1 Eclipse

1.1 Sumario

- 1 Introdución
- 2 Instalación
- 3 Primeiros pasos
- 4 Creación dun proxecto
- 5 Corrección e depuración de erros
    - 5.1 Corrección
    - 5.2 Depuración (*debug*)
- 6 Importación de librerías externas
    - 6.1 Uso de ficheiros .JAR
    - 6.2 Acceso ao código fonte das librerías
- 7 Plugins
    - 7.1 Visual Editor: Creación de contornos gráficos de usuario
        - 7.1.1 Instalación Visual Editor (VE) en Eclipse Galileo.
        - 7.1.2 Exemplo

2 Introdución

Eclipse é un IDE (*Integrated Development Environment*), é dicir, un contorno integrado de desenvolvemento para Java. Os IDE facilítannos o desenvolvemento de aplicacións. Habitualmente permiten, entre outras cousas: xestionar proxectos nos que participan diversas (ou moitas) clases, navegar facilmente polas clases Java e os seus métodos, depurar erros, etc. Todo iso pode resultar nunha redución importante do tempo de desenvolvemento. O seu uso é recomendable para desenvolver aplicacións de certo tamaño.

Hai bastantes IDE para Java. De todos eles hai dous que son de código aberto e que se usan amplamente: Netbeans e Eclipse. Eclipse é máis completo que Netbeans e proporciona todas as prestacións a un bo IDE, tendo unha comunidade de desenvolvedores moi importante que facilitan a inclusión de novas funcionalidades a través de *plugins*.

Eclipse está desenvolvido na súa totalidade en Java. Xa que logo, está accesible para calquera plataforma que soporte Java. Require a versión JDK 1.3 ou superior.

3 Instalación

Instalar Eclipse é sinxelo. Hai que descargalo da páxina web do proxecto e seguir as instrucións.

Unha vez instalado xa poderemos arrincalo. Tarda un pouco (non só a primeira vez, senón todas), aínda que unha vez en execución é bastante rápido. Eclipse é un IDE pensado para poder traballar en calquera linguaxe de programación e non só en Java.

- **Ligazón de Interese:** Instalación e Configuración Eclipse Galileo

4 Primeiros pasos

A primeira vez que lancemos Eclipse temos que escoller un espazo de traballo, que é un cartafol onde o programa almacenará os nosos desenvolvementos. A continuación veremos unha ventá de benvida (Welcome) que hai que pechar para poder empezar a traballar.

Na parte principal da ventá podemos ver diferentes partes:

- O *Package explorer*, que conterá os proxectos Java cos que se traballe. Cada un dos proxectos corresponderá a un programa que esteamos desenvolvendo (ou xa desenvolvido).
- A ventá *Outline*, que permite ver información referida ao elemento co que esteamos traballando en cada momento.
- A ventá inferior serve para dar información. Por exemplo, é onde aparecen os erros de compilación.
- A parte principal da ventá é onde poderemos editar os nosos ficheiros .java.

Eclipse permite traballar con diferentes perspectivas (a anterior é a perspectiva Java). Pode seleccionarse unha determinada perspectiva seleccionando Window->Open Perspective. Teremos dispoñibles dúas perspectivas máis: *Java browsing* e *Debug*. A primeira serviranos para

movernos comodamente polas clases e métodos destas; mentres que a segunda serviranos para, unha vez que xa teñamos o noso programa codificado, depuralo.

Cando saiamos do IDE gardarase a nosa configuración, de maneira que, cando o volvamos a executar, abrirase coas mesmas ventás e perspectivas que había xusto cando o pechamos. Iso resultaranos bastante cómodo cando esteamos desenvolvendo programas dun certo tamaño e teñamos diversos ficheiros e perspectivas abertas.

5 Creación dun proxecto

A continuación crearase un proxecto en Java para Eclipse. O programa de exemplo é sinxelo e ten unicamente dúas clases, Moble e Cadeira. Segue os seguintes pasos:

