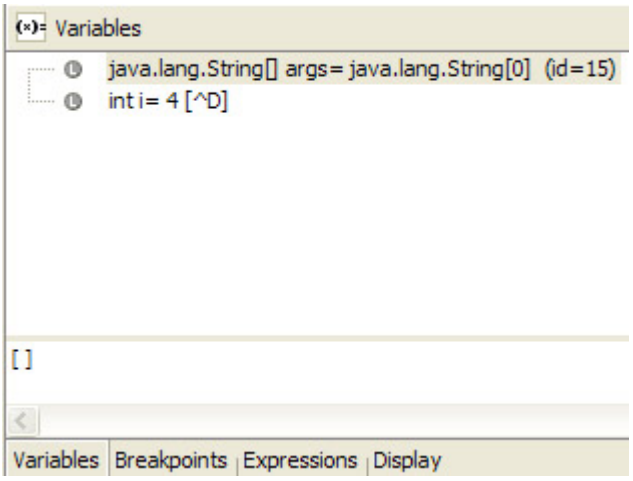

Figura 54: Variables

La vista de "Debug" también es útil para observar diferentes procesos que están siendo ejecutados simultáneamente.

Cuando el proceso de depuración ha terminado, los procesos mostrados en la ventana de depuración se muestran como "Finished" (pueden acabarse manualmente con "Click derecho > Terminate"). La información de ejecuciones previas puede elminarse realizando click derecho sobre ella y seleccionando "Terminate and Remove" o "Terminate All" más "Remove All Terminated".

3.8.3 *Gestión de Cambios*

Eclipse proporciona un potente sistema de gestión de cambios y de control de versiones. Haciendo click derecho en un archivo Java dentro del Package Explorer y seleccionando "Replace With > Local History" permitirá reemplazar la versión actual por una versión previamente guardada. La hora y fecha de modificación se muestran junto con dos ventanas que destacan las diferencias existentes entre ambas versiones.

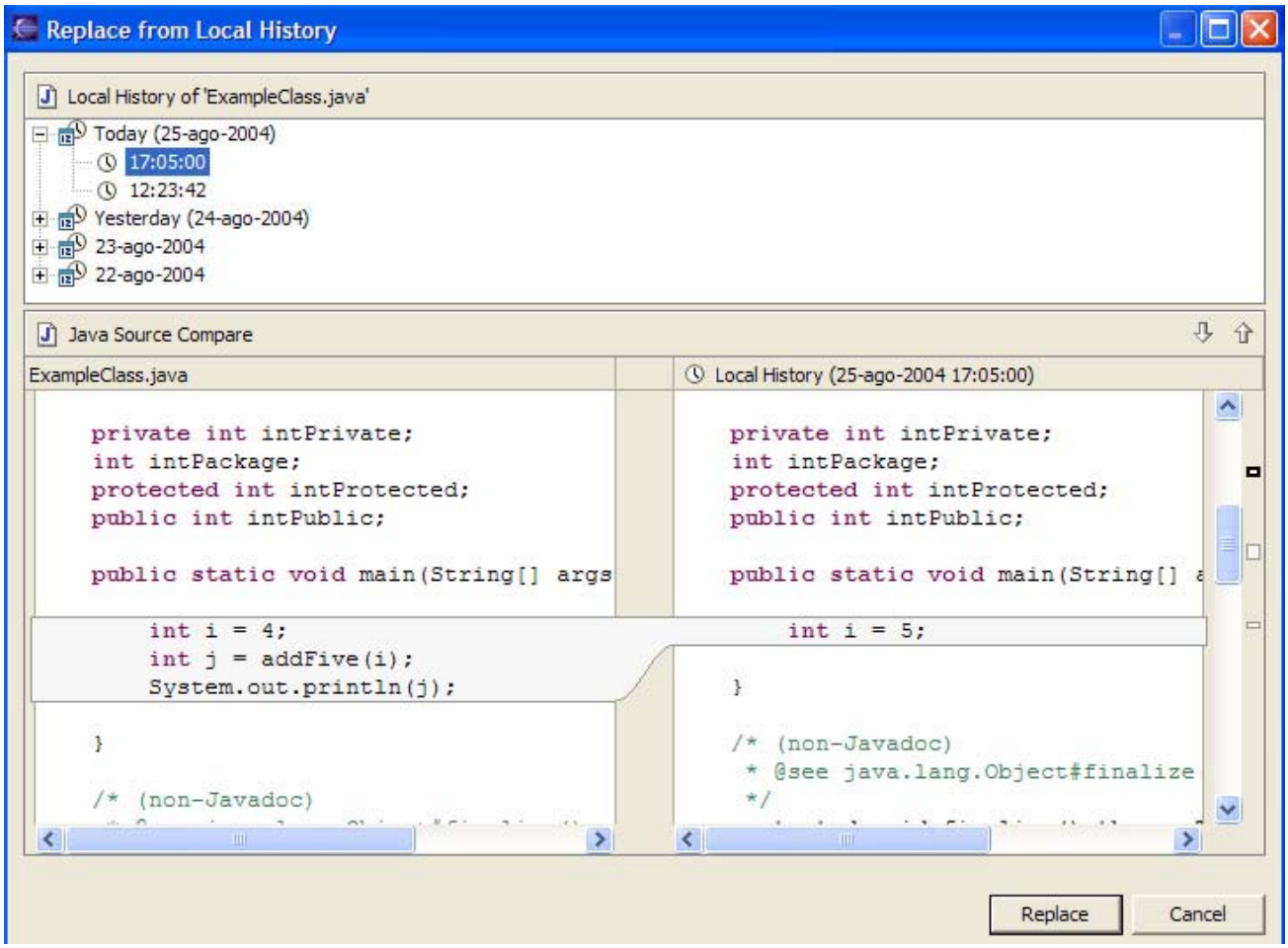

Figura55: Resumen de Modificaciones

Seleccionando "Window > Preferences > Workbench > Local History" permitirá seleccionar cuantas versiones, megas y días almacenar. Así pues, se puede obtener un buen equilibrio personalizado entre seguridad y eficiencia.

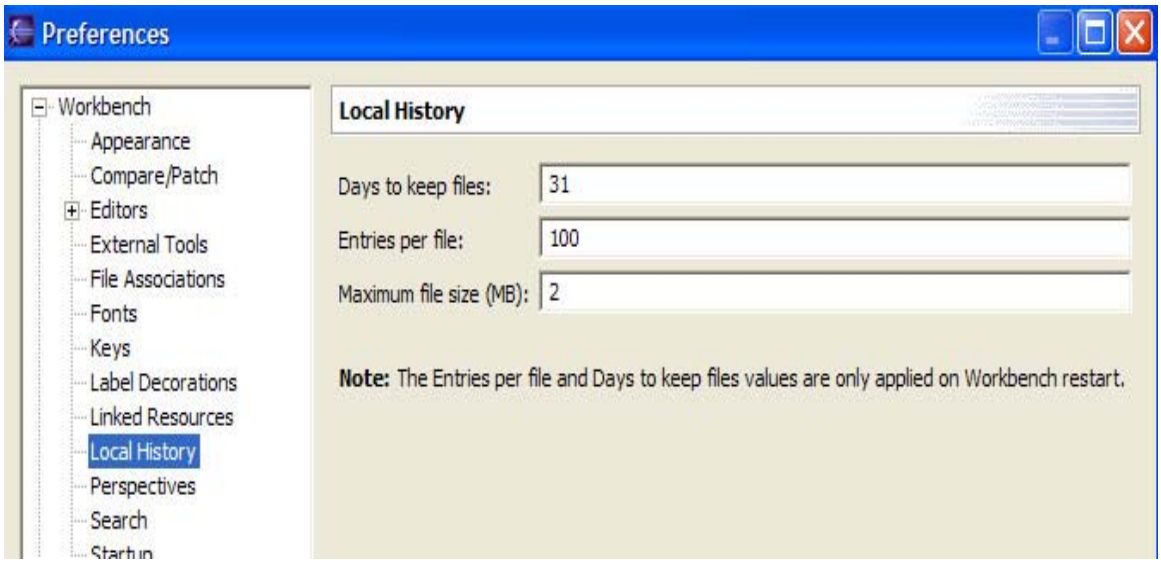

Figura56: Preferencias

3.9 OPTIMIZACION DE LA HERRAMIENTA ECLIPSE

3.9.1 DESCARGA E INSTALACIÓN DE PLUGINS

El Framework Eclipse es un entorno altamente extensible a través de plugins. Los mismos que van aportando nuevas prestaciones al entorno, volviéndolo más efectivo a la hora de crear aplicaciones.

3.9.2 VISUAL EDITOR

En este caso elegimos hablar acerca del "Visual Editor" de Eclipse, un conjunto de herramientas que permite diseñar "GUIs" Interfaces Gráficas de Usuario utilizando componentes Swing o SWT, tales como botones, cuadros de texto, etiquetas, etc; esto se puede realizar directamente tan solo arrastrando los objetos hacia un formulario para no tener que hacerlo únicamente mediante código.

El propósito de Visual Editor es promover el uso de la plataforma Eclipse añadiendo capacidades y servicios; este al igual que Eclipse pertenece a la comunidad "Open Source".

Para que Visual Editor funcione debemos instalar:

- Emf runtime (Eclipse Modeling Framework)
- GEF runtime (Eclipse Graphical Editing Framework)
- VE runtime (Visual Editor)

Si se descargan cada uno de estos directamente de la página de Eclipse se debe tomar en cuenta el obtener una versión que funcione correctamente con la versión de Eclipse instalada.

En nuestro caso descargamos las versiones:

- Emf runtime 2.3.0
- GEF runtime 3.2.1
- VE runtime 1.2.0

Después de esto lo único que debemos hacer es descomprimir estos archivos en la carpeta donde hayamos instalado Eclipse.

Callipso

En la versión de Eclipse que estamos utilizando 3.2.1 viene instalada una herramienta que nos facilita mucho la actualización de los plugins, esta se llama Callipso y nos permite a través del menú de Eclipse encontrar estas actualizaciones; es importante consultar que versión de las actualizaciones se ajusta a la versión que estamos utilizando, el proceso de descarga de actualizaciones utilizando Callipso se realiza de la siguiente manera:

Seleccionamos en el Menú Help→Software Updates→Find and Install.

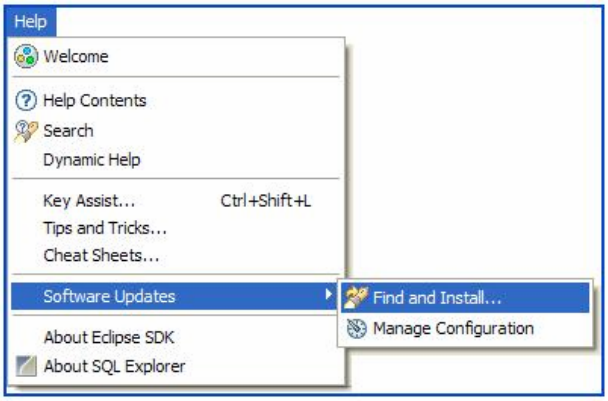

Figura57: Ruta de Acceso para descarga de actualizaciones

Luego de esto se nos desplegará una pantalla y seleccionamos esta opción:

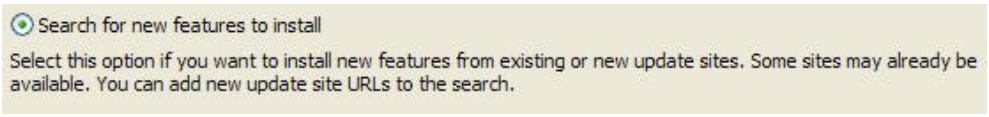

Figura 58: Opción para descarga de actualizaciones

Para descargar los plugins lo haremos seleccionando de entre todos:

Figura59: Sitio de donde se puede descargar las actualizaciones

Inmediatamente nos mostrará esta pantalla, en la cual se está estableciendo la conexión con “Callisto Discovery Site”.

