

ANA2 - Lucky Summary

Sm!le42

9 mai 2021

Table des matières

1 Diagramme d'activité	1
1.1 Transitions automatiques	1
1.2 Début et Fin	2
1.3 Conditions	2
1.4 Itérations (boucles)	2
2 Classes et objets	3
2.1 Classes	3
2.1.1 Visibilités	3
2.1.2 Attributs	4
2.1.3 Opérations	4
2.2 Objets	4
3 Associations	4
3.1 Associations 1-1 et 1-N	4
3.2 Associations N-N	6
3.2.1 Multiplicités	6
4 Énumérations, structures et compositions	7
4.1 Énumérations	7
4.2 Structures	9
4.3 Compositions	9

1 Diagramme d'activité

1.1 Transitions automatiques

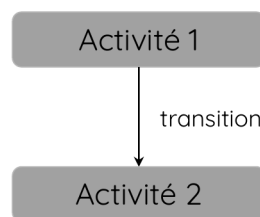


FIGURE 1 – Transitions automatiques

Une fois l'activité 1 terminée, on passe automatiquement à l'activité 2.

1.2 Début et Fin

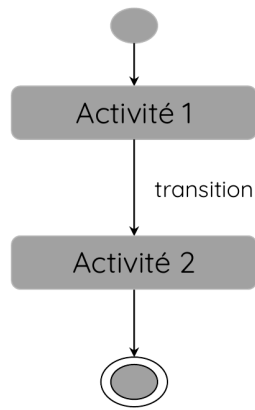


FIGURE 2 – Début et Fin

Le *rond gris* représente le **début** du programme, et le *rond gris entouré d'un cercle* représente la **fin** du programme (fin normale ou fin après erreur).

1.3 Conditions

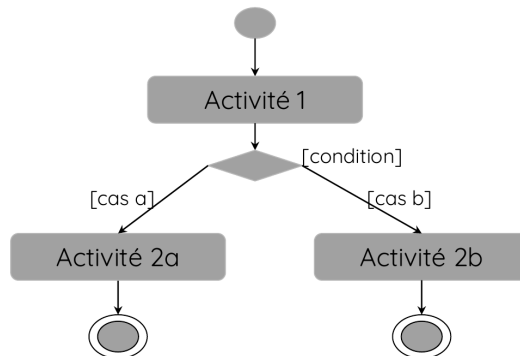


FIGURE 3 – Conditions

Arrivé au *losange* (=condition), il y a **deux possibilités**, soit **cas a**, soit **cas b**. (If, Then, Else)

1.4 Itérations (boucles)

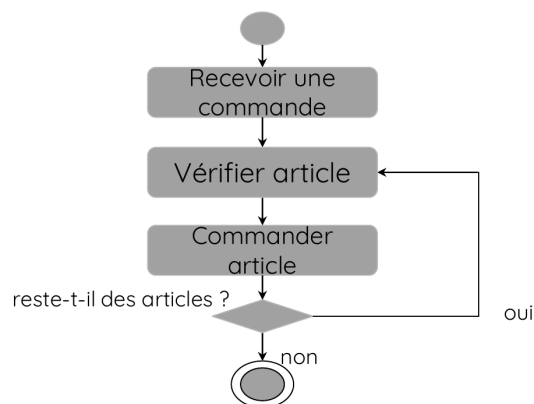


FIGURE 4 – Itérations

Les *instructions* seront exécutées **en boucle** tant que la *condition* (*losange*) est **vraie**. (For, While)

2 Classes et objets

2.1 Classes

```
1 //Exemple de classe Compte en Java
2 public class Compte {
3     //solde > decouvertMax
4     private int numero;
5     private int solde;
6     private int decouvertMax;
7
8     public Compte(int numero, int solde) {
9         //...
10    }
11
12    public int consulterSolde() {
13        //...
14        return int;
15    }
16
17    public void crediter(int somme) {
18        //...
19    }
20
21    public void debiter(int somme) {
22        //... (solde > decouvertMax)
23    }
24 }
```

Représentation UML de la classe Compte :

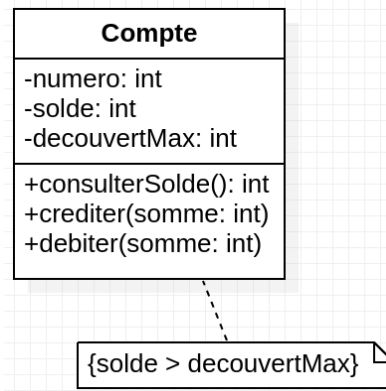


FIGURE 5 – Classes

2.1.1 Visibilités

Il existe 4 types de visibilité :

+	public
-	private
~	package
#	protected

1. **public +**

La visibilité *public* signifie que les données sont accessibles depuis n'importe où.

