Desenvolupament amb Eclipse

Professorat d'Informàtica i Multimèdia

UOC

Eclipse és un IDE (Integrated Development Environment: Entorn Integrat de Desenvolupament) per a Java. Un IDE ens facilita molt el desenvolupament d'aplicacions a partir d'una mida mitjana (per exemple, la pràctica de FP-II). Habitualment permeten, entre d'altres coses: gestionar projectes en les que participen diverses (o moltes) classes, navegar fàcilment per les classes Java i els seus mètodes, depurar errades, etc. Tot això pot resultar en una reducció important del temps de desenvolupament. És per això que, des de els estudis d'informàtica recomanem el seu ús per a desenvolupar aplicacions d'una certa mida (si bé no és obligatori).

Hi ha bastants IDEs per a Java. De tots ells, n'hi ha uns quants que són gratuïts; i d'aquest subconjunt n'hi ha dos que són de codi obert: Netbeans (www.netbeans.org) i Eclipse (www.eclipse.org). Després d'avaluar aquests darrers IDEs, des de els estudis d'informàtica de la UOC creiem que l'Eclipse resulta molt més confortable i còmode d'usar; i proporciona totes les prestacions que serien exigibles a un bon IDE. És per aquest motiu que el proporcionem en el CD de Programari Lliure i també que hem elaborat aquest document d'ajuda. ***Ara bé, si per qualsevol raó preferiu usar qualsevol altre IDE (o no usar-ne cap), sou ben lliures de fer-ho***.

L'Eclipse està desenvolupat en la seva totalitat en Java. Per tant, està accessible per a qualsevol plataforma que suporti Java. Requereix la versió JDK 1.3 o superior.

1. INSTAL.LACIÓ DE L'ECLIPSE

La instal.lació del Eclipse és bastant senzilla, heu de seguir els següents passos:

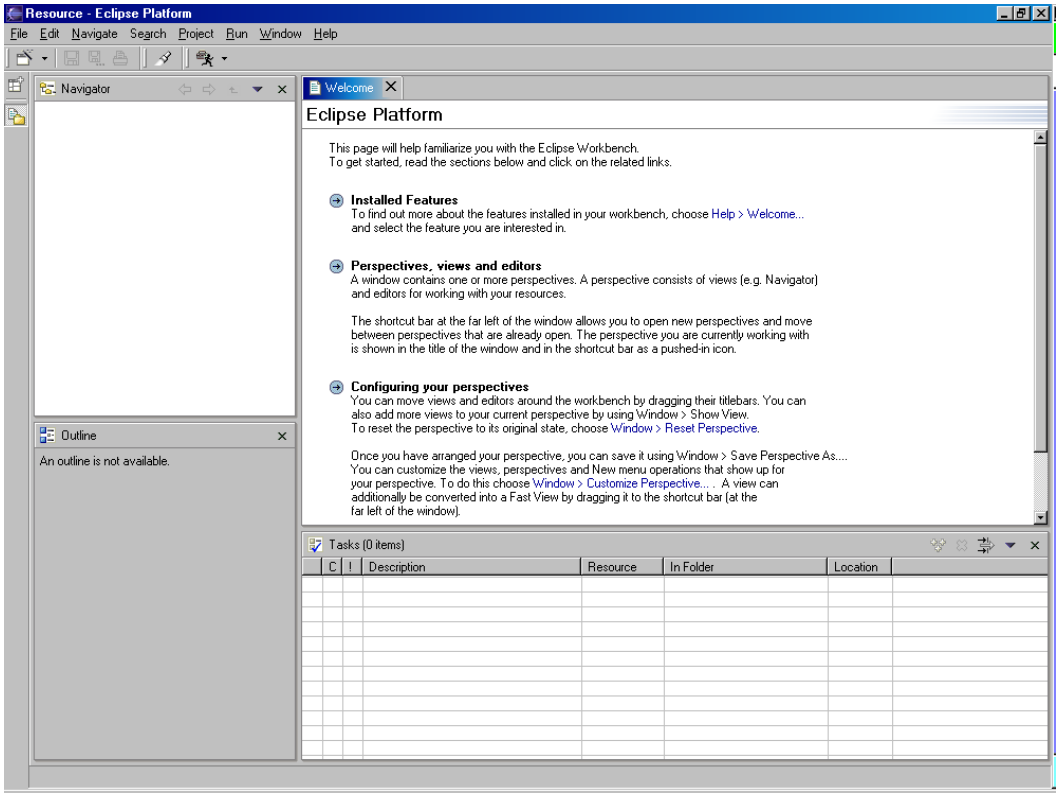
1. Tenir prèviament instal.lat el JDK 1.3 o superior.

2. Agafar el fitxer *eclipse-SDK-2.0.2-win32.zip* del directori *Java\Windows\Eclipse* del CD de programari lliure i descomprimir-lo en el directori pare d'on voleu tenir instal.lat l'Eclipse; per exemple a: *c:*\. Al descomprimir el fitxer es crearà un subdirectori anomenat eclipse, que és on estarà instal.lat l'IDE (per exemple: *c:\eclipse*).

   Si no teniu el CD de programari lliure, també podeu agafar l'Eclipse de l'adreça www.eclipse.org (teniu en compte però que són més de 50 MB i que per tant necessitareu una connexió a internet prou ràpida).

3. Un cop fet això l'Eclipse ja està instal.lat. Per a obrir-lo només cal executar el fitxer *eclipse.exe* del directori d'instal.lació de l'Eclipse. Per a accedir-hi més còmodament podeu crear una icona d'accés directe a aquest fitxer en el mateix escritori o allà on més us estimeu.

4. Executeu l'Eclipse. El primer cop que ho feu, us sortirà un missatge dient que s'està completant la instal.lació (en anglès), i a continuació se us ha d'obrir una finestra semblant a la següent:



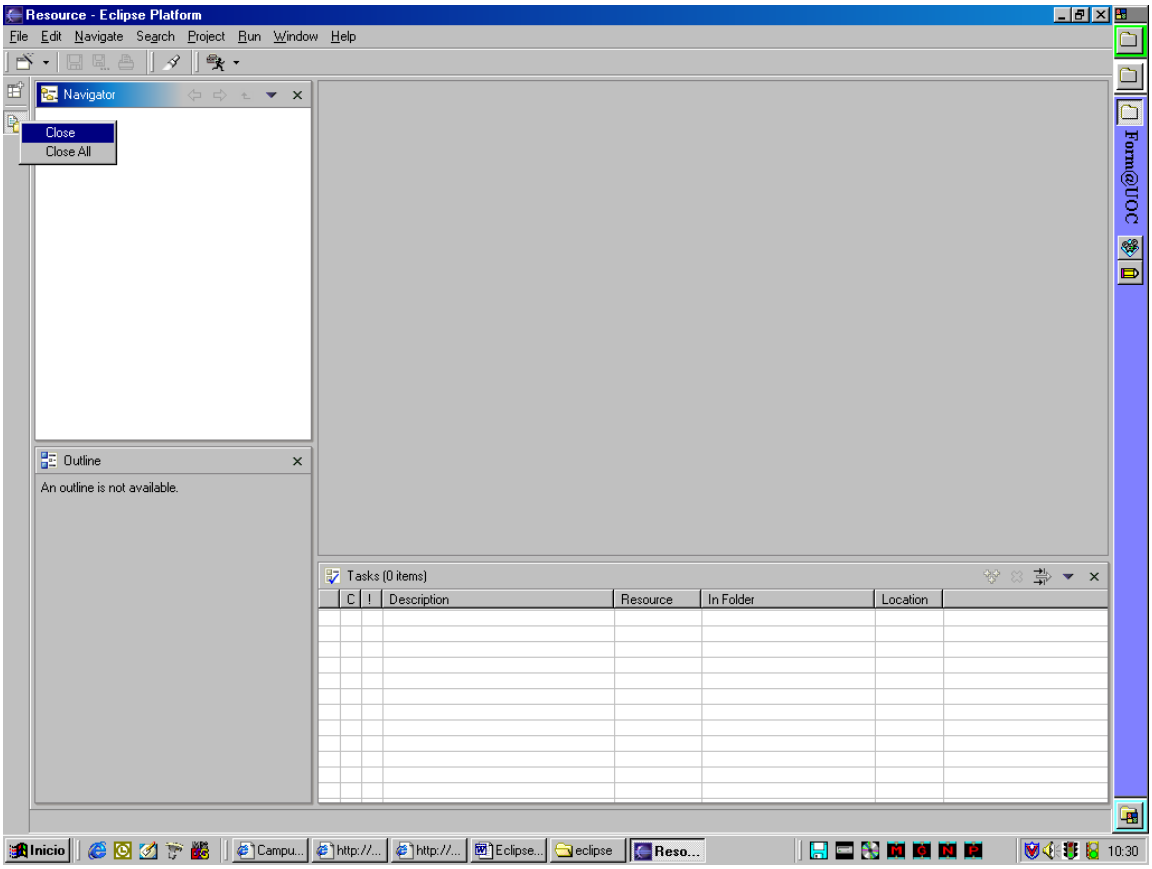
Notareu que triga una mica a engegar (no només el primer cop, sinó tots). De totes maneres, un cop en marxa és bastant ràpid.

