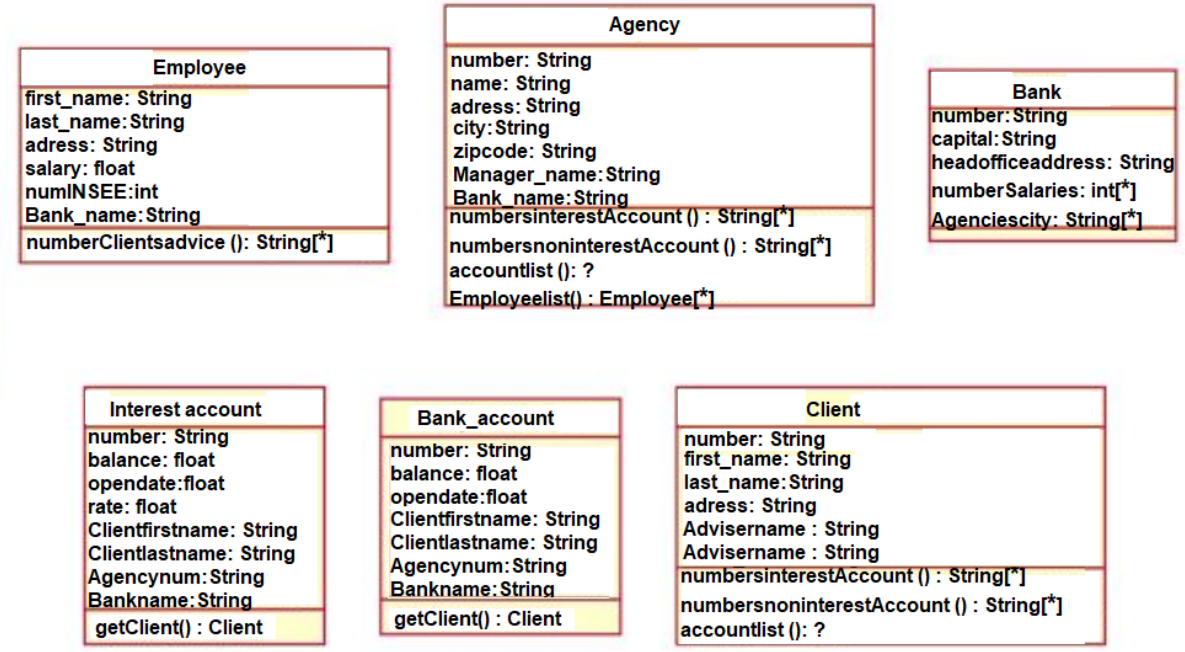
Course : Object oriented programming
Tutorial N°9 : Class diagram

Exercise N°1:

An initial analysis led to the creation of the following classes for organizing the banks in a consortium. This is only a first draft, which should be carefully reviewed and corrected: some attributes are redundant.



1- Introduce associations between classes to limit the redundancy of information.
   It will be assumed that the introduced associations give rise to accessors (getters, setters) without the need to include all of them in the diagram.
2- Use inheritance to optimize the model.
   You may introduce a new class if necessary.

Exercise N°2:

For each of the following statements, provide a class diagram. Unless otherwise specified, do not represent attributes or operations.

1. Every writer has written at least one work.
2. A rectangle can be defined by two vertices which are points (in the case where the sides are perfectly vertical and horizontal). A rectangle is constructed from the coordinates of two points. It is possible to calculate its area and perimeter, or to translate it. Model the attributes and operations.
3. People can be associated with universities both as students and as professors.
4. Cinemas are composed of several screening rooms. Films are shown in rooms. The corresponding screenings each take place at a specific time (you may use an association class).

Exercise N°3:
The preliminary study of billing management in a company has made it possible to identify the following business rules:

- Invoices are numbered and dated.
- Each invoice includes a certain number of products. A quantity is associated with each product on an invoice.
- A product is identified by a code, and a label provides a clearer identification.
- The pre-tax price of products must be known. The total price including tax (TTC) is calculated.
- The VAT applicable to each product depends on its type. For each product type, a label and the associated VAT rate are known.
- Invoices must not concern more than one customer. Each one with a name and an address.
- It must be possible to calculate the total amount including tax (TTC) and the total amount excluding tax (HT) of the invoices.
- It must be possible to add and remove products from an invoice. These are the only two ways to modify the contents of an invoice. Moreover, if a product already present on an invoice is added again, the quantity of that product is simply increased.

Provide a class diagram to represent this information.

Exercise N°4:
Provide the class diagram corresponding to the following **PointPlan** class:

```
import java.util.Scanner;
class PointPlan {
   private float abscisse;
   private float ordonnee;
   public PointPlan() {
      this.abscisse = 0;
      this.ordonnee = 0;
   }
   public PointPlan(float x, float y) {
      this.abscisse = x;
      this.ordonnee = y;
   }
   public PointPlan(PointPlan p) {
      this.abscisse = p.getAbscisse();
      this.ordonnee = p.getOrdonnee();
   }
   private float getOrdonnee() {
      return this.ordonnee;
   }
   private float getAbscisse() {
      return this.abscisse;
   }
   public boolean egaleA(PointPlan p) {
      return (this.getAbscisse() == p.getAbscisse()
&&
            this.getOrdonnee() == p.getOrdonnee());
   }
   public void init() {
      Scanner input = new Scanner(System.in);
      System.out.print("Abscisse ? ");
      this.abscisse = input.nextFloat();
      System.out.print("Ordonnée ? ");
      this.ordonnee = input.nextFloat();
   }
   public void translate(float dx, float dy) {
      this.abscisse += dx;
      this.ordonnee += dy;
   }
   public double distance() {
      return Math.sqrt(this.abscisse * this.abscisse
+
                  this.ordonnee * this.ordonnee);
   }
   public String toString() {
      return "(" + this.getAbscisse() + ", " +
this.getOrdonnee() + ")";
   }
   public void affiche() {
      System.out.println(this.toString());
   }
}
```