- Desde a perspectiva Java prememos co botón da dereita sobre o *Package explorer*. Escollemos New --> Project. Ábrese unha ventá e escollemos Java project. Prememos sobre Next. Damos un nome ao proxecto. Por exemplo, ProbaMoble. Dámoslle a Finish. Neste momento xa temos o proxecto creado. Totalmente baleiro, iso si. Podemos ver como na perspectiva Java, no panel Package explorer aparece un elemento co nome ProbaMoble. Se o abrimos, vemos como contén un elemento chamado JRE System Library. É o contorno de execución de Java correspondente ao JDK que temos instalado na nosa máquina. Aparécenos debido a que Eclipse dá a posibilidade de utilizar un JDK concreto para cada proxecto (sempre que os teñamos instalados). De todas as maneiras, de momento, utilizaremos o que Eclipse colle por defecto, que é o que se corresponde á variable do sistema JAVA_HOME.
- Imos crear a nosa primeira clase. No Package explorer, facemos clic co botón dereito sobre ProbaMoble. Escollemos New --> Class e aparécenos unha ventá a partir da cal poderemos crear a nosa clase. Dos parámetros que podemos editar/modificar na ventá, unicamente poremos o nome da clase (Moble). Prememos co rato sobre Finish para crear a clase. A continuación, abrirase unha ventá no panel central onde se poderá editar o ficheiro Moble.java, para engadir métodos e atributos á clase Moble.
- Unha vez feito isto, engadimos os métodos da clase Moble. Para facelo, copia o seguinte código e pégao directamente no Eclipse:

```
public class Moble {
private String nome;

/**
         * Constructor Moble
         */
public Moble() {
}

/**
         * Método accessor que devolve o nome do moble como unha cadena de caracteres.
         * @return java.lang.String, devolve nome
         */
public String getNome() {
  return (nome);
}

/**
         * Método accessor que modifica o nombe do moble
         * @param valor do novo nome
         */
public void setNome(String nome) {
  this.nome = nome;
}

}
```

- Garda o ficheiro Moble.java en disco. Para facelo é necesario que teñas activa a ventá de texto correspondente a Moble.java. Esta operación tamén compilará automaticamente o ficheiro Moble.java. Se houbese algún erro aparecería no panel Problems). Por outra banda, intenta expandir (ou ben no Package explorer ou ben no Outline) a clase Moble. Comproba como agora aparecen todos os métodos que engadimos. Cando selecciones algún deles, a ventá asociada a Mobles.java actualízase de maneira que se mostra a parte do ficheiro Mobles.java correspondente a este método.
- Pasamos a crear agora a clase Cadeira. Realizamos a mesma operación que antes: sobre o Package explorer, seleccionamos a clase Moble e facemos clic co botón dereito do rato, seleccionando New --> Class. Agora pomos Cadeira como nome de clase. Nota que ao facer a operación seleccionando sobre a clase Moble, xa nos pon que o nome da superclase é Moble. Se non fose así, teriámolo que facer nós a man. Esta clase terá un construtor equivalente ao de Moble e terá método main. **Eclipse dános a posibilidade de crear estes métodos de xeito automático**. Para facelo, no último apartado da ventá de creación de clases seleccionamos public static void main e tamén *Constructors from superclass*. Acto seguido facemos clic sobre Finish.
- Con iso, crearíamos a clase Cadeira. Para crear os métodos desta clase copia o seguinte código directamente no Eclipse. Non esquezades

gardar o ficheiro en disco. Igual que pasaba coa clase Moble, agora temos acceso aos diferentes métodos da clase Cadeira no Package explorer:

```
import java.io.*;

public class Cadeira extends Moble {

private int prezo;

/**
         * Construtor Cadeira
         * @param prezo inicial da cadeira
         */
public Cadeira(int prezo) {
this.prezo = prezo;
}

/**
         * Método setter que modifica o prezo da cadeira
         * @param valor do novo prezo
         */
public void setPrezo(int prezo) {
this.prezo = prezo;
}

/**
         * Método getter que devolve o prezo dunha cadeira como un float.
         * @return float, devolve prezo
         */
public int getPrezo() {
  return prezo;
}

/**
         * Método que se corresponde co método principal para a execución dun
          * programa Java. Neste caso o método crea unha cadeira e le da entrada estándar
          * unha cadea de caracteres para asignarlle un nome.
         */
  public static void main(String[] args) throws Exception {
   Cadeira n = new Cadeira(50);
   String s;
   InputStreamReader ir;
   BufferedReader in;
   ir=new InputStreamReader(System.in);
   in=new BufferedReader(ir);
   System.out.print("Introduce o nome do moble (enter para acabar): ");
   s=in.readLine();
   while (s.length()>0) {
     n.setNome(s);
     System.out.println("O nome do moble é: "+n.getNome());
     System.out.print("Introduce o nome do moble (enter para acabar): ");
     s=in.readLine();
   }
 }
}
```