L'Eclipse és un entorn pensat per a poder treballar en qualsevol llenguatge de programació, i no només en Java. Per aquest motiu, haurem de configurar-lo per a treballar amb Java (cosa, que, a més, es fa ben ràpidament).

L'Eclipse permet treballar amb diferents perspectives, que ens seran útils per a diferents tasques. La perspectiva que s'obre per defecte al executar l'Eclipse per primer cop és la perspectiva "Resource". Haurem de tancar aquesta perspectiva i obrir les perspectives corresponents a Java.

5. Tanquem la finestra de benvinguda (Welcome).

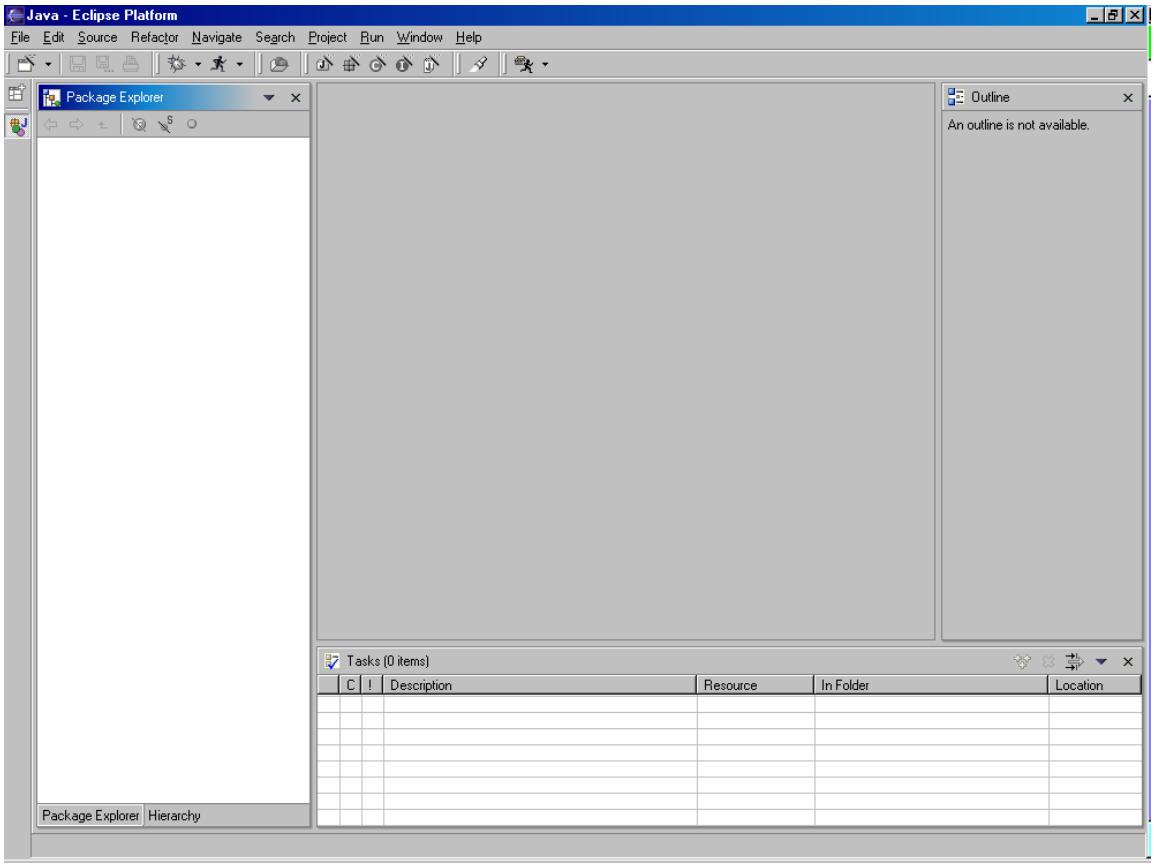
6. Per a tancar la perspectiva resource, hem d'anar a la icona situada a la part esquerra de la finestra de l'Eclipse (la barra de perspectives), fer click amb el botó dret del ratolí i seleccionar "close" (tal com es veu en el següent gràfic).



7. Un cop fet això, desapareix la icona i la finestra es queda en gris. Anem a la única icona que queda a la barra de perspectives (que ens permetrà obrir noves perspectives), fem click amb el botó esquerra i seleccionem 'other'.

   Un cop fet això, ens apareix una nova finestra on podem triar la perspectiva a obrir. Seleccionem “Java”. Això ens obrirà una perspectiva que ens permetrà treballar amb projectes Java. A continuació veiem com ens quedarà la finestra de l’Eclipse després d’obrir la perspectiva Java.

   A la part principal de la finestra hi podem apreciar diferents parts:
   - El “package explorer” contindrà els diferents projectes Java amb els que treballarem. Cada un dels projectes es correspondrà a un programa que estem desenvolupant (o haguem ja desenvolupat). Exemples de projectes podrien ser: pac2-fp2; o bé pràctica-fp2.
   - La finestra titulada “outline” ens permetrà veure informació referent a l’element amb el que estarem treballant en cada moment. Quan feu servir l’Eclipse ja veureu què us hi apareix.
   - La finestra “tasks” serveix per a donar informació. Per exemple, és on apareixen els errors de compilació.
   - Finalment, la part principal de la finestra (apareix totalment gris a la següent figura), és on podrem editar els nostres fitxers .java.



8. Igual que hem fet al punt anterior, obrim dues perspectives més: “java browsing” i “debug”. La primera ens servirà per a moure’ns còmodament per les classes i mètodes d’aquestes; mentre que la segona ens servirà per, un cop ja tenim el nostre algorisme codificat en Java, depurar-lo. Veurem el funcionament d’aquestes perspectives més endavant.

   Amb això ja hem instal.lat i configurat l’Eclipse. Quan sortim de l’IDE, es guardarà la nostra configuració; de manera que quan el tornem a executar, s’obrirà amb les mateixes finestres i perspectives que hi havia just quan l’havíem tancat.

   Això darrer ens resultarà força còmode quan estem desenvolupant programes d’una certa mida i tinguem diversos fitxers i perspectives obertes.

2. PRIMERS PASSOS AMB L’ECLIPSE

A continuació veurem com crear un projecte en Java per a l’Eclipse, de manera que ja puguem començar a desenvolupar els nostres programes Java usant l’Eclipse. El programa d’exemple serà senzillet: únicament dues classes: DNI i NIF.

