Institut Galilée PDJ
Master 1

T.P. 1

Java - Eclipse / Netbeans

Java Environments

Several environments exist for developing Java programs. It is critical to use a “good” version : the same one for compiling and executing a program (essentially when classes are dynamically loaded). The available java at Institut Galilée is the following one :

— java 1.8.0_181 (`/usr/bin/java`) from SUN/Oracle Java SE.

There exists also a GNU open source in some PC, not fully compliant with the Oracle/Sun Java.

The current Oracle/Sun Java version is 13.0.1

Terminology

— Java SE (Standard Edition) : basic API (lang, collections, ..., jar, ...) without virtual machine.
— JRE (Java Runtime Environment) : JSE + virtual machine + plugin.
— JDK (Java Development Toolkit) : JRE + softwares (java, javac, jar, ...).

1 Eclipse Environment

Eclipse is an integrated development environment (*IDE*), free, extensible and multi-language, able to plug tools for any programming language. Eclipse is mainly developed in Java (with the graphic library SWT from IBM). The IDE Eclipse is developed by means of plug-in (compliant with the OSGi norm). Other commercial softwares are developed on this free software : IBM Lotus Notes 8, IBM Symphony or WebSphere Studio Application Developer.

Download, plug-in, tutorials : `www.eclipse.org`

1.1 Initialization

Modify your file `.bashrc` such that Eclipse is in your path and verify `/LOCAL/eclipse/jee-oxygen` is in your variable `$PATH`.
Remark : the Eclipse version ‘oxygen’ is available (version 4.7.1, the most recent version is 4.13).

1.2 Project

Creation of a project `File -> New -> Java Project`
`Project Name` : write `Thread_Share`

A window `Project Explorer` allows you to travel through projects and source files (window accessible also with `Window -> Show view -> Project Explorer`).

Access to files A Workspace contains the whole set of projects and files. The workspace is a folder in your home directory, defined when launching Eclipse. You can retrieve the path by `File -> Switch Workspace`. Each project is a sub-directory containing a folder `bin` for the executable files, and a folder `src` for source files.

Creation of source files, execution Select `Thread_Share/src` in `Project Explorer`, then right button and `New -> Class`. Enter the name of the class, e.g. `TestRunnable`, click on `main` (as this class contains the main method). Write and save the code.

Execution with `Run`. `Console` is the output window.

2 Netbeans Environment

NetBeans is also an open source IDE for Java, developed by Sun. It can also support other languages : Python, C, C++, XML, HTML. Furthermore, it allows to :

— create and deploy complete environments (web)
— develop J2EE projects
— support databases plug-in (the open source data base GlassFish is also available)
— deploy a program on a cluster of hosts

2.1 Initialization

Modify your file `.bashrc` to allow Netbeans. Verify that `/LOCAL/netbeans/bin` is in your variable `$PATH`.
Remark : the version installed is the 8.0.2 (`/LOCAL/netbeans/bin/netbeans`).

2.2 Project and Group

A group is a set of projects that may be created with `File->Project Groups -> New Group`. Give a name, e.g. `TP1`.

Creation of a project `File -> New Project -> Java -> Java Application`

Access to files A Workspace contains the whole set of projects and files. The workspace is a folder in your home directory, defined when launching Eclipse. You can retrieve the path by looking at properties (right button).

3 Exercice : Class and inheritance

For the following questions, think of what should be the output, then program and run the codes and test your answer.

3.1 Values, Overloading

```
class A {
   private int val=0;
   public static void affichePlus (int a) {
      a++;
      System.out.println (a);
   }
   public static void affichePlus (A a) {
      a.val++;
      System.out.println (a.val);
   }
   public static void main (String[] args) {
      A unObj = new A ();
      A unAutreObj = new A();
      affichePlus (unObj.val);
      affichePlus (unObj.val);
      affichePlus (unObj);
      affichePlus (unObj);
      affichePlus (unAutreObj);
      affichePlus (unAutreObj);
      if (unObj == unAutreObj)
         System.out.println("Egales");
      else
         System.out.println("Differentes");
   }
}
```

3.2 Dynamic Linking

```
class A {
   public String f(B obj) { return ("A␣et␣B");}
   public String f(A obj) { return ("A␣et␣A");}
}

class B extends A {
   public String f(B obj) { return ("B␣et␣B");}
   public String f(A obj) { return ("B␣et␣A");}
}

class Test {
   public static void main (String [] args){
      A a1 = new A();
      A a2 = new B();
      B b = new B();
      System.out.println(a1.f(a1));
      System.out.println(a1.f(a2));
      System.out.println(a2.f(a1));
      System.out.println(a2.f(a2));
      System.out.println(a2.f(b));
      System.out.println(b.f(a2));
   }
}
```