2. **private -**

La visibilité *private* signifie que les données ne sont accessibles que depuis la même classe.

3. **package ~**

La visibilité *package* signifie que les données ne sont accessibles que depuis le même package.

4. **protected #**

La visibilité *protected* signifie que les données ne sont accessibles que depuis le même package ou les sous-classes.

2.1.2 Attributs

Les attributs d'une classe sont notés de la manière suivante :

Visibilité Type Nom ;

Ex :

private int numero ;

2.1.3 Opérations

Les opérations d'une classe sont notées de la manière suivante :

Visibilité Type de retour Nom (Parametres) { ... }

Ex :

public String getNom(int numero) { ... }

(Cette méthode prend donc un int en paramètre, et retournera un String)

2.2 Objets

Exemple d'objets de la classe **Compte** (vue plus haut) :

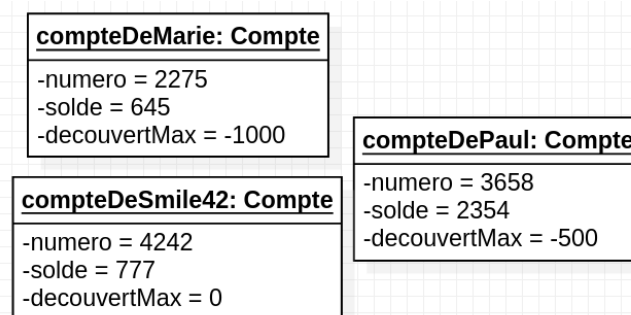


FIGURE 6 – Objets

1. **Identité :**

Permet d'identifier un objet.

2. **État :**

Caractéristiques de l'objet à un moment donné (Valeurs des attributs).

3. **Comportement :**

Ensemble des opérations qu'un objet peut exécuter ou subir.

- Create
- Read
- Update
- Delete

3 Associations

Connexions sémantiques durables entre des classes.

3.1 Associations 1-1 et 1-N

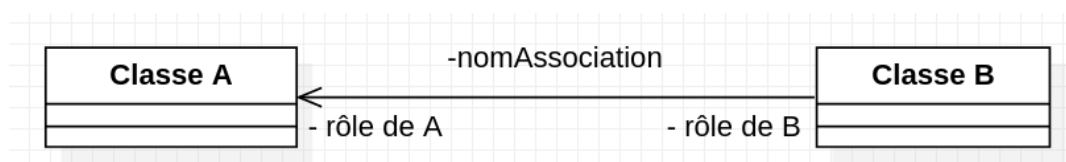


FIGURE 7 – Associations 1-1 1-N

1. Exemple "banque" :

Principe de banque avec des clients et des comptes (*Java + UML*) :

```
1 //Classe Client
2 public class Client {
3     private final String nom;
4     private final String prenom;
5     private final Compte compte;
6
7     public Client(String nom, String prenom, Compte compte) {...}
8     public String getNom() {...}
9     public String getPrenom() {...}
10    public Compte getCompte() {...}
11 }
```

```
1 //Classe Compte
2 public class Compte {
3     private final String numero;
4     private int solde;
5     private final Client titulaire;
6
7     public Compte(String numero, int solde, Client titulaire) {...}
8     public String getNumero() {...}
9     public int getSolde() {...}
10    public Client getTitulaire() {...}
11 }
```

Représentation UML de l'association des classes Client et Compte :