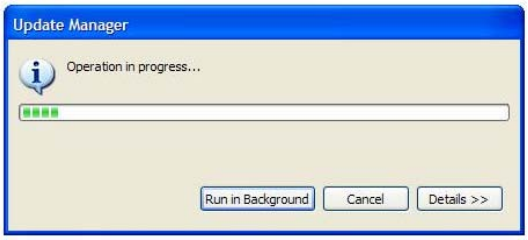

Figura 60: Pantalla que muestra el progreso de la conexión al sitio

Después al elegir el sitio de donde vamos a descargar las actualizaciones nuevamente seleccionamos “Callisto Discovery Site”.

Luego se mostrará una pantalla donde podremos elegir las actualizaciones necesarias:

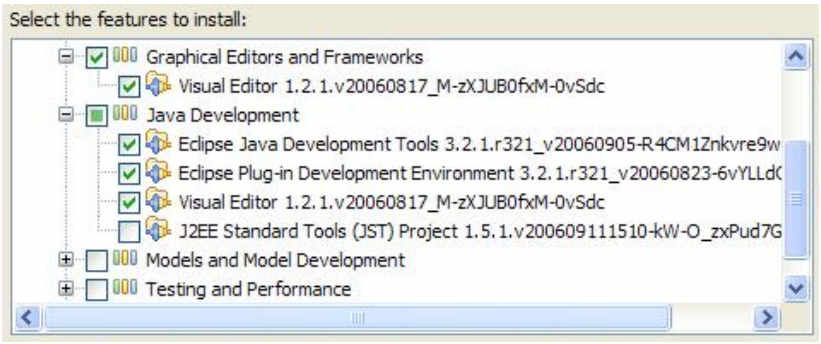

Figura 61: Pantalla con el listado de actualizaciones disponibles

Damos click en siguiente y ahora tendremos que aceptar los términos legales:

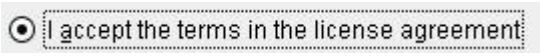

Figura 62: Opción para aceptar los términos legales

Después de esto las actualizaciones seleccionadas empiezan ya a descargarse.

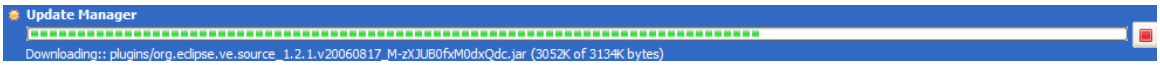

Figura63: Barra de progreso de la descarga de actualizaciones

Una vez descargados los nuevos componentes, solo queda instalarlos:

Y por ultimo nos indica que debemos reiniciar la aplicación para que los cambios sean realizados.

USO DEL VISUAL EDITOR

Para este ejemplo lo primero que haremos es crear dentro de Eclipse un “Java Project” con el nombre de “Prueba VE”, de igual manera que se realizó anteriormente en este capítulo.

El siguiente paso será crear dentro del Proyecto que creamos un elemento del tipo “Visual Class”, esto lo podemos encontrar en **File → New → Visual Class**.

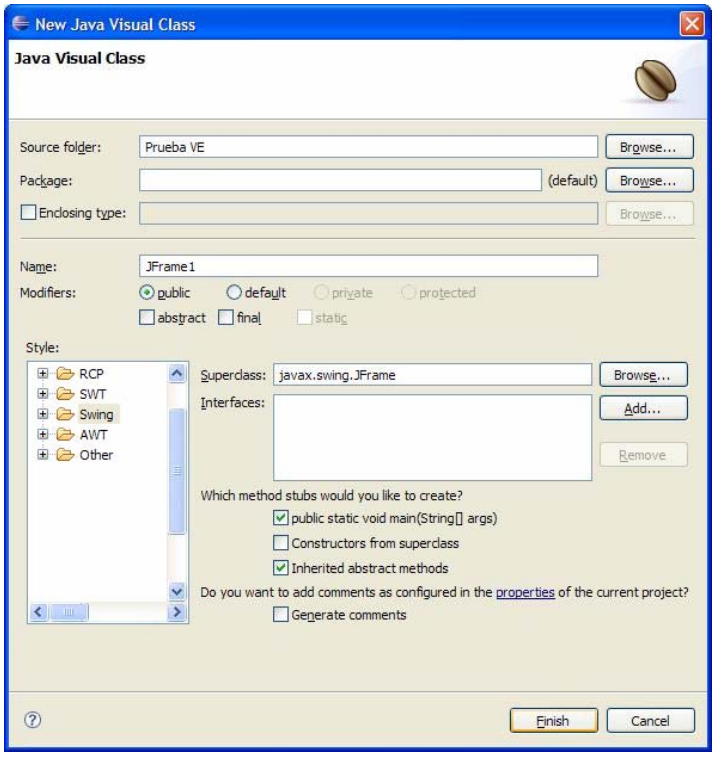

Figura64: Pantalla de creación de un elemento “Visual Class”

Ya una vez dentro lo que hacemos es escribir el nombre del elemento y además seleccionar el estilo que llevará nuestra clase, para el ejemplo seleccionaremos el estilo JFrame dentro del API Swing de Java, aunque también podemos elegir otros como Applet e incluso elementos del API SWT de Java.

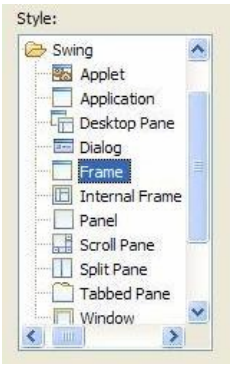

Figura65: Lista de selección del estilo del elemento “Visual Class”

Luego de esto damos click en Finish y se crea el elemento, entonces podremos apreciar en pantalla las diferentes partes que conforman el Visual Editor de Eclipse, como son:

- En la Figura 66 se muestra donde podemos ver la Barra de Herramientas de Visual Editor, que tiene controles de selección, otros para hacer y deshacer los cambios, además el botón de Pausa, etc.

Figura 66: Barra de Herramientas del Visual Editor

- Tenemos también la Figura 67 donde podemos ver primero La Paleta de Controles de donde se puede arrastrar componentes y soltarlos en la parte de la interfaz donde lo deseemos (Drag & Drop).

- La Vista de Diseño, aquí se muestra la Interfaz que estamos creando en el momento y donde colocaremos los controles que deseemos para la aplicación que estamos creando.

- La Vista de Código, en este lugar se irá colocando automáticamente el código de la ventana y de los componentes que vamos incluyendo; y además de esto podemos escribir todo el código extra que necesitemos para que nuestra aplicación funcione de la manera que deseamos.

- La Vista de Propiedad es donde después de dar click sobre el control al que queremos modificar las propiedades y después dependiendo del tipo de control se pueden modificar ciertas características como: tamaño, color de fondo, tipo de letra, tamaño de letra, etc.

- Algo importante a destacar es que cuando el Botón de Pausa de la Barra de Herramientas suspende la sincronización entre la Vista de Código y la Vista de Diseño, en este momento en lugar de Pausa se activa el botón de Reload y tan solo se puede hacer cambios en el Código, y en el momento que se considere necesario se da un click para que ambos Código y Diseño vuelvan a estar sincronizados.

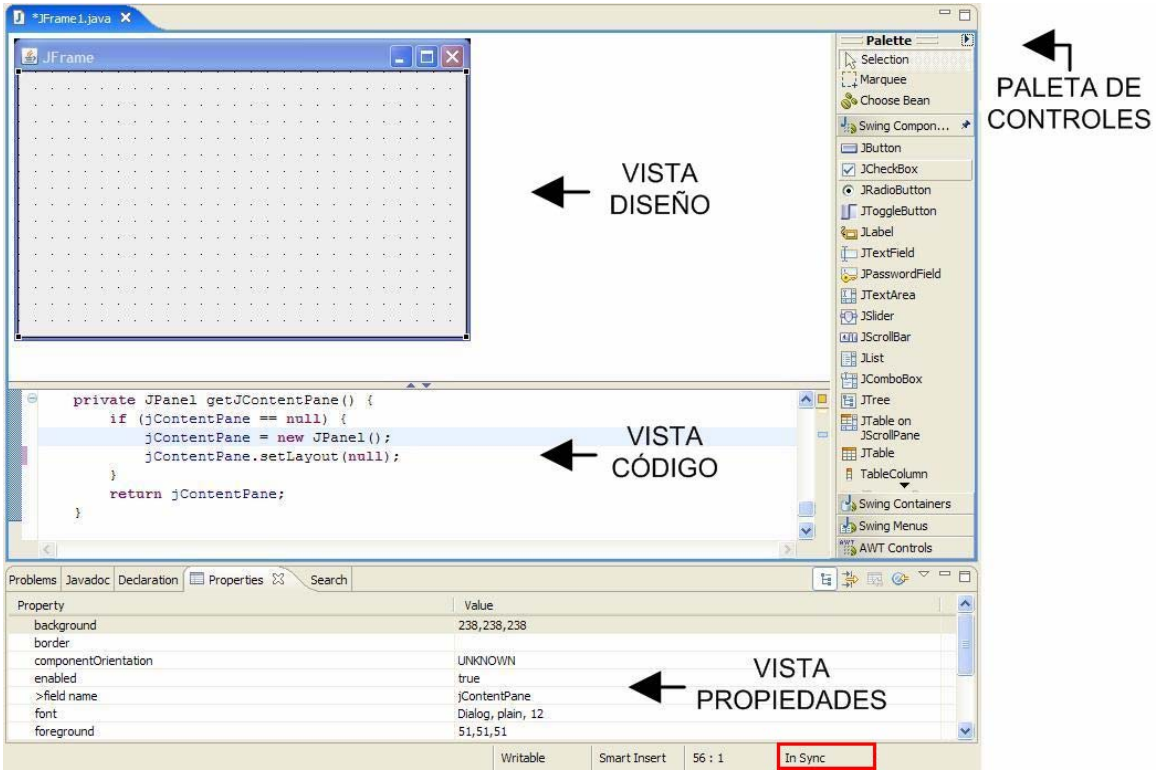

Figura 67: Elementos de la pantalla principal del Visual Editor de Eclipse

Para el ejemplo de cómo se maneja el Visual Editor vamos a realizar una calculadora, así que aparte de la clase Visual que creamos anteriormente, utilizaremos la clase Calculadora que se detalla a continuación:

```
//public class Calculadora {
//
//double numero1;
//double numero2;
//
//public Calculadora (double num1, double num2){
//		this.numero1 = num1;
//		this.numero2 = num2;
//}
//
//double sumaCalculadora(){
//		double resultado = 0;
//		resultado = this.numero1 + this.numero2;
//		return resultado;
//}
//
//double restaCalculadora(){
//		double resultado = 0;
//		resultado = this.numero1 - this.numero2;
//		return resultado;
//}
//
//double multiplicaCalculadora(){
//		double resultado = 0;
//		resultado = this.numero1 * this.numero2;
//		return resultado;
//}
//
//double divideCalculadora(){
//		double resultado = 0;
//		resultado = this.numero1 / this.numero2;
//		return resultado;
```

//}
//}

En la Figura 68 se muestra que para configurar algunas de las propiedades como set Layout o títulos de las ventanas, columnas de una tabla o una etiqueta, se lo puede realizar tan solo dando click derecho sobre el control y eligiendo la opción deseada.