- Podedes abrir a perspectiva Java browsing (menú Window->Open perspective->Java Browsing, onde poderedes navegar polas clases e métodos do proxecto dun xeito diferente a comose fai na perspectiva Java.
- Imos executar o programa. Para iso, seleccionamos do menú Run a opción Run... Aparécenos unha ventá onde seleccionaremos Java Application e prememos sobre o botón New. Teremos que seleccionar o nome de proxecto (ProbaMoble) e o nome da clase na que está o método main que queremos executar como programa principal (seleccionaremos Cadeira). Dado que temos un só proxecto, é posible que estes valores xa estean establecidos por defecto, pero non sempre será así.
- Facemos clic sobre Run. No panel inferior aparece a consola onde se nos pedirá que introduzamos un nome para o moble, sucesivas veces, até que premamos enter.

Eclipse dispón de multitude de opcións que son unha boa axuda para os desenvolvedores. Non é obxecto deste titorial presentar todas estas funcións, pero si é recomendable que botes unha ollada á documentación/axuda do sistema para facerte unha idea das facilidades do IDE. Para facelo, preme en Help --> Help contents. E dentro da axuda unicamente é necesario que miredes os puntos *Workbench user guide* e *Java development user guide*.

6 Corrección e depuración de erros

Cando codificades atoparédesvos a miúdo con que cometedes erros. Estes erros poden ser tanto a nivel **sintáctico** como **semántico**. Os erros a nivel sintáctico corresponden a fragmentos de código que non seguen as normas da linguaxe Java. Estes erros son máis fáciles de detectar. Os erros a nivel semántico corresponden a fragmentos de código que cando se executan non fan exactamente o que nós esperabamos e poden ser bastante máis difíciles de detectar.

6.1 Corrección

Un IDE pódenos ser bastante útil para atopar rapidamente tanto os erros sintácticos como os erros semánticos. Nesta sección veremos un exemplo de código con erros e como Eclipse nos pode axudar a corrixir estes erros. Imos facelo paso a paso:

- Vai á perspectiva Java, contrae todos os proxectos que teñas expandidos e pecha todas as ventás de texto que teñas abertas. Preme co botón dereito do rato sobre os proxectos que podas ter abertos e escolle close para pechalos.
- Crea un proxecto denominado ProbaErros e crea unha clase chamada Dobre. Copia o seguinte código exactamente como está e pégao na clase. Unha vez feito iso, garda o ficheiro Dobre.java en disco.

```
/** Esta clase calcula o dobre dun número introducido por liña de comandos */
public class Dobre {
  /** Método principal utilizado pola máquina virtual para executar a clase*/
    public static void main(String[] argv) {
        // Definición das varibles.
        int numero;
        int resultado;
        string mensaxe;
        System.out.println("Cálculo do dobre dun número");
        if (argv.length=<10)||(argv.length<1) {
             System.out.println("Error: Debes introducir polo menos un número e como máximo 10");
    } else {
             for(i=0;i<=argv.length;i++)
             {
                  numeor=Integer.parseInt(argv[i]);
                  resultado=numeor*2;
                  mensaxe="O dobre de "+" "+numero+" "+"é"+" "+resultado;
                  System.out.println(mensaxe);
             }
        }
    }
}
```

- Eclipse marca cun sinal vermello a liña correspondente ao if. Nesta liña, o símbolo '=<' aparece subliñado en vermello. Iso quere dicir que **o compilador atopou un erro sintáctico**. Evidentemente, o símbolo '=<' ten que ser en realidade '<='. Facemos o cambio e gardamos en disco. Segue habendo un erro na mesma liña, pero agora o que se subliñou é o operador '||'. O que pasa é que toda a expresión do if ten que ir entre paréntese. Pómola entre paréntese e volvemos gardar en disco.
- Agora aparécennos tres luceciñas amarelas. Estas luceciñas indícannos tamén erros sintácticos que o IDE considera doutra categoría, pero ao final son erros igualmente. Imos ver que pasa: sempre debemos facer as correccións empezando por arriba, xa que a corrección dun erro anterior podería corrixir un erro posterior. Na liña da primeira luceciña está subliñada a palabra string. Evidentemente é String, empezando con maiúsculas. Facemos o cambio e volvemos gardar.
- Imos á seguinte luceciña: fai referencia á liña da variable i porque non está definida. Polo tanto, engadimos unha liña ao principio do método main que incúa:

  ```
  int i;
  ```

  ...e gardamos. Agora xa non queda ningún erro sintáctico e podemos executar a aplicación.