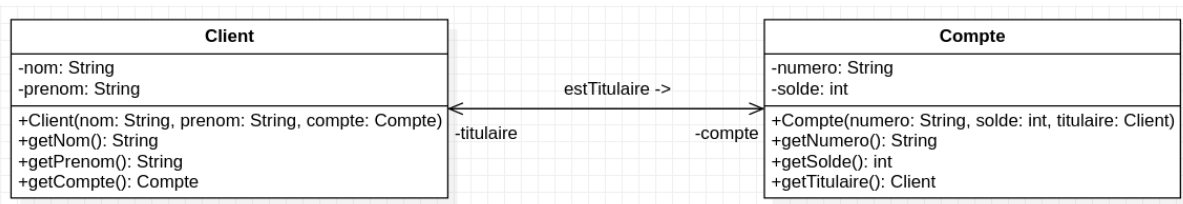


FIGURE 8 – Associations Client Compte (Exemple)

Ainsi, si on crée un Client "pierre" et qu'on lui attribue le compte "c1", on peut dire ceci :

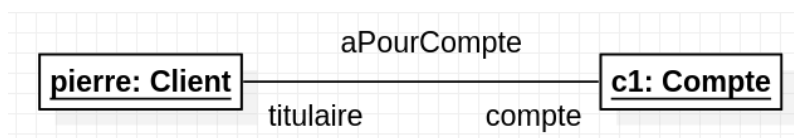


FIGURE 9 – Pierre : c1

- pierre a pour compte c1
- c1 joue le rôle de compte pour pierre
- pierre joue le rôle de titulaire pour c1
- le (*un des*) compte (*s*) de pierre est c1
- le (*un des*) titulaire (*s*) de c1 est pierre

2. Exemple "memory" :

Petit jeu de mémoire nommé "memory" (*Java + UML*) :

```
1 //Classe Memory
2 public class Memory {
3     private final Game game;
4 }
5 //Classe Card
```

```

6 public class Card {
7     private boolean hidden;
8     private int value;
9 }
10 //Classe Game
11 public class Game {
12     private Card cards; //Une seule carte!
13 }

```

Représentation UML du jeu "memory" :

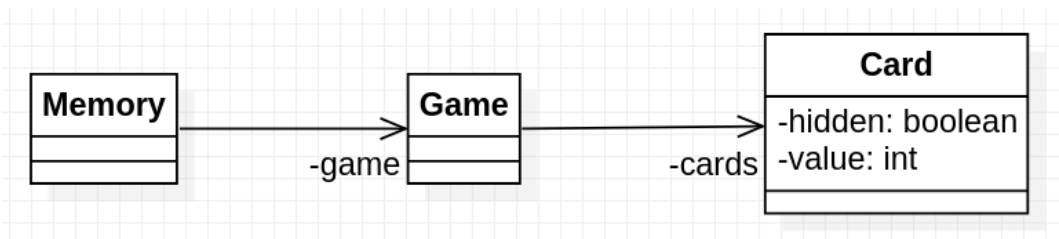


FIGURE 10 – Associations "memory" (Exemple)

3.2 Associations N-N

3.2.1 Multiplicités

Désigne le **nombre d'objets** qui peuvent participer à une **association**.

Syntaxe : { minimum } .. { maximum }

Les plus courants :

- **0..1** (= 0 ou 1)
- **1..1** ou **1** (= 1)
- **0..n** ou **n** ou **0..*** ou ***** (= tout)
- **1..n** ou **1..*** (= 1 ou plus)

Autres :

- **7..14** (= de 7 à 14)
- **0..21** (= de 0 à 21)
- **42..*** (= 42 ou plus)

Remarques :

- ***** = **0..*** (= tout)
- **3** = **3..3** (= 3)

1. Exemple "memory" :

Reprenons l'exemple "memory" (vu plus haut) :

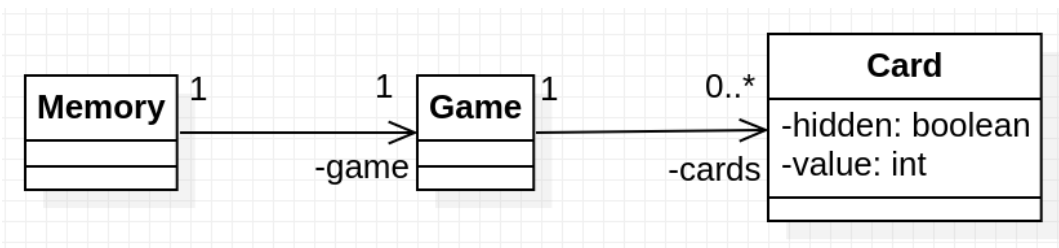


FIGURE 11 – Associations "memory" multiplicités (Exemple)