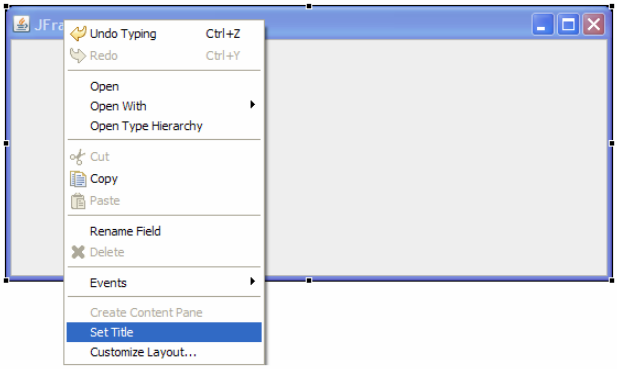

Figura 68: Ejemplo de modificación de propiedades de un control

Dentro de la paleta de controles podemos seleccionar lo que necesitemos para nuestra interfaz como pueden ser Botones, Labels, Text Fields, Tablas, Combo Box, Check Box, etc.

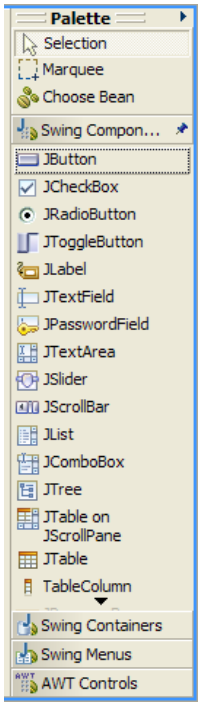

Figura69: Paleta de elección de controles de Visual Editor

Para nuestro ejemplo necesitaremos 3 Labels o Etiquetas, 3 Text Fields y 4 Botones puesto que el resultado al cual queremos llegar es el que se muestra en la Figura 70:

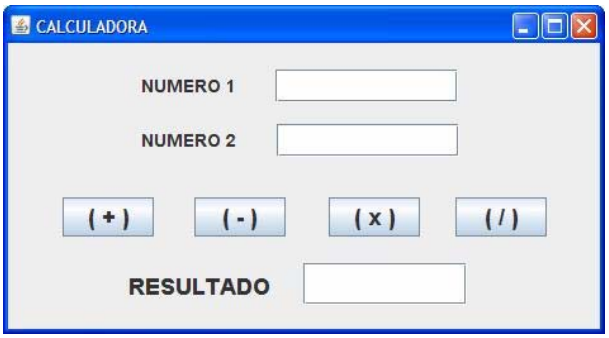

Figura70: Modelo de Interfaz del ejemplo de Visual Editor

Luego de colocar el control que deseemos sobre la interfaz nos aparecerá un cuadro como el de la Figura 71 en donde escribiremos el nombre con el cual identificaremos al control.

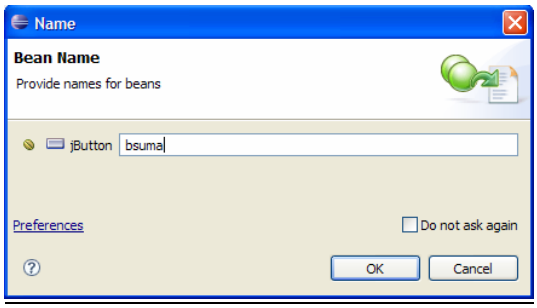

Figura 71: Creación de un control en el Visual Editor

Una vez colocados los controles que necesitamos para el ejemplo, empezaremos a trabajar con los eventos; para asignar eventos a un control damos click derecho sobre el mismo y elegimos alguno de los que se listas ahí y sino elegimos Add Events tal como se muestra en la Figura 72.

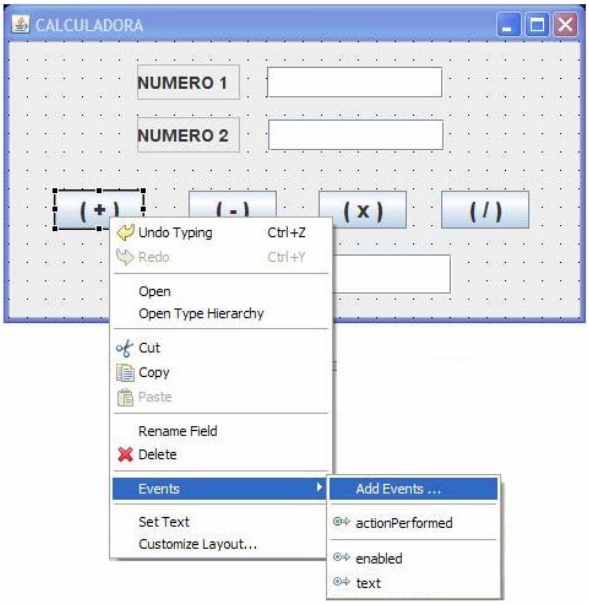

Figura72: Añadir eventos a un control

Dentro de Add Events se lista una serie de eventos que por ejemplo pueden dispararse en el caso de dar un click o al presionar una tecla, o en el momento en el que se enfoca alguno de los controles; aquí elegimos el evento que queremos programar como se

muestra en la Figura 73, entonces automáticamente se genera el código de declaración del evento y podemos empezar a escribir código que permita que el control funcione de la manera esperada.

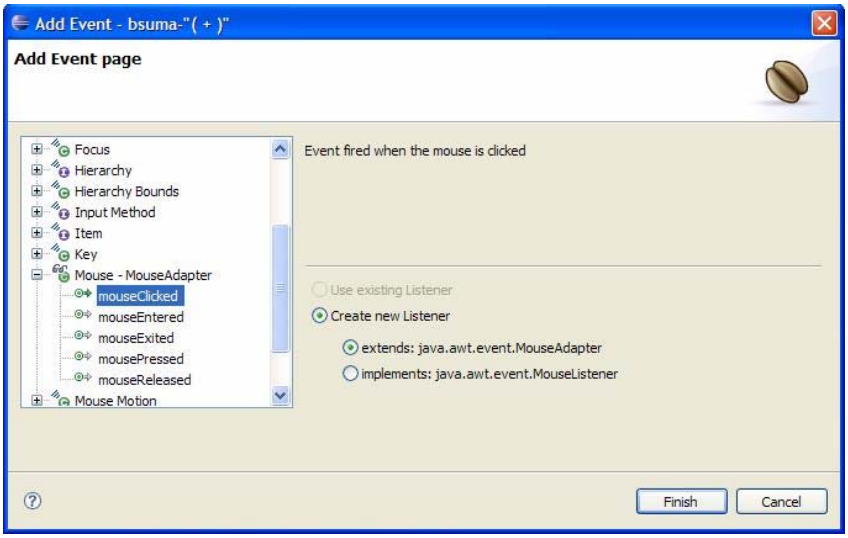

Figura73: Selección de tipo de evento en Visual Editor

El código que se genera de manera automática cuando elegimos un evento es similar en todos ellos, por ejemplo al añadir el evento suma al botón suma lo que obtuvimos en un inicio fue lo que se muestra a continuación, después como se muestra en la Figura 74, tuvimos que escribir algunas líneas más para que se sumen los números dentro de la clase Calculadora, y que el resultado se muestre en el Text Field de resultado:

```
//bsuma.addMouseListener(new java.awt.event.MouseAdapter() {
//		public void mouseClicked(java.awt.event.MouseEvent e) {
//			System.out.println("mouseClicked()"); // TODO Auto-generated Event stub mouseClicked()
//		}
//});
```