6.2 Depuración (*debug*)

O algoritmo calcula o dobre dos números introducidos como parámetros na liña de comandos. Podemos modificar estes parámetros indo á pestana *Arguments*. Modificamos os *program arguments* pondo entre 1 e 10 números enteiros separados por espazos. Por exemplo, podemos pór: 3 15 33 e executamos a aplicación. A saída será:

```
Cálculo do dobre dun número
Erro: Debes introducir polo menos un número e como máximo 10
```

Para depurar a aplicación e ir executándoa paso a paso hai que seguir os seguintes pasos:

- Vai á clase Dobre, método main. Fai dobre clic sobre el. Abrirase o ficheiro Dobre.java e visualizarase o método en cuestión.
- Sitúate na liña do if, **á esquerda** de todo (de feito, fóra do texto). Fai clic co botón da dereita e aparece un menú. Selecciona Add breakpoint. Asegúrate de que estás fóra da área de texto (se estás na área de texto aparece un menú diferente). Con isto consíguese que o programa se execute até chegar a esta liña, que se parará.
- Volve executar a aplicación premendo en debug (ten que estar en modo debug). Ábrese a perspectiva debug, coa liña do if marcada en azul, indicando que a execución está parada neste punto esperando instrucións nosas.
- Imos ver que valor ten argv.length. Seleccionamos da mesma liña do if a subexpresión argv.length e facemos clic co botón da dereita e seleccionamos Display. Vemos que o seu valor é realmente 3; cousa que nos fai pensar que a expresión do if é incorrecta. Efectivamente, tería que ser:

  ```
  ((argv.length>10)||(argv.length<1))
  ```

  Na parte superior dereita vemos tamén o valor da variable args.
- Facemos o cambio, gardamos o ficheiro e detemos a execución. Para deter a execución facemos clic co botón dereito sobre o proceso que se está executando (parte superior esquerda da fiestra) e seleccionamos Terminate and remove.
- Acto seguido volvemos executar o programa en modo Debug. Efectivamente, probándoo paso a paso podemos comprobar como agora a expresión é correcta e non sae a mensaxe de erro pola consola. Podemos facer Run --> Resume para executar a aplicación de golpe.
- Na consola obtemos o seguinte:

```
Cálculo do dobre dun número
O dobre de  3 é 6
O dobre de  15 é 30
O dobre de  33 é 66
```

E se nos fixamos no panel de debug, na esquina superior esquerda, infórmanos de que o thread está suspendido porque se produce unha excepción de tipo ArrayOutOfIndexException. Este comportamento de Eclipse é moi útil, dado que nos informa diso e á vez conxela a execución con vistas a que poidamos examinar cal é o estado que levou a xerar a excepción (podemos examinar variables, avaliar expresións, ...).

- Imos ao panel de variables, e vemos como o valor de i é 3. Efectivamente, argv é unha táboa de 3 posicións (de 0 a 2) e estamos intentando acceder á posición 3. Iso indícanos que temos que cambiar a condición do

prezo. Facémolo do seguinte xeito:

```
for(i=0;i<argv.length;i++)
```

- Gardamos, volvemos executar e finalmente a execución funciona e acaba con normalidade.

7 Importación de librerías externas

Calquera proxecto creado con Eclipse, por defecto, pode facer uso das clases que o mesmo JDK leva incorporadas (java.io.FileOutputStream ...). Agora ben, moitas veces quereremos usar outras clases que proporcionen funcionalidades diferentes ás proporcionadas polas clases do propio JDK e que non formarán parte do noso proxecto. Por exemplo, se estamos desenvolvendo unha aplicación que visualiza gráficos en tres dimensións é posible que queiramos usar unha librería de clases específica para a visualización deste tipo de gráficos, por exemplo a librería OpenGL para Java.