À gauche :

- À un objet Memory correspond un et un seul objet Game
- À un objet Game correspond un et un seul objet Memory

À droite :

- À un objet Card correspond un et un seul objet Game
- À un objet Game correspond 0 à plusieurs objets Card

2. Exemple "banque" :

Prenons un exemple similaire à l'exemple "banque" (vu plus haut) :

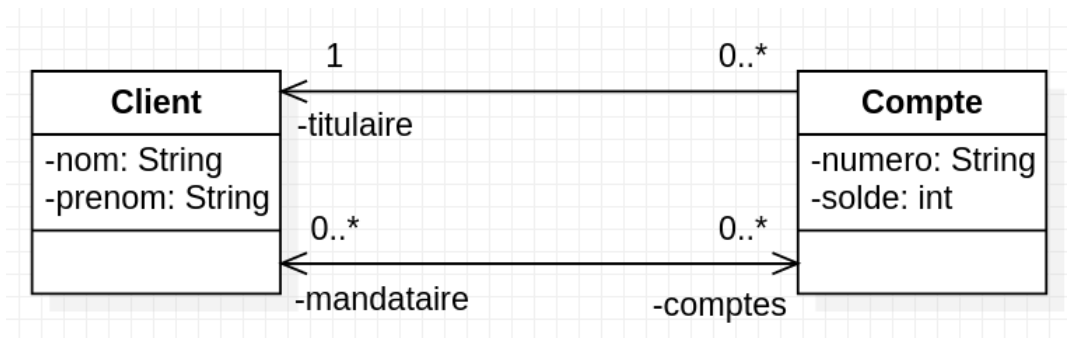


FIGURE 12 – Associations Client/Comptes multiplicités (Exemple)

On en déduit que :

- Un compte a un et un seul titulaire
- Un client peut n'être titulaire d'aucun compte, de un, ou plusieurs comptes
- Un compte peut n'avoir aucun mandataire, ou un, ou plusieurs
- Un client peut n'être mandataire d'aucun compte, de un, ou de plusieurs comptes

3. Exemple "université" :

Une université veut informatiser son administration pour gérer plus facilement les dossiers **étudiants**. Les étudiants ont un *identifiant* et un *nom*. Un étudiant fait partie d'un **département**. Ces derniers sont regroupés dans des **facultés**.

Voici un diagramme UML représentant cette structure :

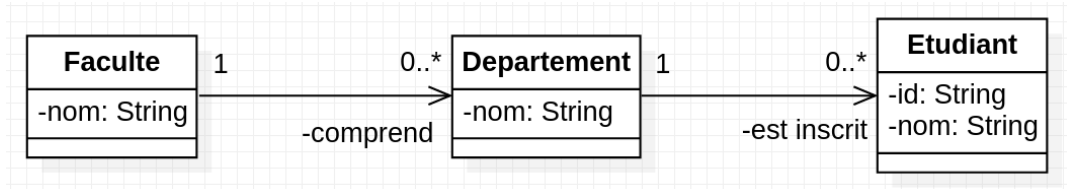


FIGURE 13 – Associations "université" (Exemple)

Et son code Java ressemblerait à ceci :

```

1  //Classe Faculte
2  public class Faculte {
3      private String nom;
4      //...
5  }
6  //Classe Departement
7  public class Departement {
8      private String nom;
9      private Faculte faculte;
10     //...
11 }
12 //Classe Etudiant
13 public class Etudiant {
14     private String id;
15     private String nom;
16     private Departement departement;
17     //...
18 }
  
```

4 Énumérations, structures et compositions

4.1 Énumérations

Une énumération n'a **jamais** d'association.

