République Algérienne Démocraique et Populaire
Ministère de l'Enseignement Supérieure et la Recherche Scientifique
Université d'Abou Bekr Beklkaid-Tlemcen
Faculté des Sciences
Département d'Informatqiue
Spécialité: 3 ème Ingénieur: Réseaux
Matière: Modélisation des Systèmes d'Information

Automne 2025 13:30-15:00

TD N° 5: Génération de code et de diagrammes

I. Objectifs du TD: Ce TD a pour objectif de :

- Comprendre et appliquer les relations UML: tels que: extends (héritage), implements (réalisation d'interface), association, agrégation, composition, dépendance, multiplicités, navigabilité.
- Convertir un diagramme de classes UML vers du code (L'ingénierie de modèles): Transformer un diagramme de classes UML en code Java, Python, et en structures Web (JS/React/Flask).

  Traduire les : attributs, méthodes, types simples, méthodes abstraites, classes associées, interfaces, collections (List, ArrayList, []), packages / modules / importations.
- Convertir un code existant vers un diagramme UML: reverse engineering.
- **Exercice 1: UML vers Java**

  - Soit le diagramme UML suivant:

Figure 1: Digramme de Classe UML: Client—>Commande

- Écrivez les classes Java correspondantes.
- Implémentez l'association **bidirectionnelle** avec cardinalité: 1 ..*.

- **Exercice 2: UML vers Python**

  Pour le même digramme de l'excercice 1, Codez les classes en Python.

- **Exercice 3:**

  Même exercice, générez le code Web correspondant:
  Créez les routes :
  - POST /client/add
  - POST /commande/add

- **Exercice 4: Java vers UML**

  Soit le code Java suivant, donnez le diagramme UML correspondant.

Listing 1: Code java: Classe Motor

```java
public class Motor {
    private int puissance;
    private String type;
    private boolean etat;

    // Constructeur
    public Motor(int puissance, String type) {
        this.puissance = puissance;
        this.type = type;
        this.etat = false; //  teint  par  d faut
    }


    public void demarrer() {
        etat = true;
        System.out.println("Moteur  d  marr  !");
    }


    public void arreter() {
        etat = false;
        System.out.println("Moteur  arr  t  !");
    }

```

```
25
26      // Getters et setters (optionnel)
27      public int getPuissance() { return puissance; }
28      public String getType() { return type; }
29      public boolean isEtat() { return etat; }
30  }
```

Listing 2: Code java: Classe Car

```
1  class Car {
2      private String brand;
3      private Motor motor;
4
5      public void start() {}
6  }
```

- **Exercice 5: Pyhton vers UML**
  - Soit le code Python suivant, donnez le diagramme Uml correspondant.

Listing 3: Code Python

```
1  class Etudiant:
2      def __init__(self, nom, age):
3          self.nom = nom
4          self.age = age
```

- **Exercice 6: Code web vers UML**

  - Soit le code React suivant, donnez le diagramme Uml correspondant.

Listing 4: Composant React simple

```
1  const [user, setUser] = useState({ email: "", password: "" });
```

- **Exercice 7: UML vers Java**

  - Donnez le code java correspodant au digramme UML suivant.

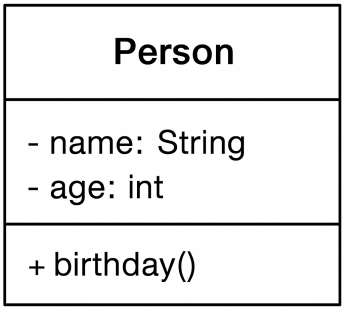

Figure 2: Classe UML: Person

- **Exercice 8: UML vers Java**

  - Traduisez le diagramme UML suiavant en code java.

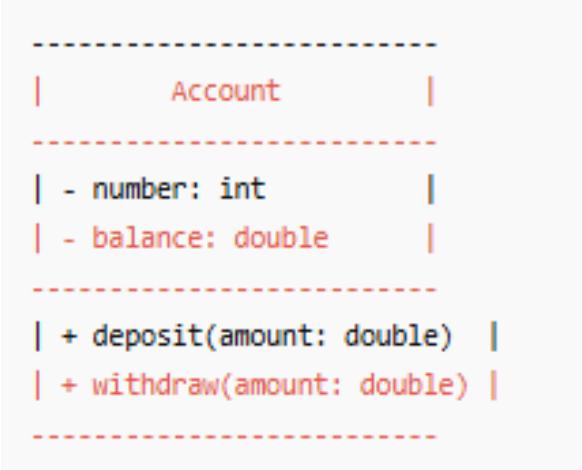

Figure 3: Diagramme UML pour Deposit/Withdraw

- **Exercice 9: Java vers UML**

  - Convertissez ce diagramme en Java.