7.1 Uso de ficheiros .JAR

Estas librerías normalmente adoitan vir nun (ou diversos) ficheiros con extensión .JAR que contén as clases da librería. Os **ficheiros JAR** (do inglés, **J**ava **AR**chive) son como os ficheiros ZIP: serven para empaquetar e comprimir outros ficheiros. Así como os ficheiros .ZIP adoitan ser de uso xeral, gardando calquera tipo de ficheiros no interior, os ficheiros JAR normalmente úsanse para agrupar conxuntos de clases Java. Un ficheiro JAR pódese crear, examinar e modificar coa ferramenta Jar que vén co JDK (facede Jar -Help para ver como executar esta ferramenta; ou tamén podedes acceder á propia documentación do JDK). Instrucións para GNU/Linux:

Ver contido

```
$ jar tvf ficheiro.jar
```

Comprimir

```
$ jar cvf ficheiro.jar ficheiro_1 ficheiro_2 ficheiro_3 ... ficheiro_n
```

Descomprimir

```
$ jar xvf ficheiros.jar
```

Para poder usar desde un proxecto Eclipse as clases contidas nun ficheiro .jar, temos que facer o seguinte:

1. Vai ás propiedades do proxecto desde o que queremos facer uso do .jar (facendo clic co botón dereito sobre o nome do proxecto na perspectiva Java).
2. Selecciona Java build path e, unha vez feito iso, escolle a pestana Libraries.
3. Fai clic sobre Add external JARs...
4. Selecciona os ficheiros JAR que queiras usar.

Unha vez feito iso, xa poderemos usar as clases contidas nos .JAR desde as clases do proxecto sen ningún problema.

7.2 Acceso ao código fonte das librerías

Moitas veces, as librerías veñen tamén co código fonte (os ficheiros .java correspondentes ás clases da librería). Algunhas veces pódenos interesar acceder a este código fonte do mesmo xeito que facemos co noso código fonte. Iso seranos útil unicamente nalgúns casos concretos, especialmente cando estamos en fase de depuración dun programa que fai uso da librería externa.

8 Plugins

Eclipse ten unha comunidade de desenvolvedores de tamaño considerable. Isto faino un IDE moi atractivo xa que existe a posibilidade de engadir funcionalidades ao mesmo, normalmente, mediante plugins desenvolvidos por terceiros.

8.1 Visual Editor: Creación de contornos gráficos de usuario

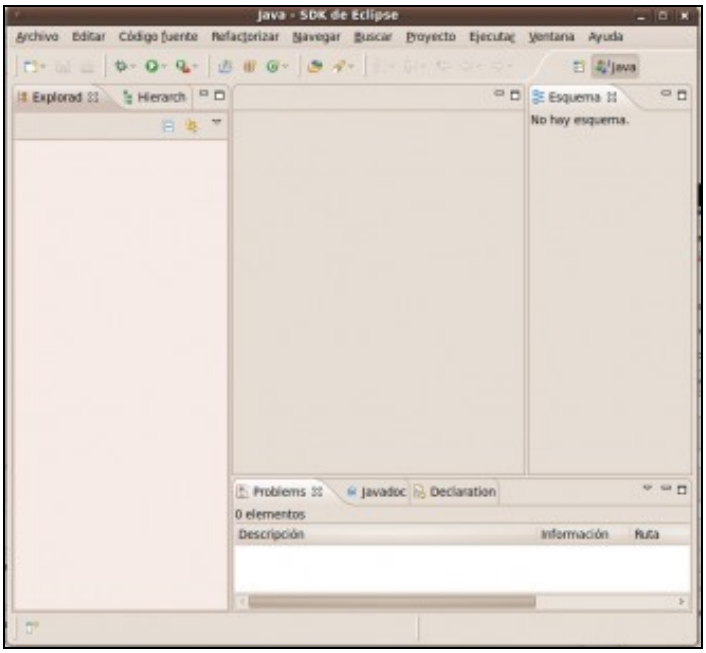
O Visual Editor (VE) é un plugin independente da plataforma que permite crear GUI (*Graphical User Interfaces*) dun xeito sinxelo, ao estilo das linguaxes de pogramación visual.