3.3 Polymorphism

```
class A {
   public String f(D obj) { return ("A␣et␣D");}
   public String f(A obj) { return ("A␣et␣A");}
}

class B extends A {
   public String f(B obj) { return ("B␣et␣B");}
   public String f(A obj) { return ("B␣et␣A");}
}

class C extends B{
}

class D extends B{
}

class Test {
   public static void main (String [] args){
          A a1 = new A();
          A a2 = new B();
          B b = new B();
          C c = new C();
          D d = new D();
          System.out.println(a1.f(b));
          System.out.println(a1.f(c));
          System.out.println(a1.f(d));
          System.out.println(a2.f(b));
          System.out.println(a2.f(c));
```

```
            System.out.println(a2.f(d));
            System.out.println(b.f(b));
            System.out.println(b.f(c));
            System.out.println(b.f(d));
    }
}
```

3.4 Fields

```
class A {
    int i;
    int f() { return i; }
    static String g() { return ("A"); }
    String h() { return g(); }
}

class B extends A {
    int i=2;
    int f() { return -i; }
    static String g() { return ("B"); }
    String h() { return g(); }
}

class Test {
    public static void main(String [] args) {
        B b = new B();
        System.out.println(b.i);
        System.out.println(b.f());
        System.out.println(b.g());
        System.out.println(b.h());
        A a = b;
        System.out.println(a.i);
        System.out.println(a.f());
        System.out.println(a.g());
        System.out.println(a.h());
    }
}
```

4 Polluter Robots

A *world* is represented by a 2D matrix of boxes. Different kind of robots move in a world : polluters, cleaners.

- Polluters walk in a world and let greasy papers on boxes. There are two kinds of polluters : jumpers and straighters (these ones go straight). Each polluter leaves a greasy paper on each box where they stand.
- Cleaners remove greasy papers if there is one in the box where they stand. They walk in a "boustrophédon" style, i.e. they go through a whole line from right to left, then through a second line from left to right, and so on. There is one specific subclass of cleaners : distracted cleaners that remove only one paper out of two.

1. Define the class `World` with the following variables, constructors and methods :
   - number of lines `nbL`, number of columns `nbC`, a boolean matrix `mat` with `nbL` lines and `nbC` columns (`true` if there is a greasy paper).
   - A constructor with parameters and another one without `World()` : this constructor creates a 10*10 world without greasy papers.
   - `public String toString()` : returns a string describing the world : if the box is empty print "." (a dot), otherwise print "o".
   - `putsGreasyPaper(int x,int y)` : puts a greasy paper on the box `(x,y)`.
   - `removesGreasyPaper(int x,int y)` : removes the greasy paper from the box `(x,y)`.

— `isDirty(int x, int y)` : tests if the box `(x, y)` contains a greasy paper.
— `nbGreasyPapers()` : returns the number of greasy papers present in the world.

2. Define a class `TestRobot` with a method `main` for creating a 10*10 world. Test methods `putsGreasyPaper`, `removesGreasyPapers` and `isDirty`.
3. Define an abstract Class `Robot` that contains the following fields and methods :
   — `posx, posy` : position of the robot on the world.
   — `m` : variable of type world
   — two constructors : `Robot(int x,int y,World m)` that creates a robot at position (x,y) and `Robot(World m)` that creates a robot at a random position (use `Math.random()`). The constructor `Robot(World m)` should call the other constructor.
   — `movesTo(int x, int y)` : to move to position `(x, y)`.
   — `visit()` : abstract method (defined in sub-classes).
4. Define an abstract class `Polluter` that contains one method :
   — `pollute()` : puts a greasy paper in the box where the robot is.
   — constructor(s).
5. Define the class `Straighter` that contains the following fields and methods :
   — `startingColumn`, column that will be visited.
   — constructor(s).
   — `visit()` : instanciation of the abstract method of the class `Robot` : from (0, startingColumn) to this end of this column one box by one box, then stops. A greasy paper is put in each visited box.
6. Define the class `Jumper` with the following fields and methods :
   — `deltax` : the number of boxes the jumper passes through.
   — constructors.
   — `visit()` instanciation of the abstract method of the class `Robot` : from (0,0), then (1, deltax), then (2,2*deltax), then (3,3*deltax) (everything modulo nbC). The robot stops on the last line. A greasy paper is put in each visited box.
7. Define the class `Cleaner` with the following methods :
   — `clean()` : removes the greasy paper at the box where the robot is.
   — constructors,
   — `visit()` : instanciation of the abstract method of the class `Robot` : From (0,0) straight to the end of the line, then next line in the other direction, and so on until the grid is fully visited. Greasy papers are removed as the robot visits the boxes.
8. Define the class `DistractedCleaner` with the following methods :
   — constructors.
   — `visit()` : redefinition of the method defined in `Cleaner` : same style of visit but removes only one paper out of two.
9. Modify the class `TestRobot` to test the previous classes and methods.