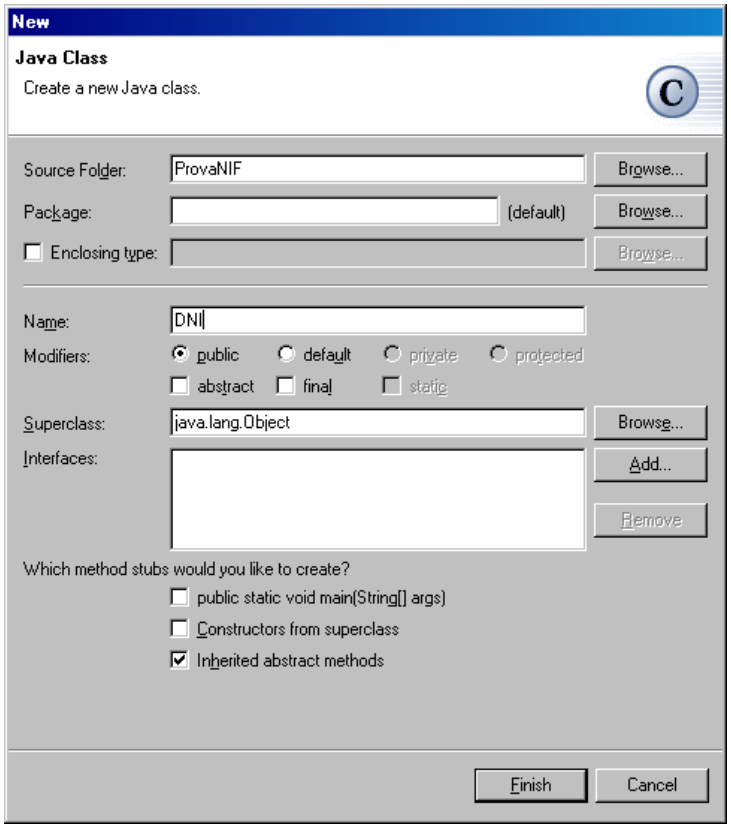
1. Des de la perspectiva Java, fem click amb el botó de la dreta sobre el package explorer. Triem new --> project. Se'ns obre una finestra i triem java project. Fem click sobre next.

   Donem un nom al projecte. Per exemple ProvaNIF. Li donem a Finish. En aquest moment ja tenim el projecte creat. Totalment buit això sí.

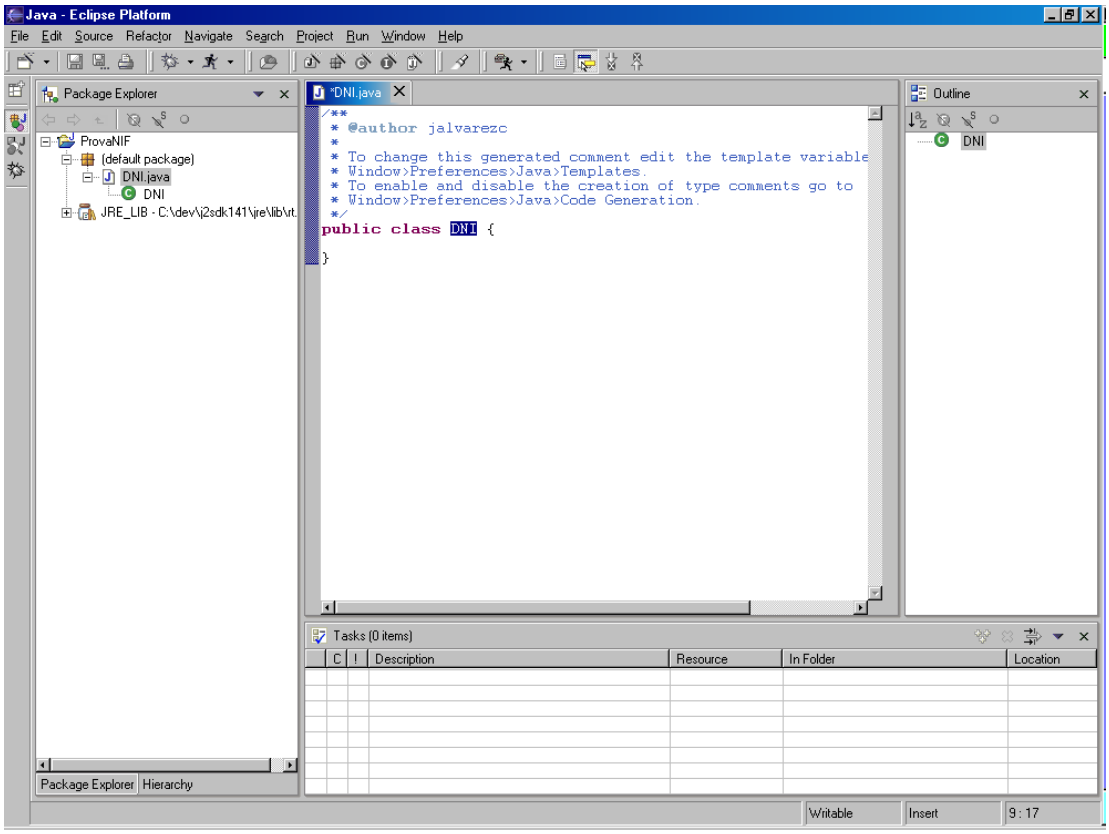
   Podem veure com a la perspectiva Java, al pannell 'package explorer' apareix un element amb el nom 'ProvaNIF'. Si l'obrim veiem com conté un element anomenat JRE_LIB. És la llibreria de classes de Java corresponent al JDK que tenim instal.lat a la nostra màquina. Ens apareix degut a que l'Eclipse dóna la possibilitat de fer servir un JDK concret per a cada projecte (sempre que els tinguem instal·lats). De totes maneres, nosaltres de moment, sempre farem servir el que l'Eclipse agafa per defecte, que és el que es correspon a la variable del sistema JAVA_HOME.

2. Anem a crear la nostra primera classe. Al 'package explorer', fem click amb el botó dret sobre 'ProvaNIF'. Triem new --> class i ens apareix una finestra a partir de la qual podrem crear la nostra classe.

   De les coses que podem editar/modificar en la finestra, únicament posarem el nom de la classe (DNI); amb lo qual, la finestra de creació de classes ens quedarà com es veu a continuació:



Fem click sobre 'finish' i ja haurem creat la classe. A continuació, se us obrirà una finestra en el pannell central on podreu editar el fitxer DNI.java, afegint mètodes a la classe DNI (veure següent gràfic).



3. Un cop fet això, afegim els mètodes de la classe DNI. Per a fer-ho, podeu agafar el text del següent requadre de codi. Podeu copiar el text i enganxar-lo directament a l'Eclipse si voleu.

```
public class DNI {

  private String dni;

  /**
   * Constructor dni
   */

  public DNI() {
  }


  /**
   * Mètode accessor que retorna el DNI com una cadena de caràcters.
   * @return java.lang.String, retorna DNI
   */
```

```
  public String getDNI() {
    return (dni);
  }


  /**
   * Mètode accessor que modifica el valor del DNI
   * @param valor del nou dni
   */

  public void setDNI(String dni) {
    this.dni = dni;
  }


  /**
   * Mètode que retorna la representació en forma de string del DNI.
   * En aquest cas, la representació en forma de string coincideix
   * amb l'atribut dni.
   * @return a string DNI
   */

  public String toString() {
    return (getDNI());
  }
}
```

4. Un cop fet això, graveu el fitxer DNI.java a disc (icona situada just sota els menús 'Edit' i 'Source'. Per a fer-ho cal que tingueu activa la finestra de text corresponent a DNI.java.

   Aquesta operació també compilarà automàticament el fitxer DNI.java. Això no us hauria de donar cap error (si n'hi hagués, apareixerien al pannell 'Tasks').

   Per altra banda, proveu d'expandir (o bé al 'package explorer' o bé al 'outline') la classe DNI. Podreu comprovar com ara apareixen tots els mètodes que hem enganxat. Quan seleccionem algun d'ells, la finestra associada a DNI.java s'actualitza de manera que es mostra la part del fitxer DNI.java corresponent a aquest mètode.

5. Passem a crear ara la classe NIF. Realitzem la mateixa operació que abans: sobre el 'package explorer', seleccionem la classe DNI i fem click amb el botó dret del ratolí, seleccionant new --> class. Ara posem NIF com a nom de classe.

   Noteu que al haver fet l'operació seleccionant sobre la classe DNI, ja ens posa que el nom de la superclasse és DNI. Si no fos així, ho hauríem de fer nosaltres a mà.