8.1.1 Instalación Visual Editor (VE) en Eclipse Galileo.

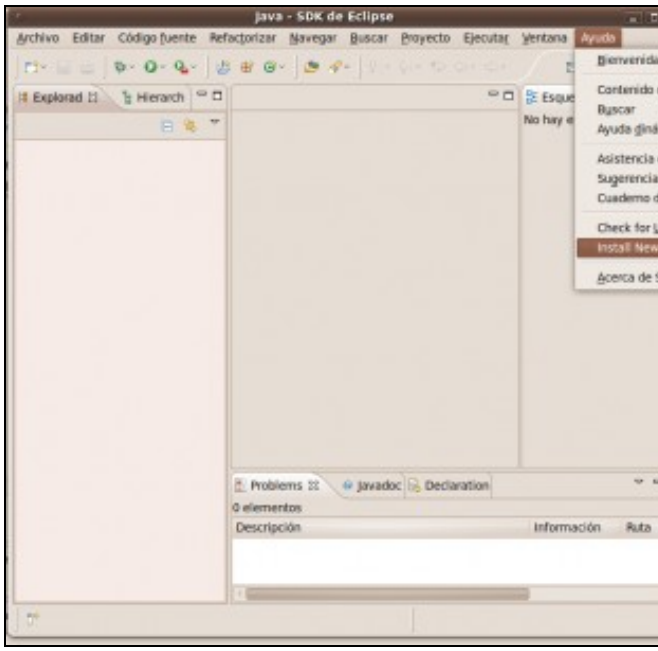
NOTAS:

1. **Previamente temos instalado e configurado o Eclipse Galileo no Idioma Español. Ver a seguinte ligazón: Instalación e Configuración Eclipse Galileo**
2. Documentación de referencia para a instalación de **VE**: http://wiki.eclipse.org/VE/Update
3. Picar nas imaxes para velas no tamaño orixinal

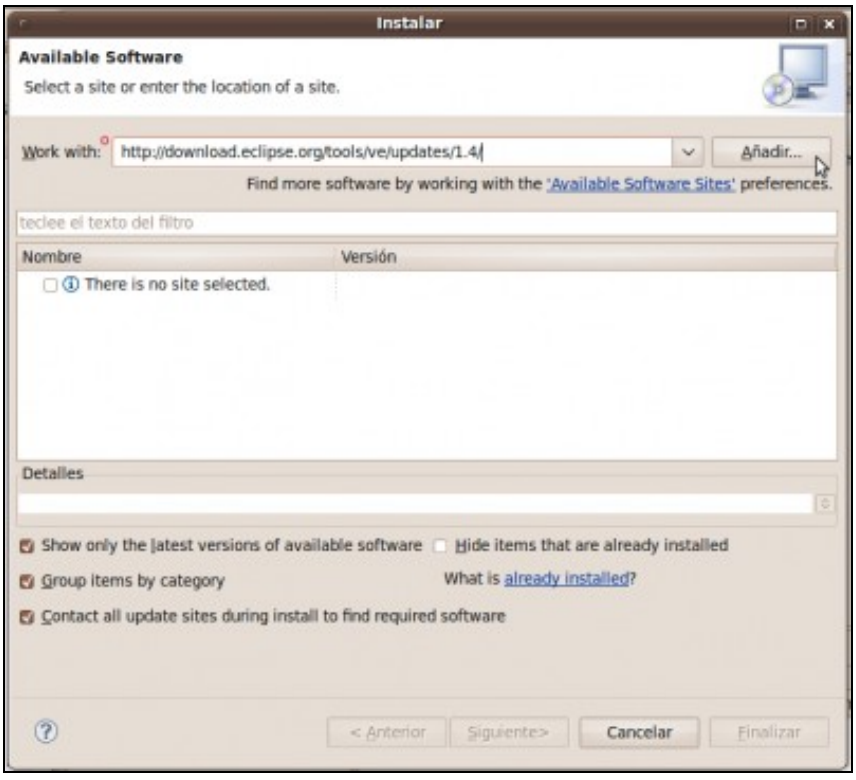
Lanzar **Eclipse Galileo** e proceder como se amosa nas imaxes seguintes:



