Serie 1

Content:
– Classes et ADT
– Programmation orientée objets en Java - types statique et dynamique
– Java : héritage - polymorphisme - interfaces - ...

(1) ADT and classes : The Dispenser example

Given the small test program implementing an ADT STACK[G] as seen during the lecture. It contains the following classes and interfaces in the `dispenser` package :
– `Dispenser.java` interface.
– `DispenserFactory.java` classe.
– `DispenserTest.java` main class.
– `ArrayStack.java` class in the sub-package `stack`.

1. Create a NetBeans project with the provided classes [2].
2. Test the program and get familiar with it.
3. In the `stack` package add at least one other implementation, by extending the `Dispenser` interface, simulating a stack (LIFO).
4. In the `queue` package add at least one other implementation, by extending the `Dispenser` interface, simulating a queue (FIFO).
5. Modify the `DispenserTest` and the `DispenserFactory` classes accordingly.
6. Improve the whole application with feature you judge necessary.

(2) Static and dynamic types

Given the classes `A`, `B`, `C` and `Start`, where `Start` is the main class of the system :

```
1   public class Start {
2
3       A f1;
4       B f2;
5       C f3;
6
7       public Start() {
8           f1 = new A();
9           f2 = new B();
10          f3 = new C();
11          f1.message();
12          f1=f2;
13          f1.message();
14          f1=f3;
15          f1.message();
16          f2.message_from_a();
17          f3.message();}
18
19      public static void main(String args[]) {
20          new Start();}
21  }
22
23  class A {
24      void message() {
25          System.out.print("A ");}
26   }
27
28  class B extends A {
29      void message() {
30          System.out.print("B ");}
```

```
31          void message_from_a() {
32              super.message();}
33      }
34
35      class C extends A {
36          void message() {
37              System.out.print("C ");}
38      }
```

Listing 1 – Source code of `A`, `B`, `C` and `Start`

What text is written to the console if the application is launched ? Explain your answer in terms of “static type” and “dynamic type” (or “runtime type”).

(3) Shapes – Part I

We would like to write a program allowing us to create some simple geometric shapes. The program should allow us to draw the shapes, show the list of all shapes, compute the sum of areas and the sum of circumference of all geometric shapes.

A typical use-case was shown during the lecture and the sample code is provided online [2] (file Shapes_Solution.zip).

After important this program into your IDE you will find the following additional functionality :
– Each geometric shape has a unique ID.
– Upon showing the list of created shapes, the ID is shown together with other information for each shape.
– The user can modify some of the properties of each shape (size etc.).
– The user can delete all existing shapes.
The geometric shapes have to be implemented following the hierarchy given in Figure 1 on page 2

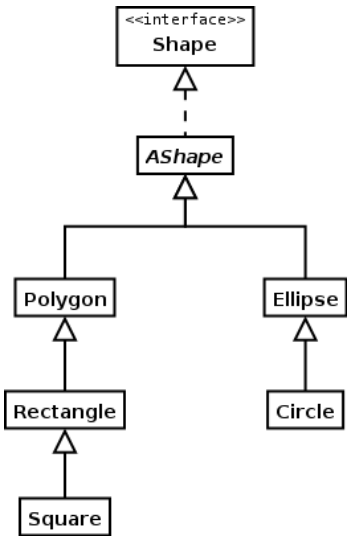

Figure 1 – Hierarchy of geometric shapes

The `Shape` interface contains the following methods :
– `double perimeter()` : return the circumference of a shape.
– `double area()` : return the area of a shape.
– `void change()` : allows the modification of a shape.
– `int getID()` : returns the unique ID of a shape.
– `void setID()` : fixes the unique ID of a shape.
– `String toString()` : prints a shape as ASCII text.
– `void draw(Graphics g)` : prints a shape on the screen.

– `boolean contains(Point p)` : test if a given point `p` is inside a given shape or not.

Some explanations regarding the implementation of the shapes application :

– The perimeter `P` of an ellipse can be approximated by a formula discovered by the indian mathematician *Srinivasa Aiyangar Ramanujan (1887-1920)* in 1914 :

$$P \approx \pi \cdot \sqrt{\frac{2(a^2+b^2)-(a-b)^2}{2}}$$

where a and b are the small and the big diameter respectively.

– The area A of a polygon can be computed by the following forumula (based on Green's Theorem in the plane) :

$$A = \frac{1}{2}\sum_{i=0}^{n-1} a_i \text{ avec } a_i = x_i y_{i+1} - x_{i+1} y_i$$

where the points $(x_i, y_i), i = 0, \ldots, n$ with $x_0 = x_n$ and $y_0 = y_n$ are the points constructing the polygon. **Remark :** this formula gives an erronous value if the poglygon is not convex.