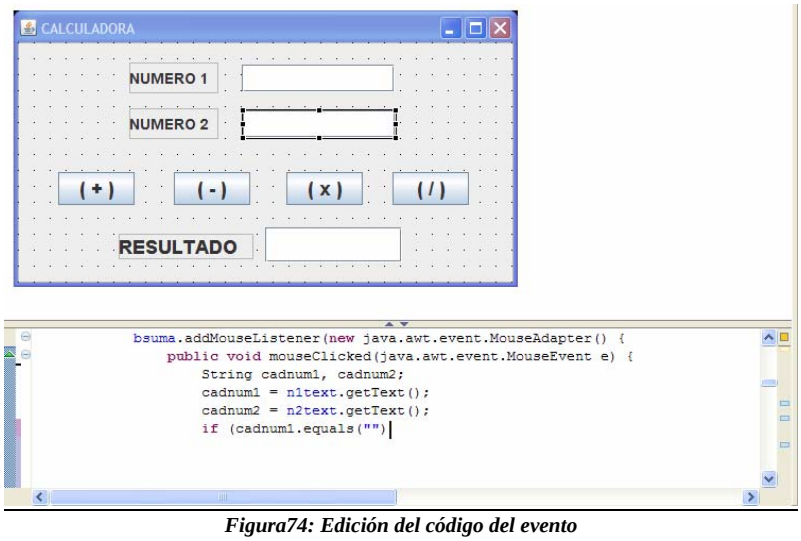

Figura74: Edición del código del evento

A continuación se muestra el código fuente resultado del trabajo realizado en el Framework Eclipse utilizando Visual Editor:

```
//import javax.swing.SwingUtilities;
//import javax.swing.JOptionPane;
//import javax.swing.JPanel;
//import javax.swing.JFrame;
//import javax.swing.JLabel;
//import java.awt.Rectangle;
//import javax.swing.JTextField;
//import java.awt.Font;
//import javax.swing.JButton;
//import java.text.DecimalFormat;
//
//public class Frame1 extends JFrame {
//
//	private static final long serialVersionUID = 1L;
//	private JPanel jContentPane = null;
//	private JLabel n1label = null;
//	private JLabel n2label = null;
//	private JTextField n1text = null;
//	private JTextField n2text = null;
//	private JLabel reslabel = null;
//	private JTextField restext = null;
//	private JButton bsuma = null;
//	private JButton bresta = null;
//	private JButton bmulti = null;
//	private JButton bdivi = null;
//
//	private JTextField getN1text() {
//		if (n1text == null) {
//			n1text = new JTextField();
//			n1text.setBounds(new Rectangle(211, 21, 144, 26));
//			n1text.setFont(new Font("Dialog", Font.BOLD, 14));
//		}
//		return n1text;
//	}
//
//	private JTextField getN2text() {
//		if (n2text == null) {
//			n2text = new JTextField();
//			n2text.setBounds(new Rectangle(212, 64, 144, 26));
//			n2text.setFont(new Font("Dialog", Font.BOLD, 14));
//		}
//		return n2text;
//	}
//
//	private JTextField getRestext() {
//		if (restext == null) {
//			restext = new JTextField();
//			restext.setBounds(new Rectangle(233, 174, 129, 33));
//			restext.setFont(new Font("Dialog", Font.BOLD, 18));
//		}
//		return restext;
//	}
//
//	private JButton getBsuma() {
//		if (bsuma == null) {
//			bsuma = new JButton();
//			bsuma.setBounds(new Rectangle(38, 122, 72, 31));
//			bsuma.setFont(new Font("Dialog", Font.BOLD, 18));
//			bsuma.setText("( + )");
//			bsuma.addMouseListener(new java.awt.event.MouseAdapter() {
//				public void mouseClicked(java.awt.event.MouseEvent e) {
//					System.out.println("mouseClicked()");
//					String cadnum1, cadnum2;
```

```
//                                          cadnum1 = n1text.getText();
//                                          cadnum2 = n2text.getText();
//                                          if (cadnum1.equals("")||cadnum2.equals(""))
//                                          {
//                                                  JOptionPane.showMessageDialog(null, "DEBE
                                                    INGRESAR TODOS LOS DATOS", "ATENCION",
                                                    JOptionPane.PLAIN_MESSAGE);
//                                          }
//                                          else
//                                          {
//                                                  double numero1 = 0, numero2 = 0;
//                                                  double resultado = 0;
//                                                  numero1 = Double.parseDouble(n1text.getText());
//                                                  numero2 = Double.parseDouble(n2text.getText());
//                                                  Calculadora obj1 = new Calculadora(numero1, numero2);
//                                                  resultado = obj1.sumaCalculadora();
//                                                  DecimalFormat Digitos = new DecimalFormat("0.00");
//                                                  restext.setText(Digitos.format(resultado));
//                                          }
//                                  }
//                          });
//                  }
//                  return bsuma;
//          }
//
//          private JButton getBresta() {
//                  if (bresta == null) {
//                          bresta = new JButton();
//                          bresta.setBounds(new Rectangle(147, 122, 72, 31));
//                          bresta.setText("( - )");
//                          bresta.setFont(new Font("Dialog", Font.BOLD, 18));
//                          bresta.addMouseListener(new java.awt.event.MouseAdapter() {
//                                  public void mouseClicked(java.awt.event.MouseEvent e) {
//                                          System.out.println("mouseClicked()");
//                                          String cadnum1, cadnum2;
//                                          cadnum1 = n1text.getText();
//                                          cadnum2 = n2text.getText();
//                                          if (cadnum1.equals("")||cadnum2.equals(""))
//                                          {
//                                                  JOptionPane.showMessageDialog(null, "DEBE
                                                    INGRESAR TODOS LOS DATOS", "ATENCION",
                                                    JOptionPane.PLAIN_MESSAGE);
//                                          }
//                                          else
//                                          {
//                                                  double numero1 = 0, numero2 = 0;
//                                                  double resultado = 0;
//                                                  numero1 = Double.parseDouble(n1text.getText());
//                                                  numero2 = Double.parseDouble(n2text.getText());
//                                                  Calculadora obj1 = new Calculadora(numero1, numero2);
//                                                  resultado = obj1.restaCalculadora();
//                                                  DecimalFormat Digitos = new DecimalFormat("0.00");
//                                                  restext.setText(Digitos.format(resultado));
//                                          }
//                                  }
//                          });
//                  }
//                  return bresta;
//          }
//
//          private JButton getBmulti() {
//                  if (bmulti == null) {
//                          bmulti = new JButton();
//                          bmulti.setBounds(new Rectangle(253, 122, 72, 31));
//                          bmulti.setText("( x )");
//                          bmulti.setFont(new Font("Dialog", Font.BOLD, 18));
//                          bmulti.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
//                                      public void mouseClicked(java.awt.event.MouseEvent e) {
//                                              System.out.println("mouseClicked()");
//                                              String cadnum1, cadnum2;
//                                              cadnum1 = n1text.getText();
//                                              cadnum2 = n2text.getText();
//                                              if (cadnum1.equals("")||cadnum2.equals(""))
//                                              {
//                                                      JOptionPane.showMessageDialog(null, "DEBE
                                                        INGRESAR TODOS LOS DATOS", "ATENCION",
                                                        JOptionPane.PLAIN_MESSAGE);
//                                              }
//                                              else
//                                              {
//                                                      double numero1 = 0, numero2 = 0;
//                                                      double resultado = 0;
//                                                      numero1 = Double.parseDouble(n1text.getText());
//                                                      numero2 = Double.parseDouble(n2text.getText());
//                                                      Calculadora obj1 = new Calculadora(numero1, numero2);
//                                                      resultado = obj1.multiplicaCalculadora();
//                                                      DecimalFormat Digitos = new DecimalFormat("0.00");
//                                                      restext.setText(Digitos.format(resultado));
//                                              }
//                                      }
//                              });
//                      }
//                      return bmulti;
//              }
//
//              private JButton getBdivi() {
//                      if (bdivi == null) {
//                              bdivi = new JButton();
//                              bdivi.setBounds(new Rectangle(353, 122, 72, 31));
//                              bdivi.setText("( / )");
//                              bdivi.setFont(new Font("Dialog", Font.BOLD, 18));
//                              bdivi.addMouseListener(new java.awt.event.MouseAdapter() {
//                                      public void mouseClicked(java.awt.event.MouseEvent e) {
//                                              System.out.println("mouseClicked()");
//                                              double numero1 = 0, numero2 = 0;
//                                              double resultado = 0;
//                                              numero1 = Double.parseDouble(n1text.getText());
//                                              numero2 = Double.parseDouble(n2text.getText());
//                                              String cadnum1, cadnum2;
//                                              cadnum1 = n1text.getText();
//                                              cadnum2 = n2text.getText();
//                                              if (cadnum1.equals("")||cadnum2.equals("")||numero2==0)
//                                              {
//                                                      JOptionPane.showMessageDialog(null, "DEBE
                                                        INGRESAR TODOS LOS DATOS Y 2DO NUMERO
                                                        DEBE SER DISTINTO DE CERO", "ATENCION",
                                                        JOptionPane.PLAIN_MESSAGE);
//                                              }
//                                              else
//                                              {
//                                                      Calculadora obj1 = new Calculadora(numero1, numero2);
//                                                      resultado = obj1.divideCalculadora();
//                                                      DecimalFormat Digitos = new DecimalFormat("0.00");
//                                                      restext.setText(Digitos.format(resultado));
//                                              }
//                                      }
//                              });
//                      }
//                      return bdivi;
//              }
//
//              public static void main(String[] args) {
//                      SwingUtilities.invokeLater(new Runnable() {
//                              public void run() {
```

```
//                                      Frame1 thisClass = new Frame1();
//                                      thisClass.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
//                                      thisClass.setVisible(true);
//                              }
//                      });
//      }
//
//      public Frame1() {
//              super();
//              initialize();
//      }
//
//      private void initialize() {
//              this.setContentPane(getJContentPane());
//              this.setTitle("CALCULADORA");
//              this.setBounds(new Rectangle(0, 0, 475, 260));
//      }
//
//      private JPanel getJContentPane() {
//              if (jContentPane == null) {
//                      reslabel = new JLabel();
//                      reslabel.setBounds(new Rectangle(95, 180, 127, 24));
//                      reslabel.setFont(new Font("Dialog", Font.BOLD, 18));
//                      reslabel.setText("RESULTADO");
//                      n2label = new JLabel();
//                      n2label.setBounds(new Rectangle(105, 62, 84, 29));
//                      n2label.setFont(new Font("Dialog", Font.BOLD, 14));
//                      n2label.setText("NUMERO 2");
//                      n1label = new JLabel();
//                      n1label.setBounds(new Rectangle(105, 19, 84, 29));
//                      n1label.setFont(new Font("Dialog", Font.BOLD, 14));
//                      n1label.setText("NUMERO 1");
//                      jContentPane = new JPanel();
//                      jContentPane.setLayout(null);
//                      jContentPane.add(n1label, null);
//                      jContentPane.add(n2label, null);
//                      jContentPane.add(getN1text(), null);
//                      jContentPane.add(getN2text(), null);
//                      jContentPane.add(reslabel, null);
//                      jContentPane.add(getRestext(), null);
//                      jContentPane.add(getBsuma(), null);
//                      jContentPane.add(getBresta(), null);
//                      jContentPane.add(getBmulti(), null);
//                      jContentPane.add(getBdivi(), null);
//              }
//              return jContentPane;
//      }
//}
```

- En la Figura 75 está la Vista Java Beans, donde se lista los controles que fueron utilizados para el ejemplo, junto con los eventos que se manejaron.

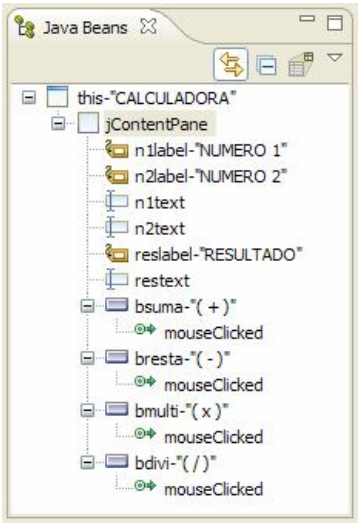

Figura 75: Vista Java Beans de Visual Editor

Y por último la Figura 76 muestra el resultado final de esta aplicación.

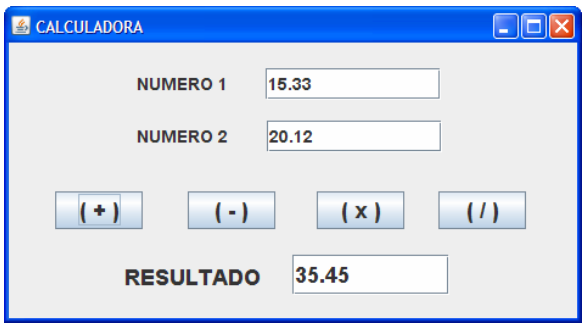

Figura76: Resultado Final de la Calculadora

Cabe destacar que este ejemplo pudo haber sido realizado sin necesidad de utilizar la clase calculadora, pero lo consideramos necesario para demostrar que aparte de trabajar con eventos, se puede seguir trabajando normalmente utilizando clases y objetos que son la base del Lenguaje Java.

3.9.3 BIRT (Business Intelligence and Reporting Tools)

Otro de los plugins de Eclipse que nos pareció interesante es el BIRT que sirve para crear reportes de nuestra BD, ya que la información que estos pueden proveer sería de mucha ayuda para una empresa.