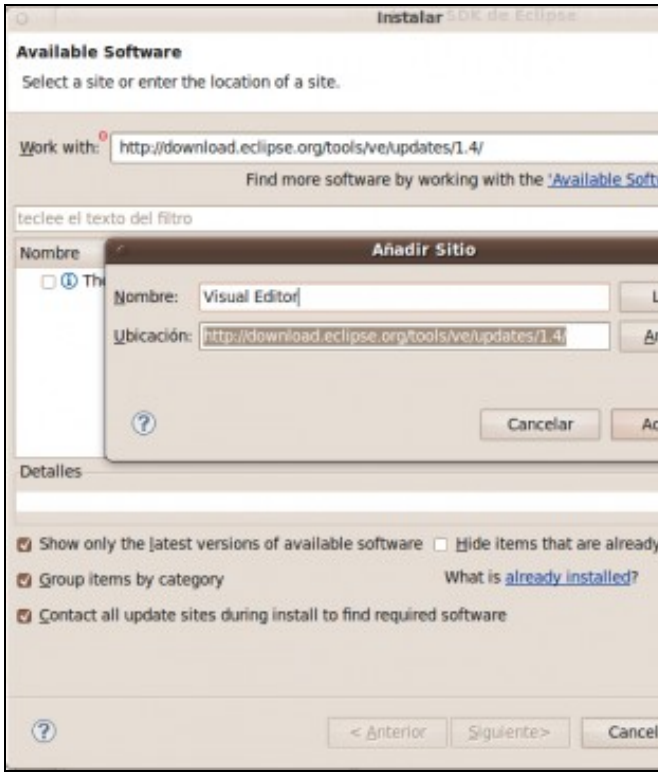
a. Interface Gráfica Eclipse Galileo



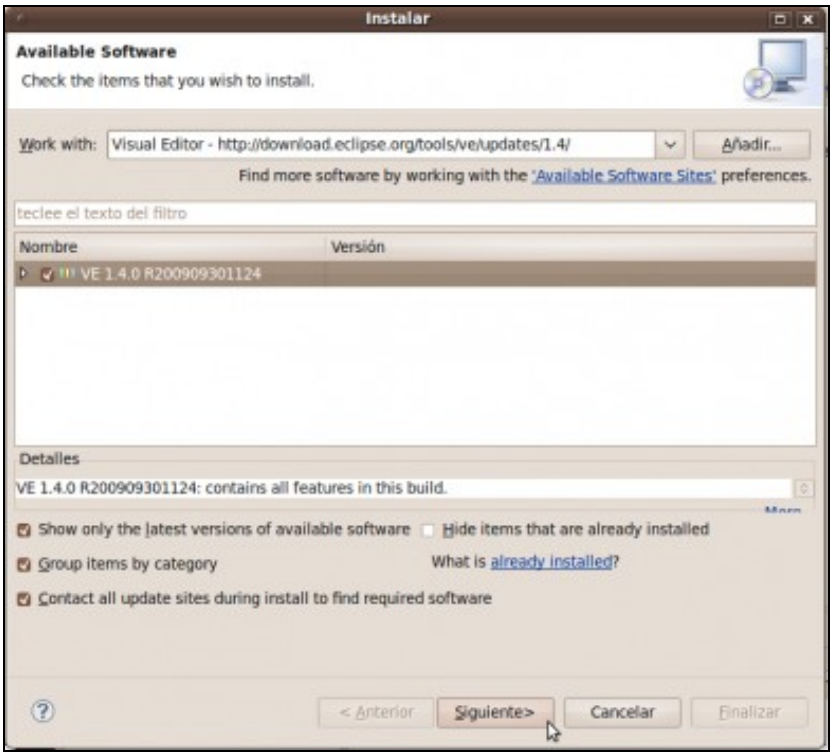
b. Menú Ayuda-->Install New Software...



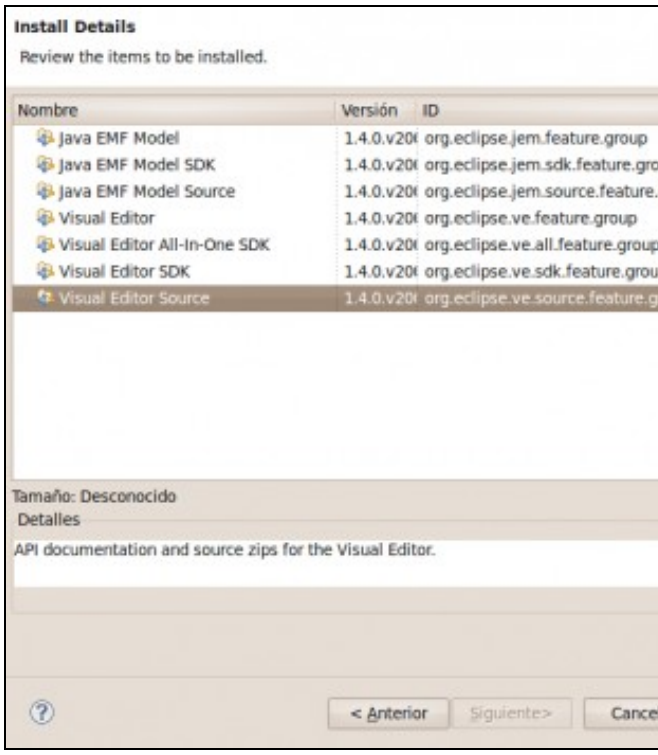
c. En Work with por a ligazón http://download.eclipse.org/tools/ve/updates/1.4/ .
Picar en **Añadir**.