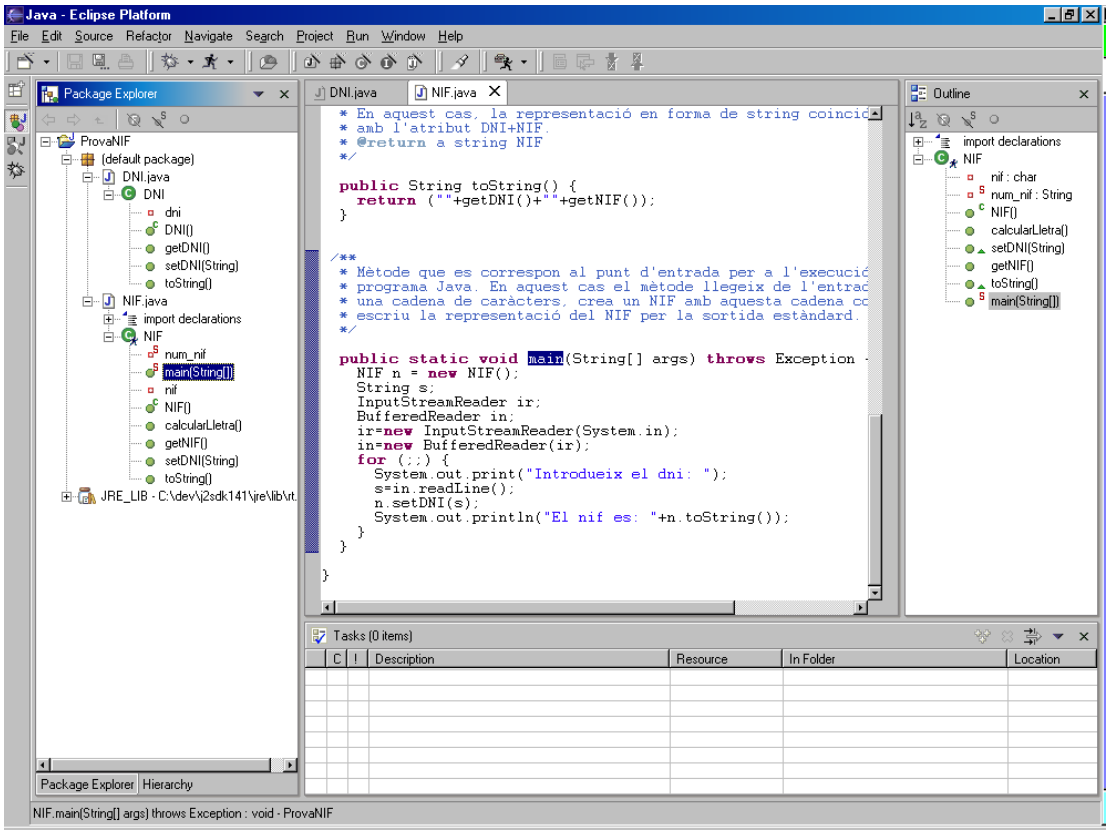
   Aquesta classe tindrà un constructor equivalent al de DNI; i també tindrà mètode main. L'Eclipse ens dóna la possibilitat de crear aquests mètodes de manera automàtica. Per a fer-ho, al darrer apartat de la finestra de creació de classes seleccionem 'public static void main...' i també 'constructors from superclass'. Acte seguit fem click sobre 'finish'.

6. Amb això, haurem creat la classe NIF. Per a crear els mètodes d'aquesta classe podeu copiar el codi del següent requadre i enganxar-lo a la finestra NIF.java adequadament (no us oblideu l'import del package java.io inicial).

```
import java.io.*;

public class NIF extends DNI {

 private char nif;
 private static String num_nif;

 /**
  * Constructor NIF
  */

 public NIF() {
   num_nif = new String("TRWAGMYFPDXBNJZSQVHLCKET");
 }


 /**
  * Calcula la lletra corresponent al DNI guardat, i modifica l'atribut lletra
  * convenientment.
  */

 public void calcularLletra() {
   Integer i = new Integer(getDNI());
   int valor = (i.intValue()/23);
   int valor2 = valor*23;
   int valor3 = i.intValue()-valor2;
   nif = num_nif.charAt(valor3);
 }


 /**
  * Redefinició del mètode de forma que cada cop que es modifiqui el valor
  * del DNI es calculi de manera automàtica la lletra del NIF.
  *
  * @param valor del nou dni
  */

 public void setDNI(String dni) {
   super.setDNI(dni);
   calcularLletra();
 }


 public char getNIF() {
   return nif;
 }


/**
 * Mètode que retorna la representació en forma de string del NIF.
 * En aquest cas, la representació en forma de string coincideix
 * amb l'atribut DNI+NIF.
 * @return a string NIF
```

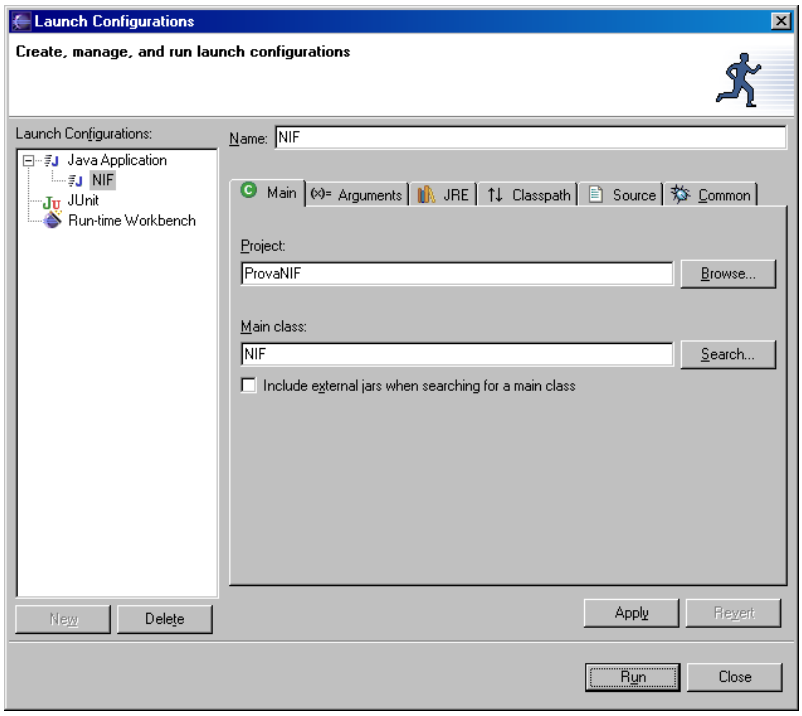
```
 */

 public String toString() {
   return (""+getDNI()+""+getNIF());
 }


/**
 * Mètode que es correspon al punt d'entrada per a l'execució d'un
 * programa Java. En aquest cas el mètode llegeix de l'entrada estàndard
 * una cadena de caràcters, crea un NIF amb aquesta cadena com a DNI, i
 * escriu la representació del NIF per la sortida estàndard.
 */

 public static void main(String[] args) throws Exception {
   NIF n = new NIF();
   String s;
   InputStreamReader ir;
   BufferedReader in;
   ir=new InputStreamReader(System.in);
   in=new BufferedReader(ir);
   System.out.print("Introdueix el dni (enter per acabar): ");
   s=in.readLine();
   while (s.length()>0) {
     n.setDNI(s);
     System.out.println("El nif es: "+n.toString());
     System.out.print("Introdueix el dni (enter per acabar): ");
     s=in.readLine();
   }
 }
}
```

No oblideu a gravar el fitxer a disc. Igual que passava amb la classe DNI, ara tenim accés als diferents mètodes de la classe NIF en el 'package explorer'. Després de fer tot això, la finestra de l'Eclipse que tindreu serà similar a aquesta:



