	Université de Corse - Pasquale PAOLI	
	Diplôme : L3 Informatique	2024-2025
	Module : Modélisation UML	
	TD N°1 : Diagrammes de classes Enseignant : Evelyne VITTORI	

L'objectif de ce TD est l'étude et la mise en pratique du formalisme des diagrammes de classes.

Les exercices vous conduiront notamment à résumer l'essentiel du formalisme. Les tableaux ainsi construits pourront vous être très utiles lors de vos révisions.

Partie I – RESUME DU FORMALISME

Exercice 1 - Classes et relations dans les diagrammes de classes

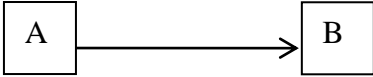
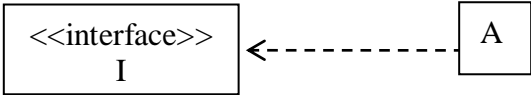
Question 1 - Représentation d'une classe

Complétez la deuxième colonne du tableau suivant en définissant les conventions de représentation des concepts de la première colonne dans un diagramme de classes UML.

Concept	Représentation graphique
Attribut/méthode de classe	
Classe abstraite	
Méthode abstraite	
Attribut/ méthode privée	
Attribut/ méthode public	
Attribut/ méthode protégé	
Constructeur	
Attribut dérivé	

Question 2 - Représentation des relations entre classes

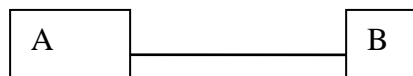
Complétez le tableau suivant en définissant les cases manquantes de la première et de la deuxième colonne. La deuxième colonne définit la représentation graphique des relations mentionnées dans la première colonne.

Relation	Représentation graphique
Association simple	
	
Agrégation	
Composition	
Dépendance	
Généralisation	
Réalisation ou implémentation	
	

Exercice 2 - Caractéristiques des associations

Complétez le diagramme suivant en insérant les informations suivantes :

- sens de lecture : de A vers B
- nom : Nom de l'association
- rA : rôle de la classe A dans l'association
- rB : rôle de la classe B dans l'association



Exercice 3 - Règles de traduction des associations lors de la phase de codage : ajouts d'attributs

Lors de la phase de conception détaillée (ou de codage) les associations sont traduites par l'ajout d'un ou plusieurs attributs dans les classes impliquées dans chaque association. Complétez le tableau suivant en indiquant dans la deuxième colonne les attributs à ajouter en précisant leur nom et leur type et la classe dans laquelle ils doivent être ajoutés.

Complétez le tableau suivant :

Association	classe A	classe B

Exercice 4 - On considère le diagramme de classe de la figure 1.

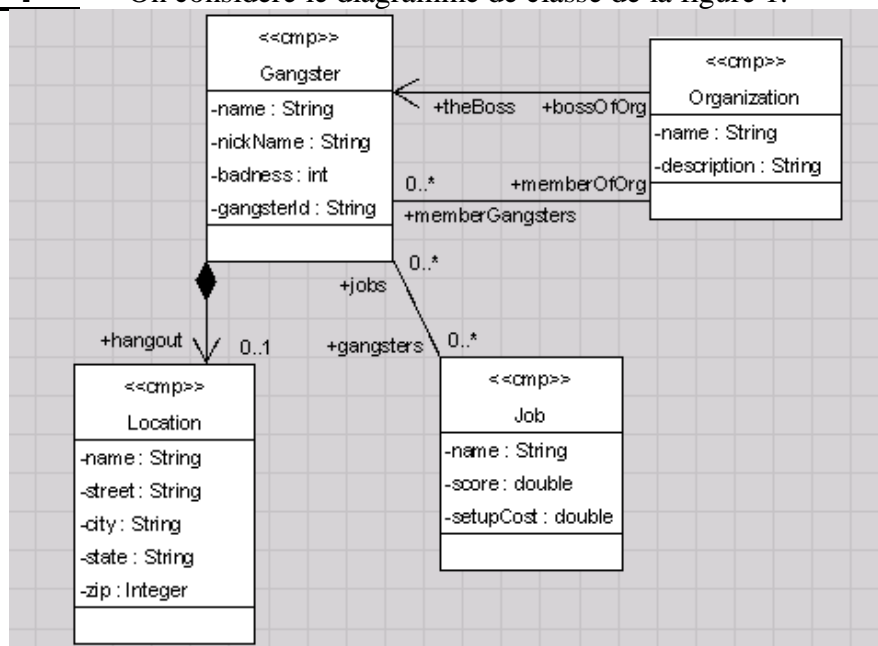


Figure 1 : Diagramme GangsterJob

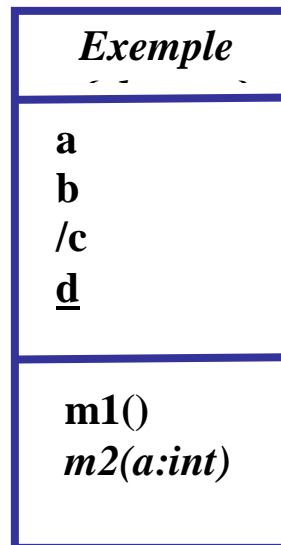
a) Quels sont les types de relations entre classes représentées dans ce diagramme ?

b) Que représente le terme « +theBoss » au niveau de la relation entre Gangster et Organization?

Précisez la signification du symbole « + ».

c) Que signifie le symbole ' - ' apparaissant devant un nom d'attribut ?