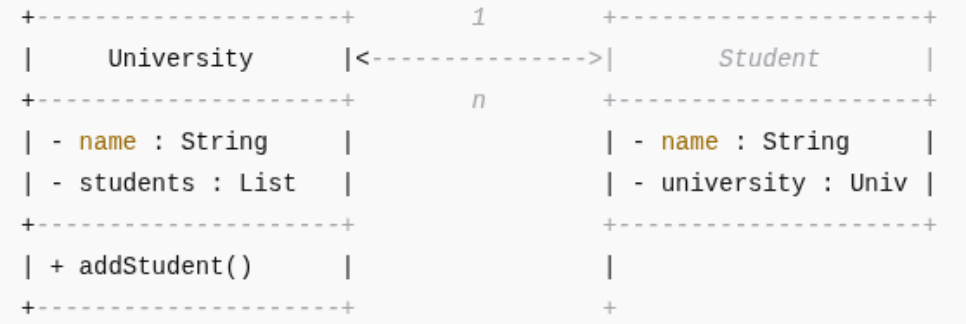

Figure 4: Diagramme UML University —> Student

- **Exercice 10: Java vers UML**

  - Donnez le code java pour les classes UML suivantes:

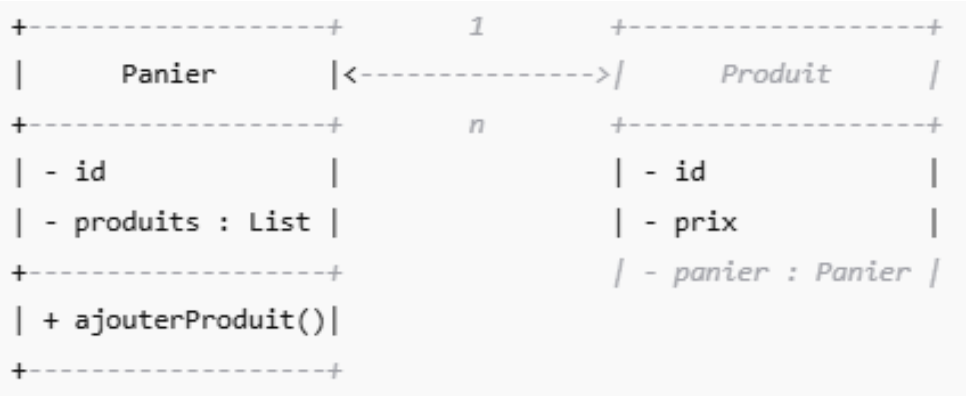

Figure 5: Diagramme UML Panier Produit

- **Exercice 11: UML –> Java**

  - Donnez le Code java correspodant à ce diagramme de classe UML.

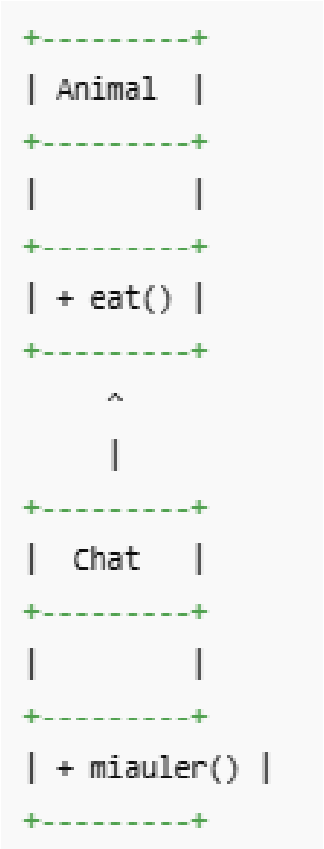

Figure 6: Diagramme de clases pour Animal

- **Exercice 12: UML—> Java**

  - Donnez le Code java correspodant à ce diagramme de classe UML.