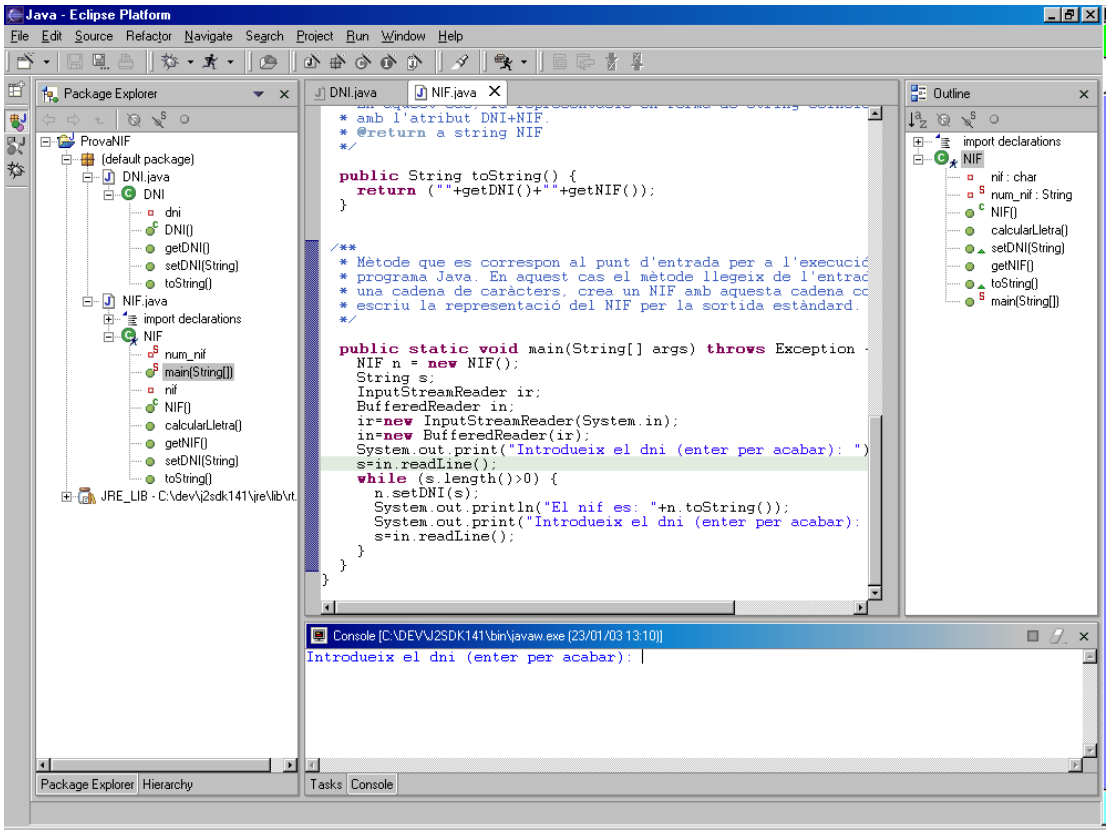
7. Podeu obrir la perspectiva ‘Java browsing’, on podreu navegar per les classes i mètodes del projecte d’una manera lleugerament diferent a com ho feu en la perspectiva ‘Java’.

8. Anem a executar el nostre programa. Per això, seleccionem del menú ‘Run’ l’opció ‘Run...’. Ens apareix una finestra on seleccionarem ‘Java Application’ i farem click sobre el botó ‘new’. Ens apareixerà la següent finestra:



9. Haurem de seleccionar el nom de projecte (ProvaNIF) i el nom de la classe on està el mètode main que volem executar com a programa principal (seleccionarem NIF). Donat que tenim un sol projecte, és possible que aquests valors ja vinguin establerts per defecte; però no sempre serà així.

10. Fem click sobre ‘Run’. Al pannell ‘tasks’ apareix la consola:



A la consola hi apareix 'Introdueix el DNI (enter per acabar): '. Cal que introduïm un DNI; i el nostre programa calcularà el NIF corresponent. El programa ens demanarà repetidament un nou DNI fins que premem únicament enter; moment en el que acabarà.

3. IMPORTACIÓ D'UN PROJECTE JA EXISTENT

Un cop ja hem creat un projecte java, anem a veure un segon projecte una mica més complex. Per exemple, la pràctica de FP2 del semestre de tardor de 2002. Aquest cop però, partirem d'uns fitxers Java que ja tindrem en el nostre disc.

Per a crear el projecte i importar els fitxers, seguirem els següents passos:

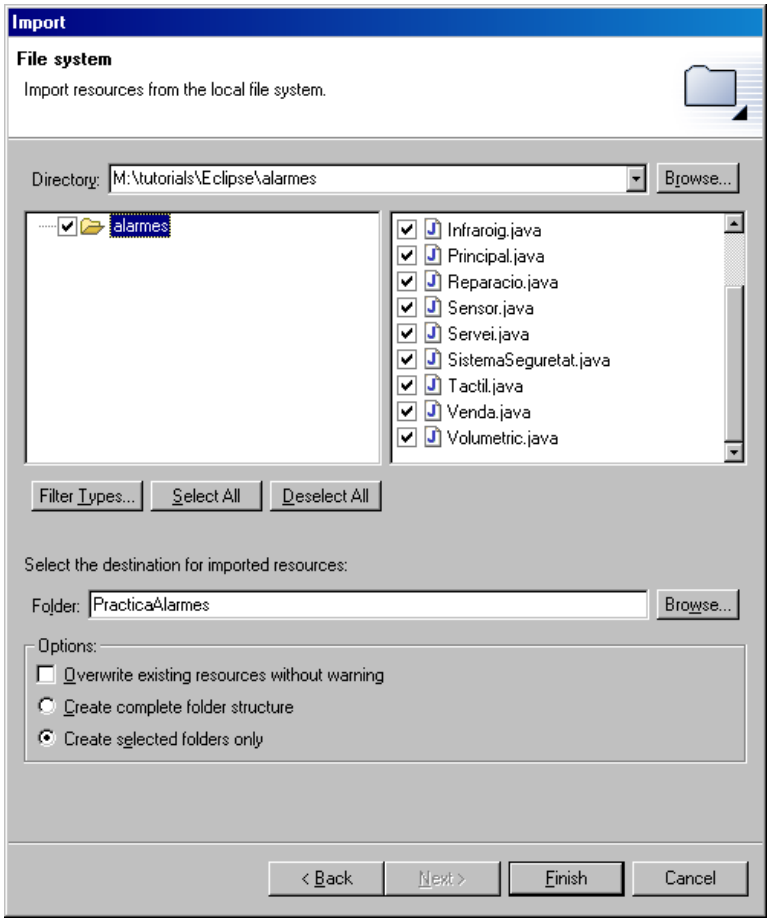
1. Tanquem els fitxers DNI.java i NIF.java, que es corresponien al primer projecte que hem creat. Col·lapsem el projecte ProvaNIF (fent click al '-' que hi ha davant de 'ProvaNIF'). En cap cas eliminarem el projecte ProvaNIF, ja que això eliminaria qualsevol rastre dels fitxers .java que hem creat.

2. Igual que hem fet amb el projecte del NIF, creem un nou projecte Java amb nom: PracticaAlarmes.

3. Un cop creat, apareix un nou projecte anomenat ‘PracticaAlarmes’ en el ‘package explorer’. Fem click amb el botó dret del ratolí sobre ell. I seleccionem ‘Import...’. S’obre una finestra i seleccionem ‘file system’ i ‘Next’.

4. Un cop aquí, ens apareix una finestra on hem de triar en primera instància el directori on tenim els fitxers .java de la pràctica de les alarmes (ho podem fer fent click sobre el botó ‘Browse’ associat a l’apartat ‘directory’).

   Seleccionem tots els fitxers .java, tal com apareix a la següent il·lustració; i fem click sobre ‘finish’.



5. Ara podem comprovar com amb un projecte d’aquestes dimensions un IDE ens pot ajudar força (imagineu-vos doncs, amb un projecte de mides reals!). Podeu navegar per les classes, mètodes, veure ràpidament els atributs, etc.

   Per exemple, podeu fer click amb el botó dret sobre la classe Sensor i triar ‘open type hierarchy’. Això us obre una nova finestra on es visualitza la jerarquia de sensors que trobem a la pràctica.