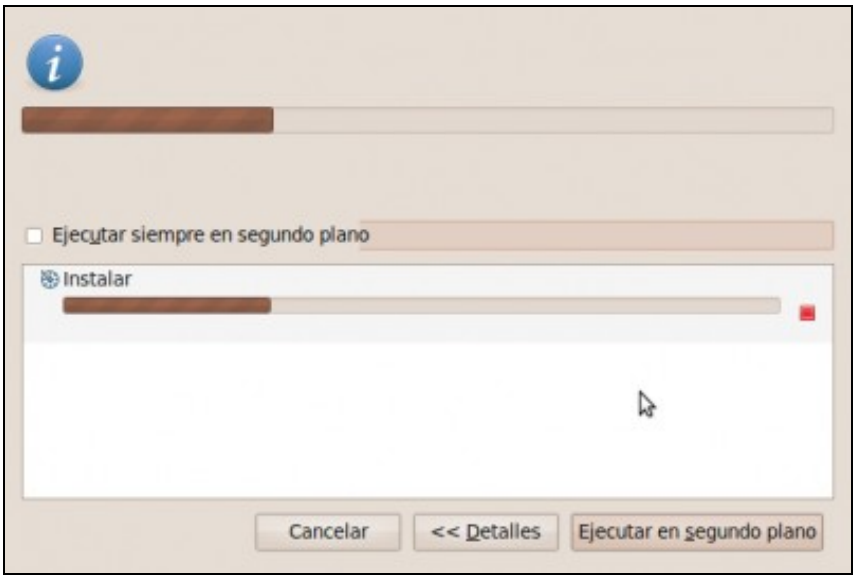
d. Engadir sitio. Por un nome para a identificación do sitio, por exem



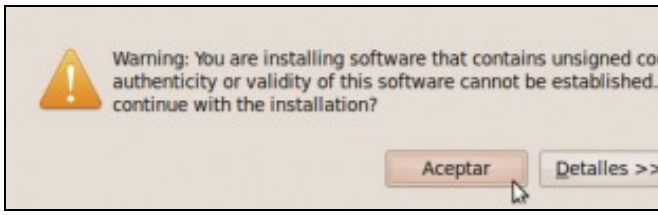
e. Enlázase a URL e os paquetes posibles a descargar. Picar en **Siguiente**.



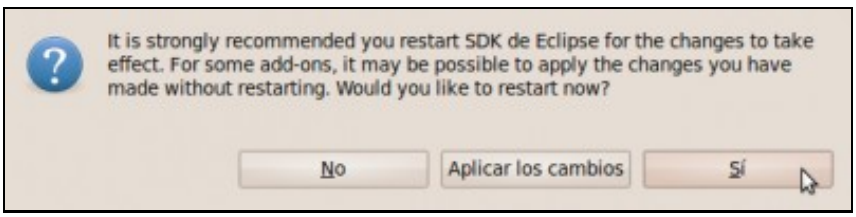
f. Paquetes a instalar. Se picamos nun paquete veremos unha desc
Picamos en **Finalizar** e comeza a instalación



g. Instalando...



h. Aceptar Warning



i. Fin da Instalación VE. Picar en **Sí** para reiniciar **Eclipse**.

--ricardofc [23/04/10]

8.1.2 Exemplo

Imos realizar unha aplicación de exemplo co Visual Editor:

1. Pulsamos no menú ?Archivo? -> ?Nuevo? -> ?Proyecto??. Seleccionamos o tipo de proxecto novo (neste caso Proyecto Java), prememos ?Siguiente? para continuar.
2. Indicamos o nome do proxecto (por exemplo probaVisualEditor), indicamos a ruta da aplicación e prememos "Finalizar".
3. Seleccionamos o novo proxecto creado e prememos co botón dereito do rato sobre el, prememos en ?Nuevo?-->?Visual class?. En caso de non aparecer ?Visual Class? prememos sobre ?Otros?? e seleccionaremos ?Java?--> ?Visual Class?.
4. Introducimos o nome do Paquete, seleccionamos a superclase javax.swing.JFrame e prememos ?Finalizar?.