1. Exemple "saison" :

Code Java :

```

1 //Exemple Saison
2 public enum Saison {
3     ETE, AUTOMNE, HIVER, PRINTEMPS;
4 }

```

Diagramme UML :

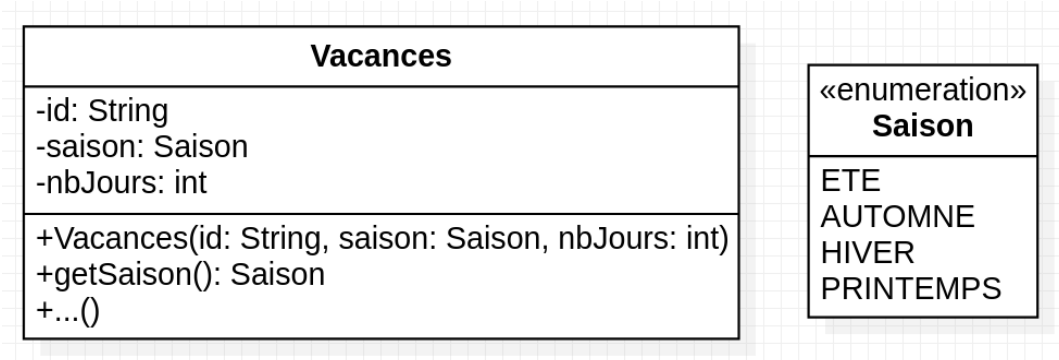


FIGURE 14 – Énumérations de saisons (Exemple)

2. Exemple "saison2" :

Classe saison2 (Java + UML) :

```

1 //Exemple Saison2
2 public enum Saison2 {
3     PRINTEMPS (21, 3), ETE (22, 6), AUTOMNE (23, 9), HIVER (21, 12);
4
5     private final LocalDate dateDebut;
6
7     private Saison2(int jour, int mois) {
8         LocalDate now = LocalDate.now();
9         this.dateDebut = LocalDate.of(now.getYear(), mois, jour);
10    }
11
12    public LocalDate getDateDebut() {
13        return this.dateDebut;
14    }
15
16    public Saison2 next() {
17        Saison2[] saisons = this.values();
18        return saisons[(this.ordinal() + 1) % saisons.length];
19    }
20 }

```

Code UML :

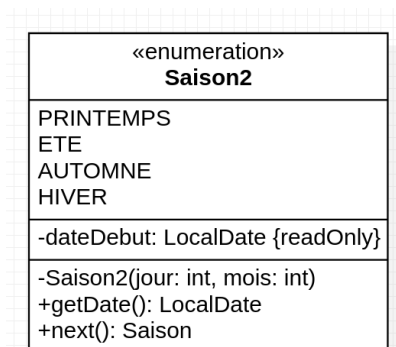


FIGURE 15 – Saison2

4.2 Structures

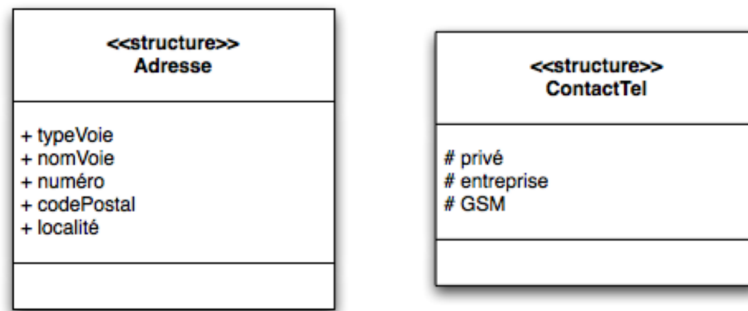


FIGURE 16 – Structures (Exemple)

Code Java de la structure *Adresse* :

```
1 //Classe Adresse
2 public class Adresse {
3     public TypeVoie typeVoie;
4     public String nomVoie;
5     public int numero;
6     public int codePostal;
7     public String localite;
8 }
9 //Enumeration TypeVoie
10 public enum TypeVoie {
11     AVENUE, RUE, CHAUSSEE, BOULEVARD;
12 }
```

4.3 Compositions

"Est composé de...", "fait partie de..."

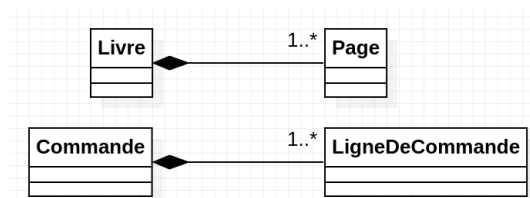


FIGURE 17 – Compositions (Exemples)