L'Eclipse disposa de multitud d'opcions que són una bona ajuda per als desenvolupadors. No és objecte d'aquest tutorial presentar totes aquestes funcions. Però si us recomanem efusivament que doneu una ullada a la documentació/ajuda del sistema per a fer-vos una idea de les facilitats aportades per l'IDE. Per a fer-ho, aneu a help --> help contents. I dins de l'ajuda únicament cal que mireu els punts 'Workbench user guide' i 'Java development user guide'. Això sí, la documentació és en anglès.

6. Per a executar la pràctica de les alarmes farem igual que amb la prova que hem fet anteriorment del NIF. Aquest cop però, ho farem pas a pas. Per això, triarem del menú 'run' l'opció 'debug...'.

   A la finestra de preparació de l'execució que ens surt, i on ens apareixerà la corresponent al NIF, seleccionem 'Java Application' amb el botó dret, i després 'new'. Llavors, a 'name' posem PracticaAlarmes, a project repetim el mateix, i a main class posem fase4 (probablement alguns d'aquests noms ja apareixeran automàticament).

   El programa de proves de la pràctica de les alarmes llegeix fitxers de text del disc (sensors.txt, clients.txt, serveis.txt i sistemes.txt). Per això, li hem de dir que s'executi en el directori on estan aquests fitxers (el directori original de la pràctica).

   Per a fer-ho, anem a la pestanya 'Arguments' i deseleccionem 'use default working directory'. Llavors triem 'local directory' i hi posem el directori on tenim aquests fitxers.

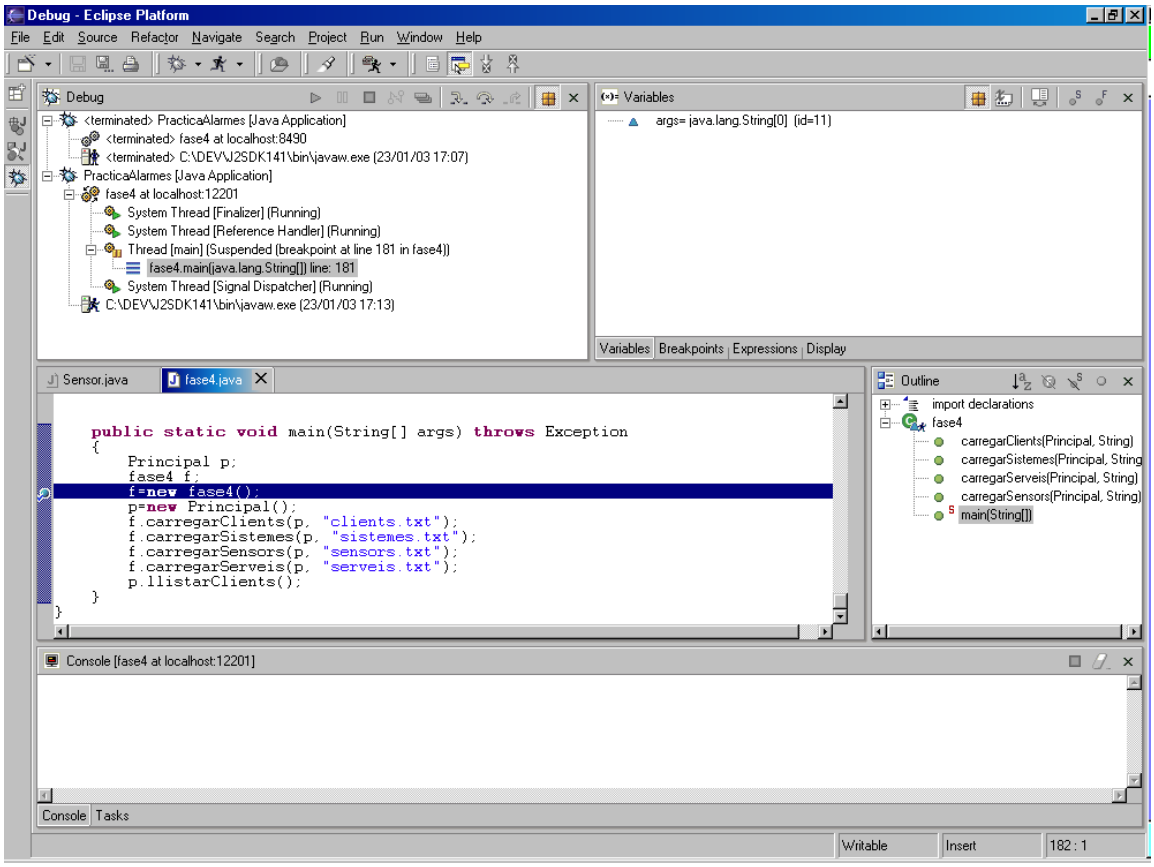
   Un cop fet això seleccionem 'Apply' i després 'Debug'. Podreu comprovar (a la consola) que la pràctica s'ha executat i ha generat una sortida.

Si volguéssim depurar l'aplicació i anar-la executant pas a pas, ho podríem fer així:

7. Aneu a la classe fase4, mètode main. Feu doble click sobre ell. S'obrirà el fitxer fase4.java i es visualitzarà el mètode en qüestió.

8. Situeu-vos a la línia 'f=new fase4();', a l'esquerra de tot (de fet fora del text). Feu click amb el botó de la dreta i apareix un menú. Seleccioneu 'add breakpoint'. Assegureu-vos que sou fora de l'àrea de text (si sou a l'àrea de text apareix un menú diferent).

9. Tornem a executar l'aplicació (ha de ser en mode debug). Ara podem fer: run --> debug history --> PracticaAlarmes. Se'ns obrirà la perspectiva 'debug'; obtenint una finestra semblant a la que trobeu a continuació:



A la part superior esquerra hi teniu tots els processos: els que s'estan executant i també els que ja han acabat. Pels que s'estan executant hi teniu també la traça de les crides entre funcions que hi ha en el moment d'execució actual. A la figura superior podem comprovar com només tenim un nivell corresponent al mètode *fase4.main*.

A la part superior dreta veiem les variables que existeixen en l'entorn actual. En aquest moment únicament tenim 'args'; però si fem F6 (executar següent instrucció sense entrar-hi), apareixerà 'f'.

A la part central tenim el codi que estem executant, i a la seva dreta una descripció rellevant a la classe (o altre objecte) que centra la nostra atenció (en aquest cas fase4).

Per últim, a la part inferior, tenim la consola, on encara no s'hi ha escrit res.

Podeu practicar amb F6 i F5 (si voleu entrar en l'execució d'una crida); i anar comprovant com es va executant l'aplicació pas a pas. I el que és més important: com la podeu anar seguint detingudament.

Un bon ús del depurador (debugger) us permetrà estalviar-vos molt de temps per a detectar i corregir errors, un cop ja hagueu codificat els vostres algorismes.

4. CORRECCIÓ I DEPURACIÓ D'ERRADES

Evidentment quan codifiqueu us trobareu sovint en que cometeu errades. Aquestes errades poden ser tant a nivell sintàctic com a nivell semàntic. Les errades a nivell sintàctic es corresponen a trossos de codi que no segueixen les normes rígides del llenguatge Java. Aquestes errades són més fàcils de detectar. Les errades a nivell semàntic es corresponen a trossos de codi que quan s'executen no fan exactament el que nosaltres esperàvem. Aquestes poden ser força més difícils de detectar.

Un IDE ens pot ser força útil de cara a trobar ràpidament tant les errades sintàctiques com les errades semàntiques. En aquesta secció veurem un exemple de codi amb errors i com l'Eclipse ens pot ajudar a corregir aquestes errades.

Anem a fer-ho pas a pas:

1. Aneu a la perspectiva Java, col·lapseu tots els projectes que tingueu expandits i tanqueu totes les finestres de text que tingueu obertes.