Para poder utilizar BIRT se debe instalar lo siguiente:

- BIRT 2.1.1
- Eclipse 3.2.1
- GEF 3.2.1
- EMF 2.3.0
- JRE 1.6.0

Todo esto se puede encontrar para descarga en la página de Eclipse www.eclipse.org, para descargar el BIRT vamos a la página principal de este proyecto

www.eclipse.org/birt/ y damos click en el botón “Download BIRT”, esto nos enviará a la zona de descarga del mismo, en donde dice “Cose a Designer Download” damos click en “Framework”, en la página que sigue elegimos el sitio de donde deseamos descargar el BIRT, obteniendo el archivo:

birt-report-framework-2.1.2.zip

Después de esto tan solo necesitamos descomprimir el archivo en el directorio donde se encuentra Eclipse y listo, ejecutamos el Framework y podemos empezar a utilizarlo.

USO DE BIRT

Para este ejemplo lo primero que hicimos fue crear una BD en MySQL Server 5.0 llamada “birt” donde creamos una tabla de Clientes e ingresamos algunos registros en la misma, a continuación se muestra la estructura de la tabla y su contenido:

CLIENTES

CAMPO	TIPO	NULL	PK
cli_cedula	**varchar(10)**	**NO**	**PRI**
cli_nombre	varchar(80)	NO	
cli_direccion	varchar(80)	NO	
cli_telefono	varchar(12)	NO	
cli_mail	varchar(40)	NO	

Datos Existentes al momento del ejemplo:

CLIENTES

cli_cedula	cli_nombre	cli_direccion	cli_telefono	cli_mail
0301928438	JOSE FRANCISCO MONTERO FLOR	AV. DON BOSCO	2819625	carowpooh@hotmail.com
0104636824	CAROLINA ISABEL FLORES CORONEL	PUERTAS DEL SOL	2452060	jofranm18@hotmail.com

Abrimos Eclipse, y mientras nos encontramos en la perspectiva original del mismo creamos un nuevo proyecto en la ruta **File → New → New Project.**

Una vez aquí dentro crearemos un Proyecto para reportes tal y como se indica en la siguiente figura **Business Intelligence and Reporting Tools → Report Project**, aunque se podría trabajar creando los reportes dentro de un proyecto de Java; bueno una vez seleccionado esto damos click en **Next** y en el siguiente paso escribimos el nombre que tendrá nuestro proyecto que será REPORTE.

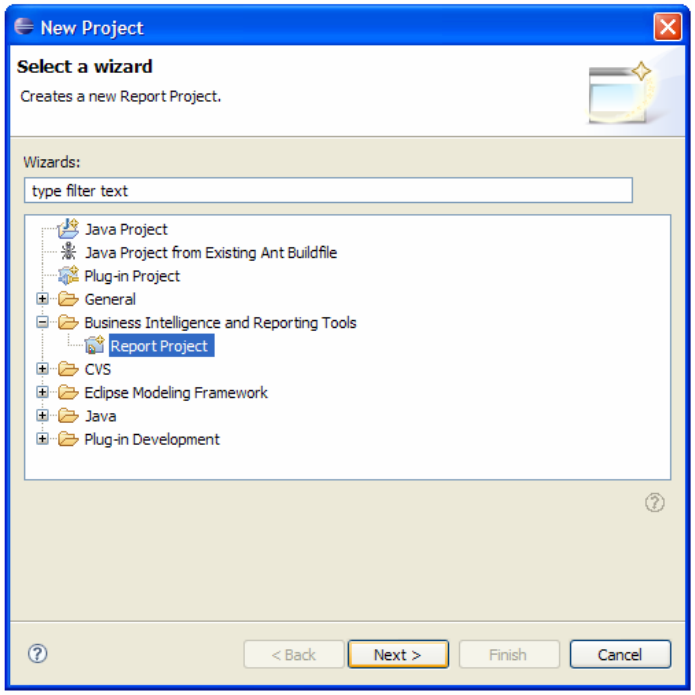

Figura 77: Creación de Nuevo Proyecto

Después de escrito el nombre damos click en **Finish** y nos saldrá una pantalla en donde nos pregunta si queremos cambiar a la perspectiva **Report Desing** que es donde crearemos nuestro reporte, elegimos si y se cambia automáticamente a la perspectiva; otra forma de realizar esto es mediante el botón “Open Perspective” que se encuentra en la parte superior derecha de la pantalla con lo que se desplegará la pantalla que se muestra a continuación que nos permitirá cambiar de perspectiva:

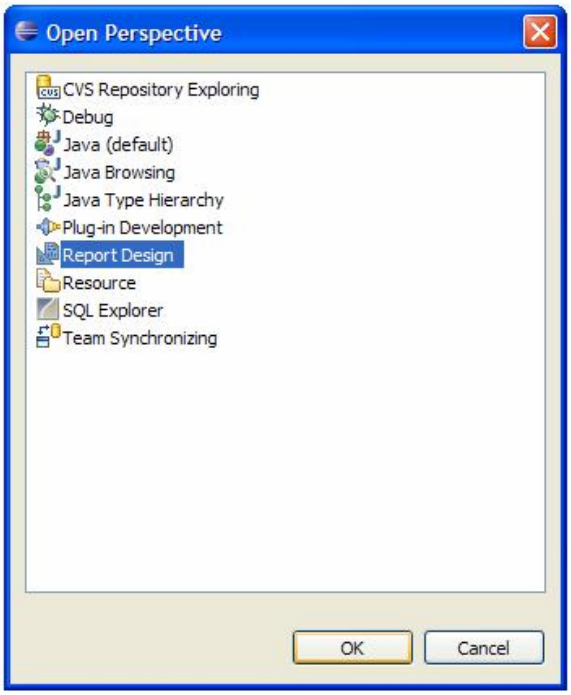

Figura 78: Open Perspective

Lo siguiente que haremos será crear una nueva hoja de reporte en la siguiente ruta de la barra de menú **File → New → Report,** aquí lo asociaremos al proyecto que creamos y escribimos su nombre “Clientes.rptdesign”.

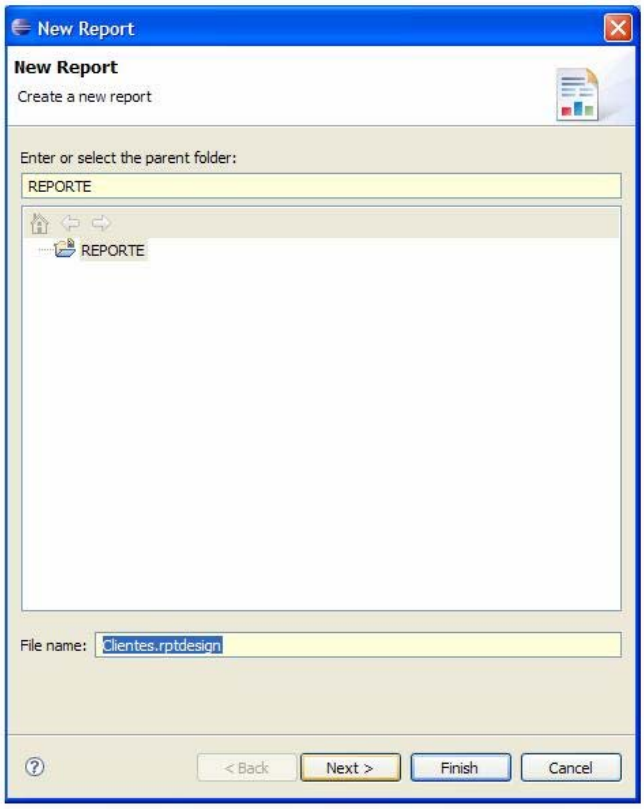

Figura79: Creación de Reporte

Al dar click en **Next** podremos elegir crear un reporte nuevo (Blank Report) o algún modelo predefinido, elegimos Blank Report y presionamos **Finish**.

A continuación se muestra la perspectiva Report Design con todos sus componentes:

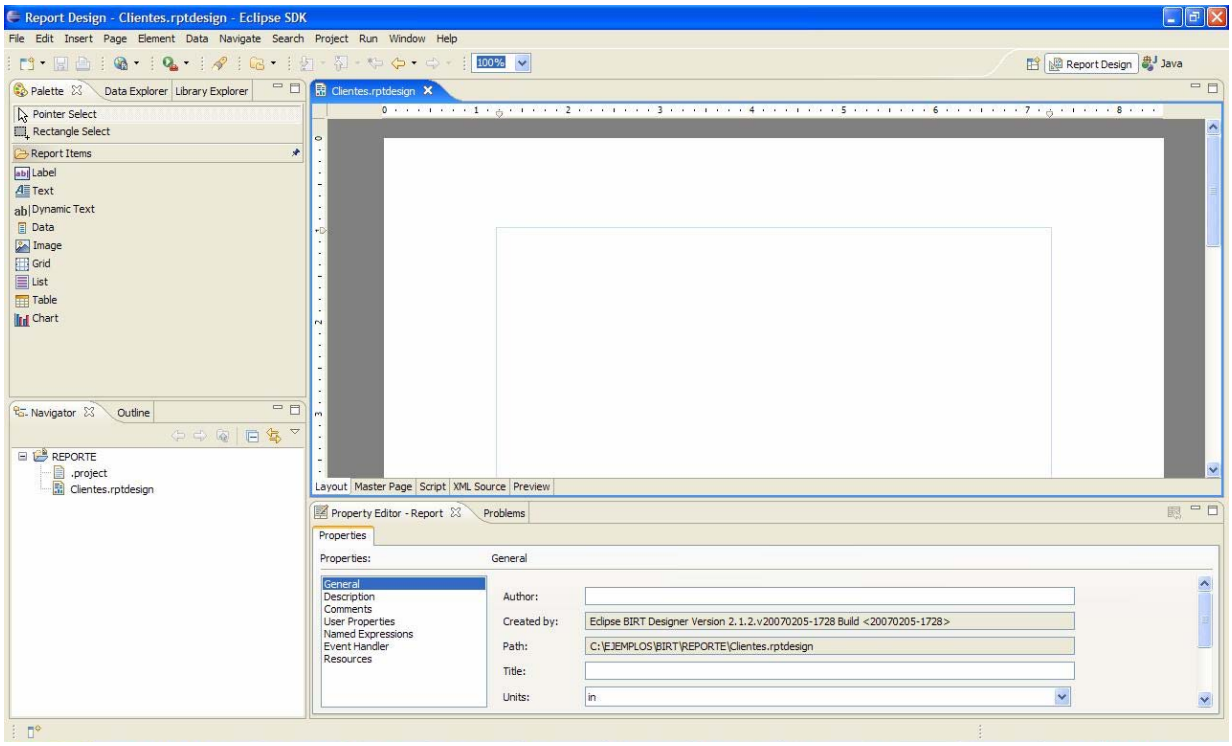

Figura80: Perspectiva Report Design

El componente más importante es la pantalla donde diseñaremos el reporte, en la parte baja de esta está una serie de botones que nos permitirán tener una vista previa del reporte “Preview”, ver el código XML del reporte “XML Source” y el principal ”Layout” para poder diseñar el reporte.

Debajo de la ventana de Diseño se encuentra la ventana de propiedades donde se podrán cambiar algunas características de los elementos que pongamos en la hoja de reporte.

En la parte superior izquierda se pueden encontrar dos pestañas también importantes, “Palette” de donde seleccionaremos y arrastraremos los componentes a la ventana de diseño de manera similar a como se trabajaba en Visual Editor; la otra pestaña es “Data Explorer” donde configuraremos las conexiones y generaremos consultas para utilizarlas en el reporte.