– A point $P\ (p_1, p_2)$ is inside a circle C with center (x_1, x_2) and diameter r if

$$(p_1 - x_1)^2 + (p_2 - x_2)^2 \leqslant r^2$$

– A point $P\ (p_1, p_2)$ is inside an ellipse $\mathcal{E}$ with width w,height h and center (x_1, x_2) if

$$\left(\frac{p_1 - x_1}{w}\right)^2 + \left(\frac{p_2 - x_2}{h}\right)^2 \leqslant 1$$

– The mathematical function `sqrt(x)` is found in the `java.lang.Math` package (see [1]).

– The class `java.awt.Graphics` allows you to draw geometric shapes (see [1]).

– You will find online [2] a skeleton of the implementation. The code has gaps you have to fill. They are indicated by the following comment : `Add your code for the Serie 2(1) of Genie Logiciel here!!.`

– All the necessary classes are in the distributed zip file :
  – Complete the `Polygon.java` and the `WorkShapes.java` files.
  – Create the files `Ellipse.java` and `Circle.java` accordingly to the above discussion.

– The class `WorkShapes` is the main class of the application. It implements most parts of the GUI. Fill the gaps indicated by the comment : `Add your code for the Serie 2(1) of Genie Logiciel here!!.`

– Starting with the distributed files, create a new project in your IDE and edit the code and the GUI as discussed.

– Check that your code is working by playing around with the new application.

– Edit the dialog box `shapes.gui.About.java` and put your names in here.

– The application can be launched the following ways :
  – From console : `java shapes.WorkShapes`
  – From ANT : `ant run`
  – From your IDE

(4) Shapes – Part 2

Starting from Part 1, implement the following extensions :

– The application should also handle triangles. Thus, create the class `Triangle.java` exending `Polygon.java`. Remember that the area of a triangle can be computed with the following formula :

$$A = \sqrt{p(p-a)(p-b)(p-c)} \text{ avec } p = \frac{a+b+c}{2}$$

– Implement a function allowing to sort the list of shapes in ascending order :
  – by area
  – by perimeter
  – by ID

  These operations refer to the menu entries *Sort by Area*, *Sort by Perimeter* and *Sort by ID* of the *Edit* menu.

– It should be possible to move either one particular shape or all shapes togheter. These operations refer to the menu entries *Move* and *Move all* of the *Edit* menu.

– It should be possible to delete a given shape. This operation refers to the menu entry *Delte a shape* of the *Edit* menu.
– It should be possible to save a list of shapes to the harddrive. Moreover, it should also be possible to open such a file and show all the shapes as defined in the file. These operations refer to the menu entries *Open* and *Save as...*

Some indications :

– The solution of Part 1 is available online [2] : `Shapes_Solution.zip`. Download this archive and get familiar with the application.
– The application can be started with : `java -jar Shapes_Solution.jar`
– To sort the list of shapes, use the `java.util.Collections` class and its static method `sort(List _list, Comparator _c)`
– You have to implement the following comparators by extending the `java.util.Comparator` interface :
  – `ByAreaShapesComparator`,
  – `ByPerimeterShapesComparator` and
  – `ByIDShapesComparator`.

  See [2] for additional information.
– In order to be able to save the list of shapes to the harddrive, all shapes have to implement the `java.io.Serializable` interface (see [1]). Therefore, it is recommended to modify the `Shape` class like :

```
public interface Shape extends java.io.Serializable
```

– To learn how to write objects into files and read them back from files, have a look at the documentation of the `java.io.ObjectOutputStream` and `java.io.ObjectInputStream` classes. (see [2]).
– Rember that after loading a list of shapes from the harddrive new shapes can be added to the list (Hint :

  ID).

  Don't spend too much time about handling special cases when saving/reading files (like erase already existing file etc.). A short error message is enough.

  Take as starting point the source code provided online [2] (`shapes_source.zip`) and fill the gaps indicated by the comments :
  – `Add your code for the Serie 1(1) of Genie Logiciel here!!`
  – `Add your code for the Serie 1(2) of Genie Logiciel here!!`

Références

[1] Java API, 2012. http://docs.oracle.com/javase/7/docs/api/ (accessed Feb 25, 2013).

[2] Jacques Pasquier. Génie logiciel I, 2013. http://moodle2.unifr.ch/course/view.php?id=1252 (accessed Feb 25, 2013).