2. Creeu un projecte anomenat 'ProvaErrades' i creeu una classe anomenada tdoble2. Copieu el codi del següent bloc de text i enganxeu-lo a la classe. Un cop fet això, graveu el fitxer tdoble2.java a disc.

```
/** Aquesta classe calcula el doble del nombre entrat */
public class tdoble2 {
  /** Mètode utilitzat per la màquina virtual per executar la classe*/
    public static void main(String[] argv) {
        // Definició de les varibles.
        int nombre_1;
        int resultat;
        string missatge;

        System.out.println("Càlcul del doble d'un nombre entrat");
        if (argv.length=<10)||(argv.length<1) {
            System.out.println("Error: Has d'entrar un mínim d'un nombre i com a molt
10");
    } else {
            for(i=0;i<=argv.length;i++)
            {
                nombre_1=Integer.parseInt(argv[i]);
                resultat=nombre_1*2;
                missatge="El doble de "+" "+nombre_1+" "+"és"+" "+resultat;
                System.out.println(missatge);
            }
        }
    }
}
```

3. Un cop fet això, l'Eclipse marca amb una senyal vermella la línia corresponent a l'if. En aquesta línia, el símbol '=<' apareix subratllat en vermell. Això vol dir que el compilador ha trobat una errada sintàctica.

Si anem al pannell de tasques (si estéssim veient la consola hauríem de canviar a ‘tasks’), també podrem comprovar com ens marca l’errada.

Evidentment, el símbol ‘=<’ ha de ser en realitat ‘<=’. Fem el canvi i gravem a disc.

4. Segueix havent un error a la mateixa línia, però ara el que hi ha subratllat és l’operador ‘||’. Clar! el que passa és que tota l’expressió del if ha d’anar entre parèntesi. La posem tota ella entre parèntesi i tornem a gravar a disc.

5. Ara ens apareixen tres llumetes grogues. Aquestes llumetes ens indiquen també errors sintàctics que l’IDE considera d’una altra categoria; però a la fi... tot són errades. Anem a veure que passa:

   Sempre fem les correccions començant per dalt, doncs la correcció d’una errada anterior podria corregir una errada posterior.

   A la línia de la primera llumeta hi ha subratllada la paraula ‘string’; i al pannell de ‘tasks’ ens diu que no coneix el símbol string (no el pot resoldre, traduint de l’anglès).

   Evidentment! és String, començant amb majúscules. Fem el canvi i tornem a gravar.

6. Anem a la següent llumeta: fa referència a la línia del for. Mirant el pannell de ‘tasks’ veiem que no coneix i. Es clar! no està definida. Doncs afegim una línia al principi del mètode main que digui:

   ```
   int i;
   ```

   I gravem.

Ara ja no queda cap errada sintàctica, i podem executar l’aplicació.

7. Anem a run --> debug... i triem ‘Java application’ --> new, i omplim adequadament si cal.

8. L’algorisme calcula el doble dels nombres entrats com a paràmetres a la línia de comandes. Podem modificar aquests paràmetres anant a la pestanya ‘Arguments’. Modifiquem els ‘program arguments’ posant entre 1 i 10 nombres enters separats per espais.

   Per exemple, hi podem posar: 4 10 23

9. Fem click sobre debug. L’aplicació s’executa, i ens diu:

   ```
   Càlcul del doble d'un nombre entrat
   Error: Has d'entrar un mínim d'un nombre i com a molt 10
   ```

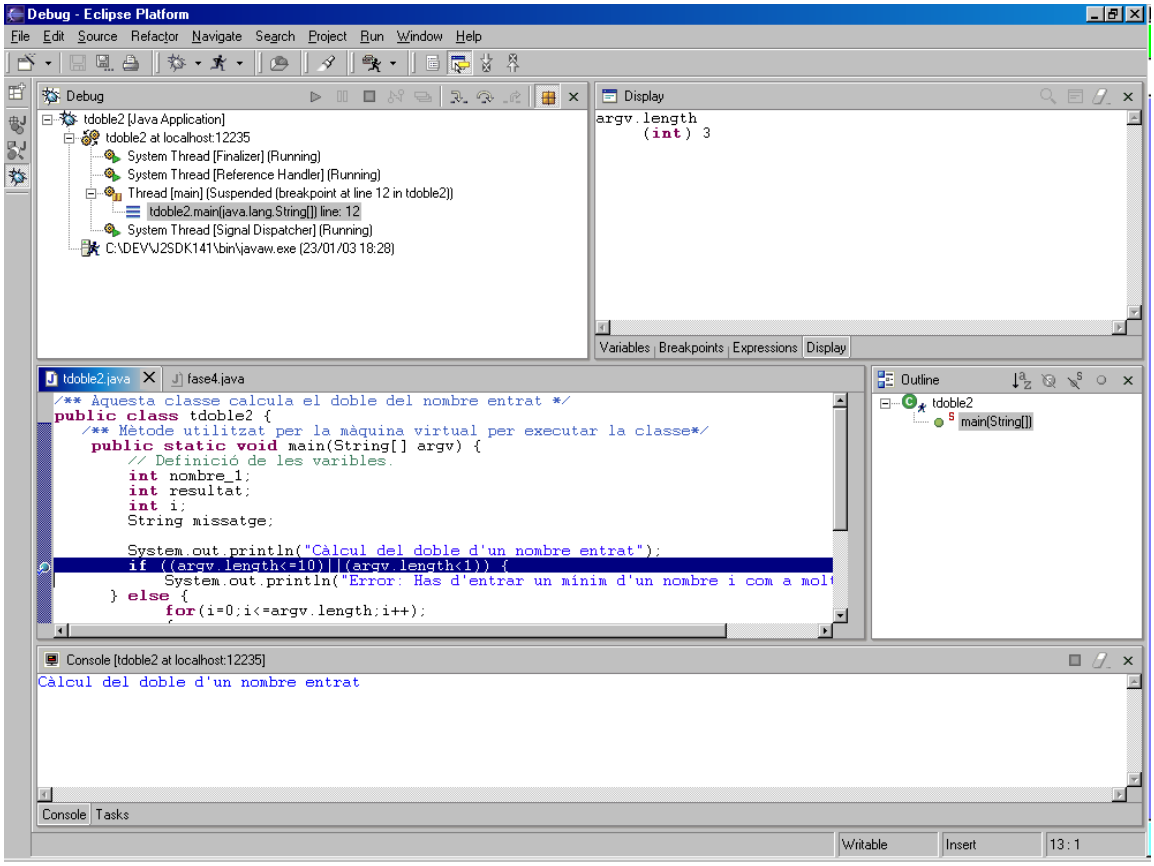
Això no ens acaba d'agradar, doncs nosaltres hi hem posat 3 nombres, cosa que entra dins del que el programa contempla.

10. Així doncs, posem un breakpoint a la línia del if i tornem a executar (no cal tornar a introduir els nombres).

    S'obre de nou la perspectiva 'debug', amb la línia del 'if' marcada en blau, indicant que l'execució està parada en aquest punt esperant instruccions nostres.

11. Anem a veure quin valor té argv.length. Per això, seleccionem de la mateixa línia del 'if' la subexpressió *argv.length* i fem click amb el botó de la dreta i seleccionem 'Display'.

    A la part superior dreta apareix l'expressió i el seu valor (tal i com veiem a la següent figura).



    Veiem que el seu valor és realment 3; cosa que ens fa pensar que l'expressió del if és incorrecta. Efectivament, hauria de ser: *((argv.length>10)||(argv.length<1))*.