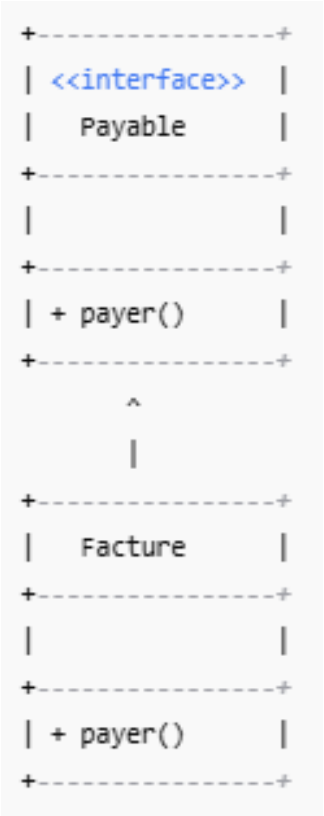

Figure 7: Diagramme de classe Payable-facture

- **Exercice 13: UML –> Python/Java**

  - Donnez le Code Python e Jva respectivement correspodant à ce diagramme de classe UML.

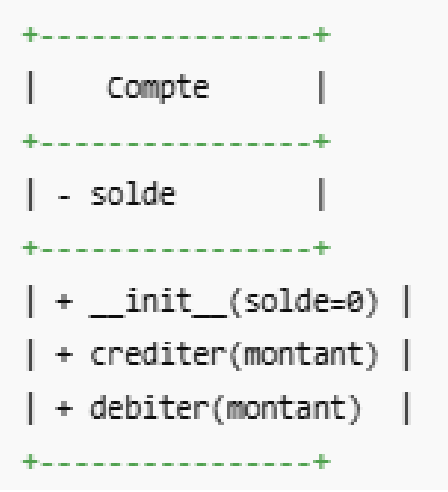

Figure 8: Diagramme de classe Compte montant

Corrigé des exercices

- **Corrigé exercice 1**

Listing 5: Code java : classe Commande

```java
import java.util.Date;

public class Commande {

    private int id;
    private Date date;
    private Client client;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public Client getClient() {
        return client;
    }

    public void setClient(Client client) {
        this.client = client;
    }
}
```

Listing 6: Code java : Classe Client

```java
import java.util.ArrayList;
import java.util.List;

public class Client {

    private int id;
    private String nom;
    private List<Commande> commandes = new ArrayList<Commande>();

    public int getId() {
        return id;
    }

```

```
    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public List<Commande> getCommandes() {
        return commandes;
    }

    public void setCommandes(List<Commande> commandes) {
        this.commandes = commandes;
    }
}
```

- **Corrigé exercice 2**

Listing 7: Code Python: Classe commande

```
class Commande:
    def __init__(self, id, date):
        self.id = id
        self.date = date

class Client:
    def __init__(self, id, nom):
        self.id = id
        self.nom = nom
        self.commandes = []

    def ajouter_cmd(self, commande):
        self.commandes.append(commande)
```

- **Corrigé exercice 3**

Listing 8: Code API Flask

```
@app.post("/client/add")
def add_client():
    data = request.json
    return {"status": "client ajout "}

@app.post("/commande/add")
```

```
def add_commande():
    data = request.json
    return {"status": "commande ajout e"}
```

- **Corrigé exercice 4**

Figure 9: diagramme UML Car—>Motor

- **Corrigé exercice 5**

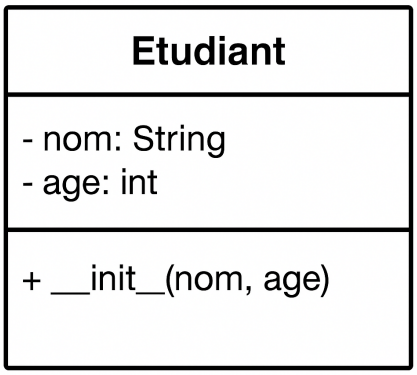

Figure 10: Diagramme de classe Etudiant

- **Corrigé exercice 6**

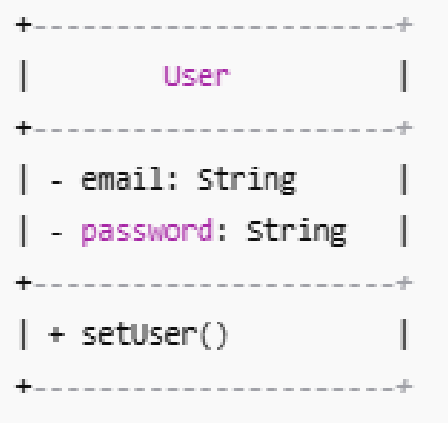

Figure 11: Diagramme de classe UML user

- **Corrigé exercice 7**

Listing 9: Code java: Classe Person

```java
public class Person {
    private String name;
    private int age;

    public void birthday() {}
}
```