Ahora seleccionamos la pestaña “Data Explorer” y damos click derecho sobre “Data Sources” y elegimos New Data Source del menú contextual.

En la pantalla que se despliega elegimos de la lista “JDBC Data Source” para poder utilizar la base de datos de MySQL y damos click en **Next** con lo que se mostrará una pantalla como la siguiente donde configuraremos la conexión a la BD, aquí escribimos lo siguiente:

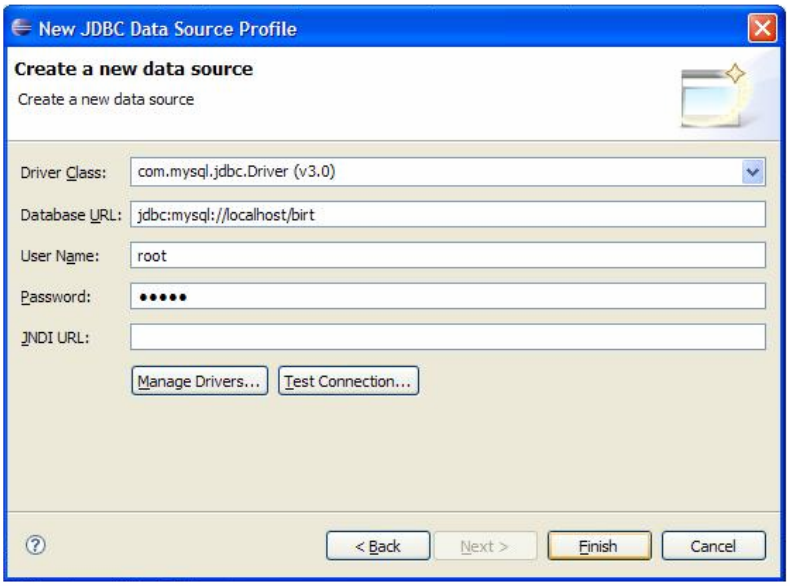

Figura 81: Configurar Conexión

Luego dentro de "Data Explorer" damos clic derecho ahora sobre "Data Sets" y elegimos New Data Set.

Ahora en la nueva pantalla en Data Set Name escribimos un nombre para el conjunto de datos, en este caso Clientes, en Data Source elegimos lo que acabamos de crear, por último en Data Set Type elegimos SQL Select Quero y damos click en siguiente.

Se nos desplegará una pantalla donde podremos escribir en base a la información almacenada en la tabla una consulta, para el ejemplo la consulta que utilizaremos es:

SELECT clientes.cli_cedula, clientes.cli_nombre, clientes.cli_direccion, clientes.cli_telefono, clientes.cli_mail
FROM clientes
ORDER BY 2

Luego a partir de los datos que obtuvimos podremos crear un reporte utilizando los componentes de la Paleta, al final la vista de Layout de nuestro reporte quedó de la siguiente manera, para esto se utilizó un elemento lista, una tabla, varios labels y además arrastramos de nuestro Data Set las columnas que queremos que se muestren.

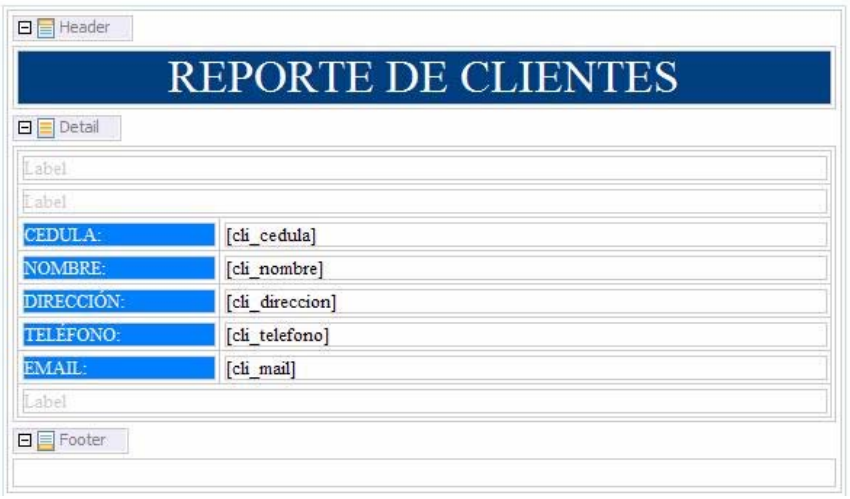

Figura 82: Layout del Reporte

Y por último al elegir Preview podremos ver el resultado del reporte que creamos y además si damos click derecho sobre la pantalla podremos enviarlo a impresora.

REPORTE DE CLIENTES

CEDULA:	0104636824
NOMBRE:	CAROLINA ISABEL FLORES CORONEL
DIRECCIÓN:	PUERTAS DEL SOL
TELÉFONO:	2452060
EMAIL:	carowpooh@hotmail.com

CEDULA:	0301928438
NOMBRE:	JOSE FRANCISCO MONTERO FLOR
DIRECCIÓN:	AV. DON BOSCO
TELÉFONO:	2819625
EMAIL:	jofranm18@hotmail.com

Mar 22, 2007 2:20 AM

Figura 83: Vista Preliminar del Reporte

3.10 CONCLUSION

Eclipse es un editor de código, de uso fácil, se compila de una forma rápida, el entorno es amigable. Esta herramienta de desarrollo es una pieza clave para la calidad y productividad de un proyecto.

Eclipse es totalmente extensible, es decir se pueden fabricar plug-ins (desarrollo a parte de la plataforma de eclipse que aporta una determinada funcionalidad) o utilizar los que ya están desarrollados.

Eclipse es una estructura (workbench) que puede soportar distintas herramientas de desarrollo y para cualquier lenguaje.

CAPITULO 4: APLICACIÓN PRÁCTICA

4.1 PROBLEMA

Un pequeño restaurante requiere un sistema de control de cuentas de gastos de sus clientes que deberá ser utilizado por el cajero del restaurante.
Cada cliente debe poder hacer un depósito inicial que le dará crédito para consumir los productos y servicios del restaurante.
La compra sólo se deberá efectuar si hay crédito suficiente en la cuenta del cliente para el débito en cuestión.
Las cuentas se manejarán de acuerdo a tres estados:

ABIERTA: Al momento de abrir una cuenta el valor mínimo en la acreditación es de $5. En este momento se puede disponer de los productos a su elección.

CERRADA: La cuenta se cambiará a este estado, en el momento en el que el mismo cliente así lo desee. Y en el caso de existir saldo se procederá al respectivo reembolso.

CRITICA: La cuenta cambiará de estado de abierto a crítico automáticamente, cuando después de una operación de débito el saldo sobrante se detecte inferior o igual a $1, en este caso existen dos opciones: al cliente se le puede reembolsar su saldo o el mismo puede volver a acreditar un nuevo valor siempre y cuando sea este mayor o igual a $5.

Todos los movimientos que los clientes realicen en sus cuentas, como abrir, cerrar, reabrir, debitar o acreditar, quedarán registrados.
Para una mayor comodidad de los clientes, se contará con un terminal de consulta; el mismo que les servirá para realizar sus propias consultas sobre su cuenta o sus datos personales.

Notas:

- Los datos de los clientes se podrán eliminar solo en el caso de que la cédula de identidad esté incorrecta, con esto se eliminarán automáticamente también las cuentas y los saldos para que puedan volver a ser ingresados correctamente.
- Los clientes podrán disponer de varias cuentas.

4.2 ANALISIS DE LA APLICACIÓN.

4.2.1 DIAGRAMA DE CASO DE USO

Un caso de uso representa una funcionalidad del sistema que utiliza un actor, siento actor un tipo de usuario que desempeña una función dentro del sistema. El diagrama de caso de uso se basa en las funcionalidades del sistema y en el actor que las utiliza.