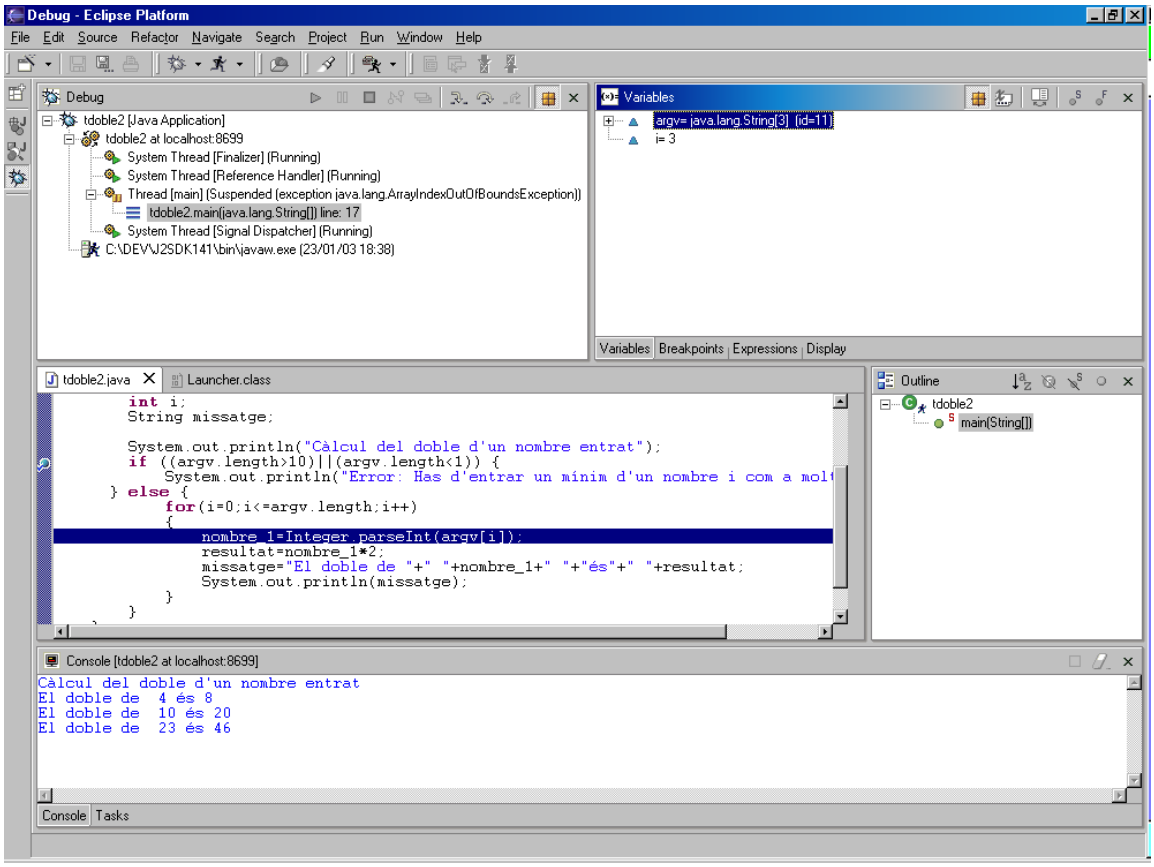
12. Fem el canvi, gravem el fitxer i aturem l'execució. Per a aturar l'execució fem click amb el botó dret sobre el procés que s'està executant (part superior esquerra de la finestra) i seleccionem 'terminate and remove'.

13. Acte seguit tornem a executar el programa en mode ‘Debug’. Efectivament, provant-ho pas a pas podem comprovar com ara l’expressió és correcta i no surt el missatge d’error per la consola. Podem fer run --> resume per executar l’aplicació de cop.

14. A la consola obtenim el següent:

    Càlcul del doble d'un nombre entrat
    El doble de 4 és 8
    El doble de 10 és 20
    El doble de 23 és 46

    L’execució però, s’atura a la línia 17. I si ens fixem en el pannell de debug, ens informa que el thread està suspès per què es produeix una excepció de tipus ArrayOutOfIndexException (veure següent figura).



Aquest comportament de l’Eclipse és molt útil donat que ens informa d’això i a la vegada congela l’execució de cara a que puguem examinar quin és l’estat que ha dut a generar l’excepció (podem examinar variables, avaluar expressions...). Si ara tornéssim a fer ‘resume’, es generaria l’excepció (a la consola) i l’execució s’aturaria.

Si voleu ho podeu provar i tornar a reproduir l’excepció de nou.

Anem al pannell de variables, seleccionem visualitzar les variables (teníem les expressions) i veiem com el valor de i és 3 (tal i com apareix a l'anterior figura).

Efectivament, argv és una taula de 3 posicions (de 0 a 2) i estem intentant accedir a la posició 3. Això ens indica que hem de canviar la condició del for.

Ho fem de la següent manera:

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

15. Gravem, tornem a executar, i.... finalment l'execució funciona i acaba amb normalitat!

5. ÚS DE LLIBRERIES EXTERNES

Qualsevol projecte creat amb l'Eclipse, per defecte, pot fer ús de les classes que el mateix JDK porta incorporades (java.util.Vector...). Ara bé, molts cops voldrem usar altres classes que proporcionin funcionalitats diferents a les proporcionades per les classes del propi JDK; i que no formaran part del nostre projecte.

Per exemple, si estem desenvolupant una aplicació que visualitza gràfics en 3 dimensions és possible que vulguem usar una llibreria de classes específica per a la visualització d'aquest tipus de gràfics, com per exemple la llibreria OpenGL per a Java.

5.1. ÚS DE FITXERS .JAR

Aquestes llibreries normalment acostumen a venir en un (o diversos) fitxer amb extensió .JAR que conté les classes de la llibreria. Els fitxers JAR venen a ser com els fitxers ZIP: serveixen per a empaquetar i comprimir d'altres fitxers. Així com els fitxers .ZIP acostumen a ser d'ús general, guardant-hi qualsevol tipus de fitxers a l'interior, els fitxers JAR normalment s'usen per a agrupar conjunts de classes Java.

Un fitxer JAR es pot crear, examinar i modificar amb l'eina 'jar' que ve amb el matex JDK (feu 'jar –help' per a veure com executar aquesta eina; o també podeu accedir a la documentació mateixa del JDK).

Per a poder usar desde un projecte Eclipse les classes contingudes en un fitxer .jar hem de fer el següent:

1. Anar a les propietats del projecte desde el qual volem fer ús del .jar (fent click amb el botó dret sobre el nom del projecte a la perspectiva Java).
2. Seleccionar 'java build path' i un cop fet això, triar la pestanya 'libraries'.
3. Fer click sobre 'add external JARs...'.
4. Seleccionar els arxius JAR que volem usar.

Un cop fet això ja podrem usar les classes contingudes en els .JAR desde les classes del projecte sense cap problema.

5.2. ACCÉS AL CODI FONT DE LLIBRERIES

Molts cops, les llibreries acostumen a venir també amb el codi font (els fitxers .java corresponents a les classes de la llibreria). Algunes vegades ens pot interessar accedir a aquest codi font igual com fem amb el nostre codi font. Això ens serà útil únicament en alguns casos concrets, especialment quan estem en fase de depuració d'un programa que fa ús de la llibreria externa.

Una mètode bastant còmode amb el que podem tenir accés al codi font d'una llibreria externa és el següent. És important remarcar que aquest mètode substitueix l'explicat a l'apartat 5.1. Per tant, si realitzem el que aquí s'explica, no s'ha d'afegir el .JAR corresponent a la llibreria com a JAR extern al projecte.

Cal dir també que és imprescindible de disposar dels fitxers .java corresponents a la llibreria. Si no és així, no podem fer res.

En cas que disposem dels fitxers .java, podem actuar com s'explica a continuació:

1. Creem un nou projecte a l'Eclipse, amb el nom de la llibreria (per exemple 'OpenGL').
2. Importem tots els fitxers .java corresponents a la llibreria. Podem fer aquests dos passos de la mateixa forma que està explicat al punt 3 d'aquest document per a la pràctica de les alarmes de l'assignatura FP2.
3. Anar al projecte on volem fer servir la llibreria externa.
4. Accedir a les seves propietats.
5. Fer click sobre 'java build path', i després sobre la pestanya 'Projects'.
6. Seleccionar el projecte corresponent a la llibreria externa, i que hem creat en els passos 1 i 2.

Desde aquest moment ja podrem accedir a la llibreria externa desde el nostre projecte Eclipse, i tindrem accés també al seu codi font.

Aquesta opció té l'avantatge de poder fer servir les funcionalitats que proporciona l'Eclipse (nevagador de classes, mètodes...) per a familiaritzar-vos amb l'estructura de les classes de la llibreria.