TP 2 : Aggregation, Composition, Inheritance

©2022 Ghiles Ziat
ghiles.ziat@epita.fr

In the following exercises, you are free to answer the questions using the language of your choice among Java and C++, except when it is explicitly requested to use one of the two languages or both.

Exercice I : Aggregation, Composition

Q1 – Propose a class architecture for the following specification : We want to manipulate geometric shapes in two dimensions : Triangle, Circle, Rectangle and Polygon. All our forms will have as a common point to have a number of vertices and a way of calculate their areas.

Q2 – Create a class `Point` with two fields `double x,y` and a constructor with parameters.

Q3 – Add a `Point vector(Point p)` method that returns a new point instance corresponding to the vector[1] between the calling instance and the point passed as a parameter. By example, the vector between the point $(0, 1)$ and the point $(3, 5)$ will give the point $(3, 4)$.

Q4 – Add a `Point translate(Point v)` method that returns a new point instance corresponding to the translation of the calling instance, by the vector passed in parameter. We reminds that a translation consists in adding the components x and the components y between them. For example, the translation of the point $(0, 1)$ by the vector $(3, 5)$ will give the point $(3, 6)$.

Q5 – Add a `Point scal(double coef)` method that returns a new instance of point or the components x and y have been multiplied by the coefficient passed as a parameter.

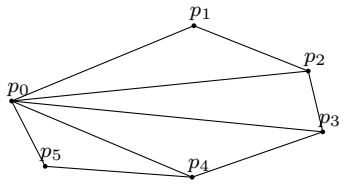
Q6 – Propose a modeling and create a class for the `Triangle`.

Q7 – Add a `double area()` method that calculates the area of the triangle. **Reminder :**

- the area of a triangle (ABC) is $\frac{1}{2} \cdot |det(\overrightarrow{AB}, \overrightarrow{AC})|$ where det is the determinant of two vectors.
- the determinant of two vectors $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ and $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$ is given by : $x_1y_2 - y_1x_2$

1. A vector has, like a point, fields x and y. To go quickly, we will not define a `Vector` class and we will will reuse the `Point` class to model vectors.

Q8 – Create a class `Polygon` with a field `vertices` containing the points of the convex hull of the polygon, and a `asTriangles` field containing a array/vector of triangles.



Q9 – Add a constructor that takes an array/vector of points $(p_0, p_1, p_2, p_3, ..., p_{n-1}, p_n)$ corresponding to vertices of the polygon. The constructor will additionally store an array/vector of triangles covering the polygon in the `asTriangles` field. For this, carry out a course two by two of the vertices to construct the list of triangles $(p_0, p_1, p_2), (p_0, p_2, p_3), ..., (p_0, p_{n-1}, p_n)$. We will assume that we have more than 2 vertices and that these are given clockwise, from so that you can connect them in the order in which they are supplied.

Q10 – Add to the class `Polygon` a method `double area()` which calculates the area of a polygon.

Q11 – **Bonus** Define a method `Point random()` in class `Triangle` which draws a random point in a triangle. One way [2] to do is to draw random coefficients r_1 and r_2 then,

— if $r_1 + r_2 < 1$ to translate of the point A by the vectors $r_1 \cdot \overrightarrow{AB}$ and $r_2 \cdot \overrightarrow{AC}$

— otherwise, to translate the point A by the vectors $(1 - r_1) \cdot \overrightarrow{AB}$ and $(1 - r_2) \cdot \overrightarrow{AC}$

Q12 – **Bonus** We now want to draw randomly and uniformly a point in a polygon. A way of do is first randomly draw a triangle, then draw a point in this triangle. Pay attention to uniformity : the triangles with the largest area are more likely to be drawn. Define the method `Point random()` which draws a point randomly in a polygon.

Q13 – **Bonus** Attached is a script bash *show.sh* which will allow you to visually test the uniformity of your draw : in your `main`, display on the standard output 1000 randomly drawn points in "x y" format and display on the error output the vertices of the polygon. You can now run your program by prefixing the command with *./show.sh* as follows :

`./show.sh java Main` (or the *c++* equivalent : `./show.sh ./a.out`)

Make sure you have *gnuplot* installed on your machine and to have given execution rights to the script beforehand (in running `chmod +x show.sh`) En savoir plus sur ce texte source Vous devez indiquer le texte source pour obtenir des informations supplémentaires Envoyer des commentaires Panneaux latéraux

Exercice II : Inheritance

Q1 – Propose a class architecture for the following specification : wxe want to handle a general inventory management structure, with a maximum capacity, a number of products stored, methods to

2. For curious, a geometric interpretation of this formula is given in appendix.

tell if the structure is empty or full, and methods for adding and product withdrawal. We also want to have two more structures specifications, a FIFO (first-in first-out) structure and a LIFO (last-in first-out) with different withdrawal methods, but with the same method of addition.

Q2 – Define an abstract class `Storage` in which to will use a `memory` array to store integers, an integer which will count the current number of elements stored in the array, as well as an integer which will correspond to the index of the next value to write.

Q3 – Add a constructor that takes a size as a parameter maximum, as well as methods `boolean isFull()` and `boolean isEmpty()` which respectively check if the array is full or empty.

Q4 – Add to the class a method `void add(int i)` which adds an integer to memory. The addition of an element will be done in a way cyclic in the table : we will bring back the index of addition of the next element in range $[0; CAPACITY]$ using operator %. If the array is full, the `add` method will do nothing.

Q5 – We are now going to create two subclasses which will inherit of `Storage` and which will respectively implement structures FIFO (first-in first-out) and LIFO (last-in last-out).

Appendices

Display script

Copy-paste the following code into a file *show.sh*

```
#!/bin/bash
# we run the arguents of the script as a standalone command
$@ > pts.txt 2> vert.txt
# we rewrite the first vertices at the end of the file to "close" the
# drawing of polygon
echo $(head -n 1 vert.txt) >> vert.txt
# we ask gnuplot to draw the border of the polygon and to plot the dots
gnuplot -persist -e "reset; unset key; set terminal pngcairo size 500,400;\
set output 'uniform.png'; set style line 1 linetype 1 pointtype 4;\
plot 'pts.txt' with points pointtype 3, 'vert.txt' with linespoints linestyle 1"
# we remove the files we have generated and open the image
rm vert.txt pts.txt; open uniform.png
```

Random draw in a triangle

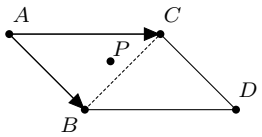
To draw a point randomly in a triangle ABC we will construct a parallelogram $ABDC$ (where D is the mirror image of A by ratio in the middle M of BC), draw a point randomly in $ABDC$ and bring that point back into ABC. Let's now draw a point P randomly in $ABDC$:

$$P = r_1 \cdot \overrightarrow{AB} + r_2 \cdot \overrightarrow{AC}$$

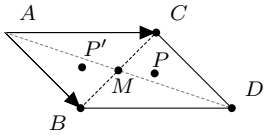
or r_1 and r_2 are two random coefficients between 0 and 1.

There are then two possibilities :

1. If P is in ABC then we keep it
2. if it is in BCD, we return its symmetric with respect to M



1) On garde P



2) On prend son symétrique par rapport à M

We can finally notice that if $r_1 + r_2 \leq 1$, then P will be necessarily in ABC, and otherwise P is in BCD.