- **Corrigé exercice 8**

Listing 10: Code java: Classe bankAccount

```java
public class BankAccount {
    private int number;
    private double balance;

    public void deposit(double amount) {}
    public void withdraw(double amount) {}
}
```

- **Corrigé exercice 9**

Listing 11: Code java: classe University

```java
import java.util.ArrayList;
import java.util.List;

public class University {

    private String name;
    private List<Student> students = new ArrayList<>();

    public University(String name) {
        this.name = name;
    }

    // Ajouter un   tudiant   (bidirectionnel)

     public String getName() {
        return name;
    }
   public void setName(String name) {
        this.name = name;
    }
    public List<Student> getStudents() {
        return students;
    }
 public void setStudents(List<Student> students) {
        this.students = students;
        // Synchronisation bidirectionnelle
        for (Student s : students) {
            s.setUniversity(this);
        }
    public void addStudent(Student s) {
   ....
    }

}
```

Listing 12: Code java: classe Student

```java
public class Student {

    private String name;
    private University university; // r f rence    l'
       universit

    public Student(String name) {
        this.name = name;
    }
  public University getUniversity() {
        return university;
    }
    public void setUniversity(University u) {
```

```
13          this.university = u;
14      }
15      public String getName() {
16          return name;
17      }
18
19        public void setName(String name) {
20          this.name = name;
21      }
22  }
```

- **Corrigé exercice 10**

Listing 13: Code java : Classe Produit

```
1   public class Produit {
2
3       private int id;
4       private double prix;
5       private Panier panier;
6
7       public Produit() {
8       }
9
10      public int getId() {
11          return id;
12      }
13
14      public void setId(int id) {
15          this.id = id;
16      }
17
18      public double getPrix() {
19          return prix;
20      }
21
22      public void setPrix(double prix) {
23          this.prix = prix;
24      }
25
26      public Panier getPanier() {
27          return panier;
28      }
29
30      public void setPanier(Panier panier) {
31          this.panier = panier;
32      }
33  }
34
35
36  \begin{lstlisting}[style=JavaStyle, caption ={Code java : Classe
```

```
   Panier}]
```

```
import java.util.List;
import java.util.ArrayList;

public class Panier {

    private int id;
    private List<Produit> produits = new ArrayList<>();

    public Panier() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public List<Produit> getProduits() {
        return produits;
    }

    public void setProduits(List<Produit> produits) {
        this.produits = produits;
    }

    public void ajouterProduit(Produit produit) {
        ....
    }
}
```

- **Corrigé exercice 11**

Listing 14: Code java : Caasse Animal

```

public class Animal {
    public void eat() {}
}
```

Listing 15: Code java : Classe Chat

```

public class Chat extends Animal { //relation d'h ritage
    public void miauler() {}
}
```

- **Corrigé exercice 12**

Listing 16: Code java: Classe Payable

```java

public interface Payable {
   public void payer();
}
```

Listing 17: Code java: Classe Facture

```java

public class Facture implements Payable { //relation d'
   implementation d'interface
    public void payer() {}
}
```

- **Corrigé exercice 13**

Listing 18: Code Python: Classe Compte

```python

class Compte:
    def __init__(self, solde=0):
        self.solde = solde

    def crediter(self, montant):
        self.solde += montant

    def debiter(self, montant):
        self.solde -= montant
```

Listing 19: Code Java: Classe Compte

```java

public class Compte {

    private double solde;
 //Constructeur avec solde initial (par d faut = 0)
    public Compte() {
       ...
    }

    public Compte(double solde) {
        this.solde = solde;
    }
 public double getSolde() {
        return solde;
    }
  public void setSolde(double solde) {
        this.solde = solde;
    }

    public void crediter(double montant) {
      ....
    }

    public void debiter(double montant) {
       ....
```

```
26          }
27
28
29  }
```

Bon courage