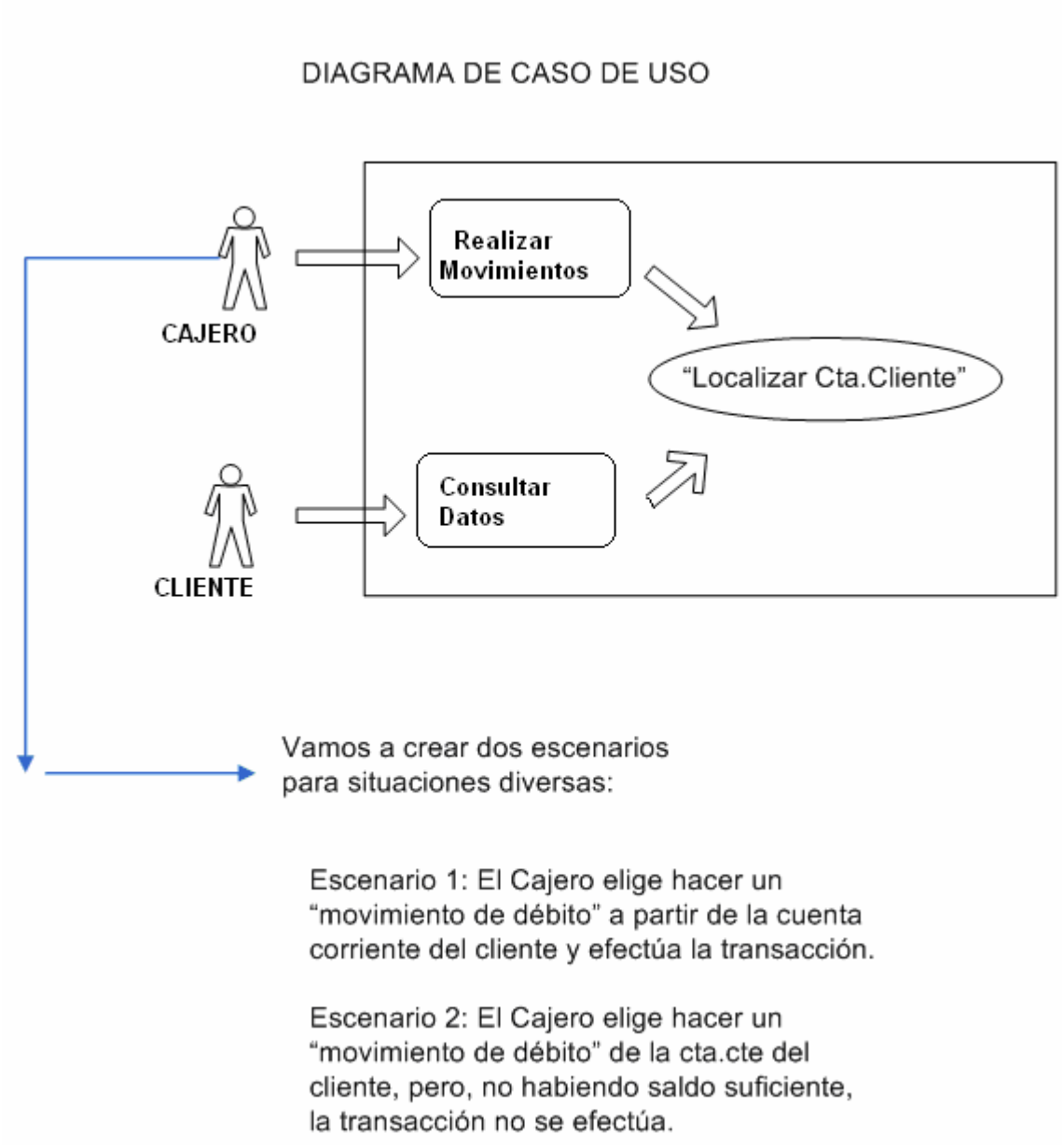

Figura 84: Casos de Uso

4.2.2. DIAGRAMA DE OBJETOS

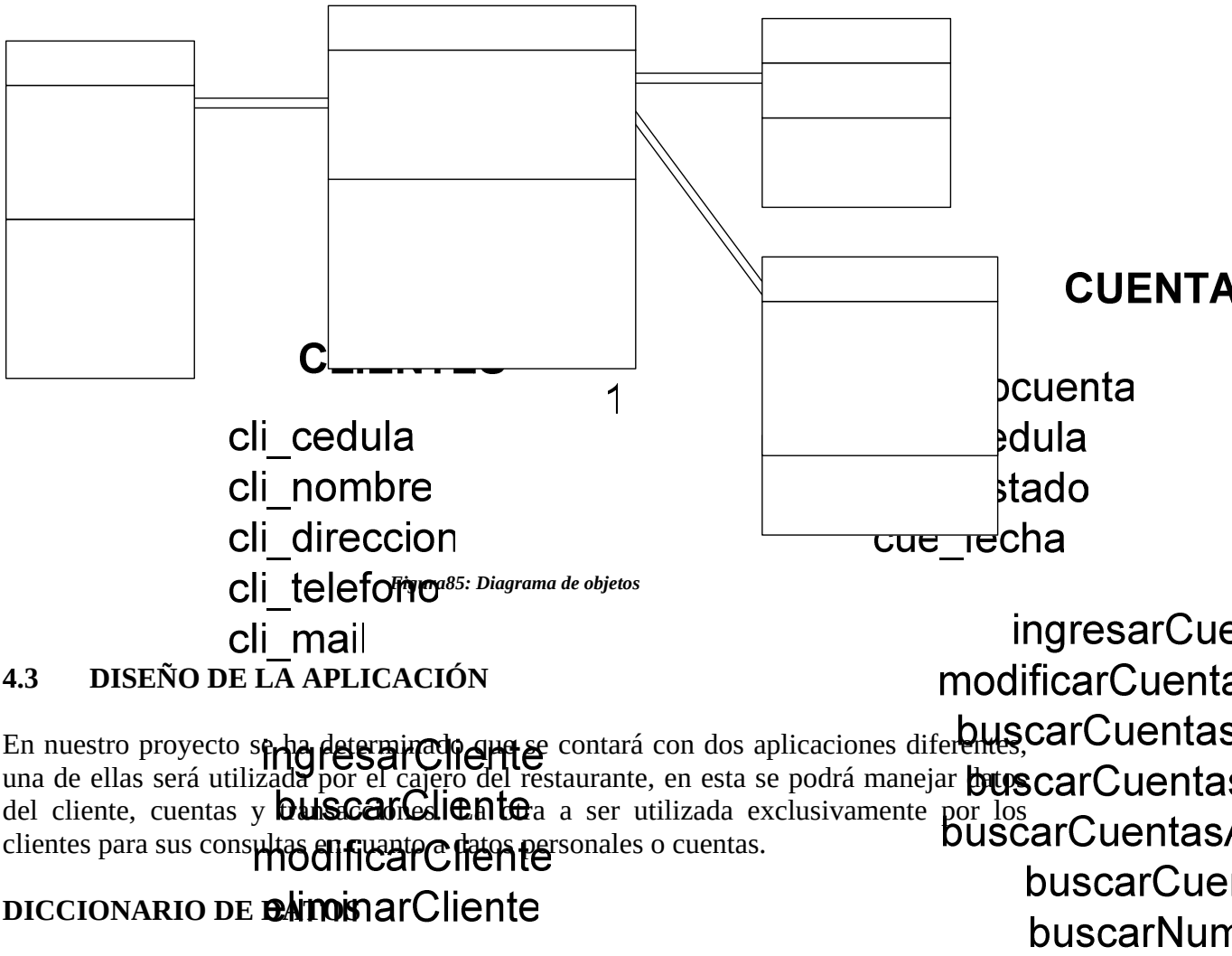

Figura85: Diagrama de objetos

4.3 DISEÑO DE LA APLICACIÓN

En nuestro proyecto se ha determinado que se contará con dos aplicaciones diferentes, una de ellas será utilizada por el cajero del restaurante, en esta se podrá manejar datos del cliente, cuentas y transacciones. La otra a ser utilizada exclusivamente por los clientes para sus consultas en cuanto a datos personales o cuentas.

DICCIONARIO DE DATOS

Tabla	Columna	Descripcion	Tipo	Tam	Llave Primaria	Llave Foranea
CLIENTES	cli_cedula	Cédula del Cliente	varchar	10	si	no
CLIENTES	cli_nombre	Nombre del Cliente	varchar	80	no	no
CLIENTES	cli_direccion	Dirección del Cliente	varchar	80	no	no
CLIENTES	cli_telefono	Teléfono del Cliente	varchar	12	no	no
CLIENTES	cli_mail	Mail del Cliente	varchar	40	no	no
CUENTAS	cue_nocuenta	No. De cuenta	int	6	si	no
CUENTAS	cue_cedula	Cédula del Cliente	varchar	10	no	si
CUENTAS	cue_estado	Estado del la cuenta	varchar	7	no	no
CUENTAS	cue_fecha	Fecha	date		no	no
SALDOS	sal_nocuenta	Número de Cuenta	int	6	no	si
SALDOS	sal_saldo	Saldo de la Cuenta	double	8,2	no	no
TRANSACCIONES	tra_notransaccion	Número de Transacción	int	10	no	no
TRANSACCIONES	tra_nocuenta	Número de Cuenta	int	6	no	si
TRANSACCIONES	tra_tipo	Tipo de Cuenta	varchar	7	no	no
TRANSACCIONES	tra_fecha	Fecha de la Transacción	date		no	no
TRANSACCIONES	tra_valor	Valor de la Transacción	double	8,2	no	no
TRANSACCIONES	tra_saldo	Saldo de la Cuenta	double	8,2	no	no

4.3.1 REALIZACIÓN DE LA APLICACIÓN RESTAURANTE

Contamos con un menú principal en el cual se procederá a elegir la opción de clientes.

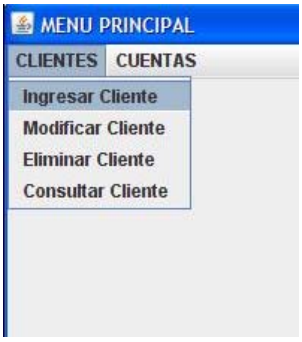

Figura86: Aplicación –MenúPrincipal Clientes

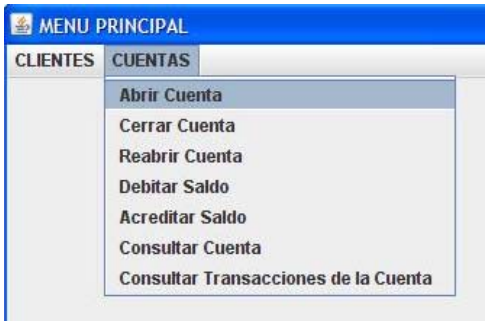

Figura 87: Aplicación-Menú Pricipal Cuentas

CLIENTES

Figura88: Aplicación –Ingreso Clientes

Se ingresarán los datos del cliente como cédula, nombre, dirección, teléfono y mail.

Figura89: Aplicación –Modificación Clientes

Se modificarán los datos del cliente en caso de haber sido mal ingresados basándose en el número de cédula.

Figura90: Aplicación – Consulta de clientes

Se consultarán los datos basándose en el número de cédula

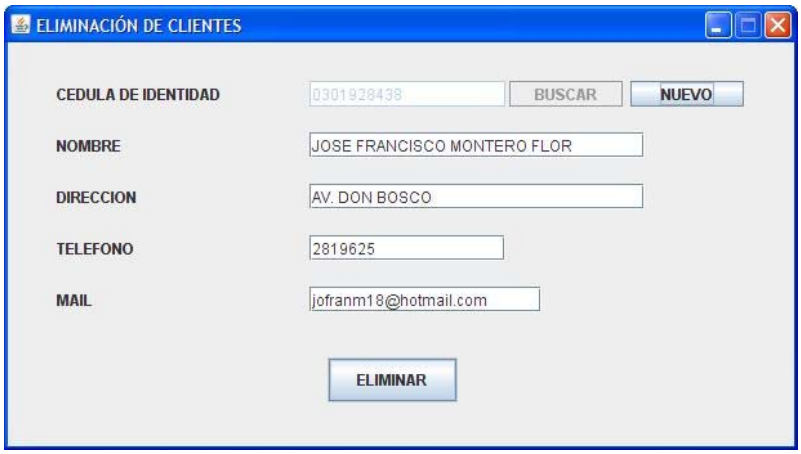

Figura91: Aplicación –Eliminar clientes

Para eliminar un cliente se lo hace a través de la cédula y automáticamente se eliminan todos sus datos y sus cuentas.

CUENTAS

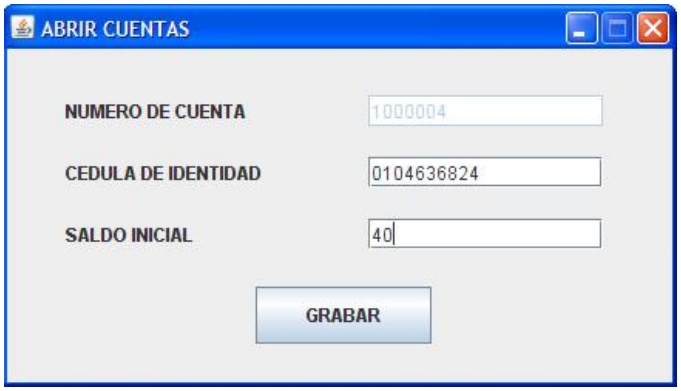

Figura 92: Aplicación – Abrir Cuenta

Se procederá a abrir una cuenta, este número se incrementa automáticamente, se pide la cédula al cliente y se ingresa el saldo inicial el mismo que tiene que ser mayor a $5

Figura 93: Aplicación –Cerrar Cuenta

Mediante la cédula de identidad y el número de cuenta se procede a la búsqueda y se la cierra, reembolsándole el sobrante de su saldo.

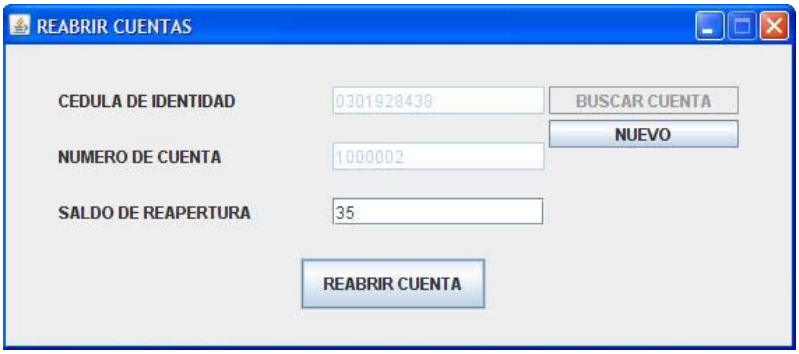

Figura 94: Aplicación –Reabrir Cuentas

Con el saldo y el número de cuenta se procede a reabrir una cuenta siempre y cuando el saldo a ser acreditado sea mayor a $5

Figura95: Aplicación –Debitar Saldo

Solo se podrá debitar de la cuenta con el número de cédula y el número de la cuenta y siempre y cuando no se quede el saldo restante con un valor menor o igual a $1.

Figura96: Aplicación –Acreditar Saldo

Se puede acreditar cuando el cliente lo desee en un estado abierto u obligatoriamente cuando el saldo este en estado crítico es decir igual o menor a $1.

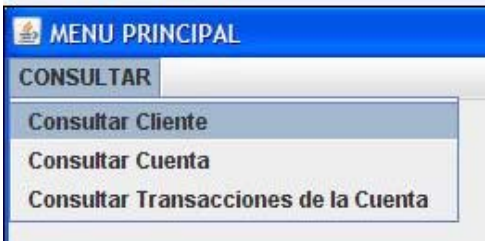

Figura 97:Aplicación-Menú Consultas

Muestra el nombre del cliente y los datos de la cuenta.
La siguiente consulta nos muestra mediante la CI y el número de la Cta., el nombre del cliente, el estado, la fecha de apertura, y el saldo

Figura 98:Consulta de Transacciones

Mediante la cédula de identidad, podemos consultar todas las transacciones que han sido realizadas por una persona.

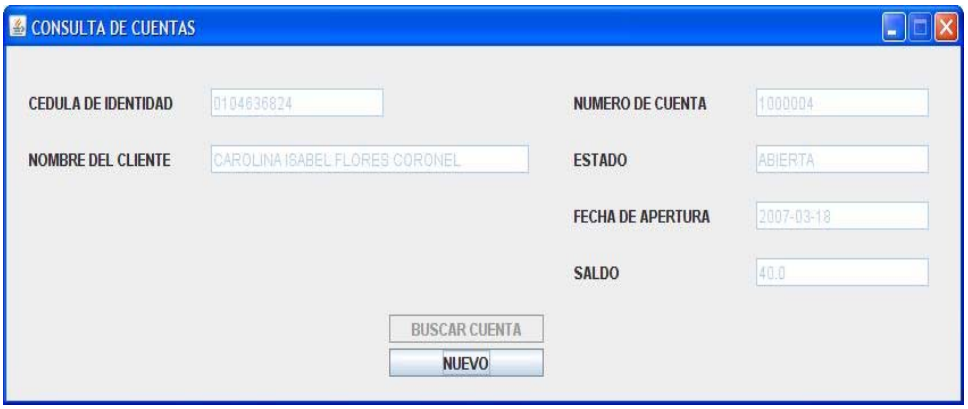

Figura 99: Aplicación –Consulta Cuentas

A continuación se detallará la parte que corresponde a los REPORTES, dentro del entorno de Eclipse podremos acceder a un Reporte que generamos utilizando BIRT, la siguiente imagen muestra el Layout del mismo:

Figura 100: Aplicación – Layout Reporte

Por último mostraremos la vista preliminar del reporte que creamos, en el cual se muestra la cédula y el nombre de cada cliente y una lista de todas las cuentas que pertenecen a este cliente con su estado y su saldo:

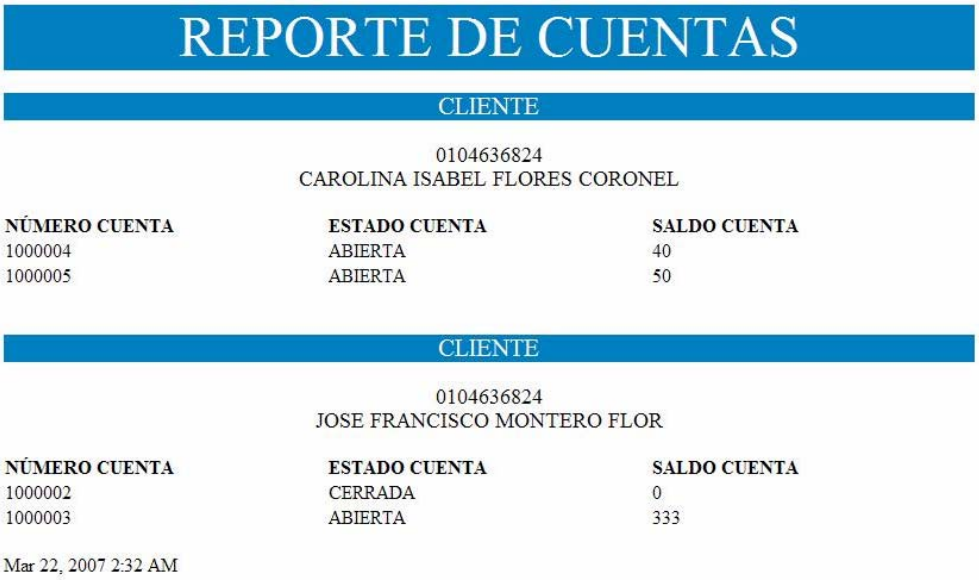

Figura 101: Aplicación – Vista Preliminar Reporte

CAPITULO 5

CONCLUSIONES

Eclipse es una herramienta muy potente y totalmente gratuita para desarrollo Java, creada por Eclipse.org en sociedad con IBM y otros proveedores de productos para Java, puede funcionar en varios sistemas operativos como Windows, Linux, Solaris.

Cuando se están desarrollando proyectos es importante poder utilizar una herramienta de desarrollo, que conozcan todos los desarrolladores, esto hará que aumente la productividad, puesto que si es un entorno de desarrollo como Eclipse viene con wizards (asistentes) que ayudan en la codificación, tiene detectores de errores que hace que el código desarrollado sea de mayor calidad o permite compartir código entre un grupo de desarrolladores. Eclipse puede integrar varios productos en un único entorno, que para la gente que está empezando es mucho más sencillo de utilizar. Todo esto hace que los tiempos de desarrollo sean menores y por consiguiente los costes también.

CAPITULO 6

RECOMENDACIONES

Partiremos en primera instancia del análisis del entorno java para posteriormente enfocarnos en la Herramienta Eclipse.

Para el caso de Java:

En cuanto al sistema operativo no existe ningún problema, debido a que existen versiones para casi todos.

Al diseñar una aplicación web, los mínimos requerimientos son: Intel Pentium de 166Mhz, 32 MB de RAM y 125 MB de disco duro. Para propósitos prácticos esto no es factible, debido a que la plataforma se cargará pero no se podrá visualizar nada.

Recomendamos Pentium III, mínimo 256MB de RAM, esto dependerá en gran medida de la complejidad del programa.

Para el caso de Eclipse:

El aplicativo ocupa más memoria. En cuanto al Sistema Operativo no hay limitantes puesto que están disponibles versiones para todos.

Lo primordial es tener en cuenta la memoria RAM.

Usualmente se sugiere como mínimo un Pentium III y 256 MB de RAM, pero en base a nuestro estudio, recomendamos Pentium III y al menos 512 MB de memoria.

BIBLIOGRAFIA

Definición de Código Abierto - GleduWiki
http://wiki.gleducar.org.ar/wiki/Definici%C3%B3n_de_C%C3%B3digo_Abierto
Fecha de Ingreso al Sitio:
05-ENERO-2007

Help - Eclipse SDK
http://help.eclipse.org/help32/index.jsp
Fecha de Ingreso al Sitio:
05-ENERO-2007

Las largadas de Sneb » Blog Archive » Instalación de Visual Editor para Eclipse
http://antares.escomposlinux.org/?p=110
05-ENERO-2007

Tutoriales en AdictosAlTrabajo: Java, J2EE, Visual C++, Linux, UML, OOP y mucho más
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=callisto
05-ENERO-2007

El Desarrollo del Framework Orientado al Objeto
http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html
05-ENERO-2007

¿Que es un plug-in? - Definición de plug-in
http://www.masadelante.com/faq-plug-in.htm
Fecha de Ingreso al Sitio:
06-ENERO-2007

Visual Editor Project
http://www.eclipse.org/vep/WebContent/main.php
16-ENERO-2007

Visual Editor FAQ
http://www.eclipse.org/vep/WebContent/faq.html
16-ENERO-2007

Curso de Java - Monografias.com
http://www.monografias.com/trabajos/java/java.shtml
22-ENERO-2007

Aprenda Java
http://mat21.etsii.upm.es/ayudainf/aprendainf/Java/Java2.pdf
22-ENERO-2007

MySQL Hispano - La comunidad de usuarios de MySQL

http://www.mysql-hispano.org/page.php?id=2
23-ENERO-2007

WikiLearning.com - Cómo configurar MySQL
http://www.wikilearning.com/capitulo.php?id_contenido=9433&order=1
23-ENERO-2007

WikiLearning.com Completo tutorial de MySQL
http://www.wikilearning.com/capitulo.php?id_contenido=4467&order=1 -->
ESTE
23-ENERO-2007

¿Qué es MySQL? :: Bases de datos MySQL. Contenidos eSePé Studio
http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm
23-ENERO-2007

MySQL AB :: MySQL 5.1 Reference Manual :: 1.4.1 What is MySQL?
http://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html
23-ENERO-2007

GLOSARIO

Ant Herramienta Open-Source utilizada en la compilación y creación de programas Java.

APIs aplication programming interface (swing awt) conjunto de controles

CVS (concurrent version system) Sistema de Control de Versiones

GNU GPL (General Public License o licencia pública general)
Licencia creada por la Free Software Foundation a mediados de los 80.

GUIs" Interfaces Gráficas de Usuario

IDE Entorno Integrado de Desarrollo (Integrated Development Environment)

JUnit Herramienta utilizada para realizar pruebas unitarias en Java.

JRE (java runtime enviroment) entorno olbigatorio para la ejecucion de programasen java. Compuesto por jvm y por el conjunto de aPIs de j2se (jvm +apis=jre)

J2SE Java2 Standard Editíon

J2EE Java2 Enterprise Edition

J2ME Java Micro Edition

JVM Java Virtual Machine

JDK Java Development Kit

JVM (Java Virtual Machine)

OTI (Object Technology International)

OSGi Open Services Gateway initiative (Puerta de Entrada de Servicios)

Su objetivo es el definir las especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios. Fue pensado principalmente para su aplicación en redes hogareñas y por ende en la llamada informatización del hogar.

OSI Open Source Initiative

SDK (software development kit) Conjunto de herramientas para compilación, documentación y depuración de errores de
aplicativos java, está compuesto por jre y por sus herramientas de desarrollo.

SWT (Standard Widget Toolkit)

SDK (Software Development Kit)
Conjunto de herramientas para compilación, documentación y depuración de errores de aplicativos Java. El SDK está compuesto por JRE y por sus herramientas de desarrollo.