Exercice 5 - On considère le diagramme de classe suivant :



a) Que signifie le symbole / devant l'attribut c ?

b) Que signifie le fait que l'attribut d soit souligné ?

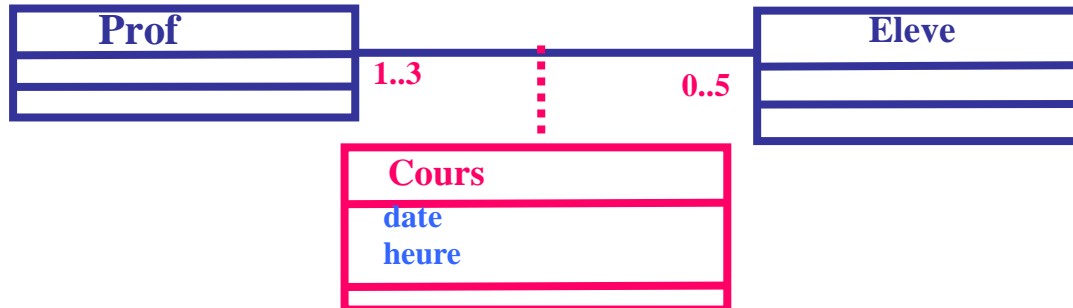
c) Que signifie le fait que la méthode m2 apparaisse en italique ?

d) Que signifie le fait que le nom de la classe apparaisse en italique ?

e) Donner une représentation équivalente ne faisant pas apparaître le nom de la classe en italique ?

Exercice 6 - Classe Association

Donner une représentation équivalente du diagramme ci-dessous sans utiliser de classe association.



Partie II – PRATIQUE DU FORMALISME

Exercice 7 - Reverse engineering

Définissez un diagramme de classe UML correspondant au programme Java ci-dessous en respectant les étapes suivantes:

- 1- Représentez les classes avec leurs attributs ;
- 2- Représentez les liens de généralisation/spécialisation (héritage) ;
- 3- Représentez les associations.

Attention : Lors de la phase de programmation, les associations présentes dans un diagramme UML sont traduites par l'ajout d'attributs dans les classes concernées. Lors du reverse-engineering, ces attributs ne doivent pas figurer dans le diagramme UML, ils sont représentés graphiquement par les associations. Ainsi, si vous faites figurer de tels attributs dans le diagramme, cela n'est pas correct car votre représentation devient redondante : vous écrivez deux fois la même chose (l'attribut dans la classe et le lien d'association qui le sous-entend).

```
public abstract class Bateau {
    private String nom;
    protected float temps;
    private Equipage equipage;

    public Bateau(String nom, Equipage e){
        this.nom=nom;
        this.temps=0;
        equipage=e;
    }

    public abstract float tempsPondéré();

    public String toString(){
        return nom+ " de " + equipage.getCapitaine() ;
    }

    public String résultat(){
        return nom+ " de " + equipage.getCapitaine() + " a réalisé un
temps de " + tempsPondéré();
    }
}
```

```

    public String getNom() {
        return nom;
    }

    public void enregistrerTemps(float t) {
        temps=t;
    }
    public void affecterEquipage(Equipage equipage) {
        this.equipage = equipage;
    }
    public float getTemps() {
        return temps;
    }
}

public class Equipier {
    private String nom;
    private String spécialité;
    public Equipier(String nom, String spécialité) {
        this.nom = nom;
        this.spécialité = spécialité;
    }
    public String toString() {
        return nom + "(" + spécialité + ")";
    }
}

public class Equipage {
    public static final int NBEQUIPMAX=6;
    private String nom;
    private ArrayList<Equipier> equipiers;
    private Equipier capitaine;
    private int nbEquipiers;

    public Equipage(String nom, Equipier capitaine){
        this.nom=nom;
        this.capitaine=capitaine;
        equipiers=new ArrayList<Equipier> ();
        nbEquipiers=1; //le capitaine
    }

    public Equipier getCapitaine() {
        return capitaine;
    }

    public void ajoutEquipier(Equipier e) {
        equipiers.add(e);
        nbEquipiers++;
    }

    public String toString(){
        String mes="Equipage du capitaine " + capitaine + " :
\n";

        for (int i=0; i<equipiers.size();i++){
            mes+= equipiers.get(i)+"\n";
        }
        return mes;
    }
}

public class Monocoque extends Bateau {
    public static final int COEFMONO=2;

    public Monocoque(String nom) {
        super(nom);
    }
}

```

```

    public float tempsPondéré() {
        return temps*COEFMONO;
    }
    public String toString(){
        return super.toString()+ " (monocoque) " ;
    }
}
public class Multicoque extends Bateau {
    public static final int COEFMULTI=5;

    public Multicoque(String nom) {
        super(nom);
    }

    public float tempsPondéré() {
        return temps*COEFMULTI;
    }
    public String toString(){
        return super.toString()+ " (multicoque)";
    }
}
public class Course {
    private static final int NBMAXBAT=20;
    private String titre;
    private int nbBateaux;
    private ArrayList <Bateau> participants;
    private ArrayList <Bateau> gagnants ;

    public Course( String titre){
        this.titre= titre;
        participants = new ArrayList <Bateau>();
        gagnants = new ArrayList <Bateau>();
        nbBateaux=0;
    }
    public void inscrireBateau(Bateau b, Equipage e){
        if (participants.size()==NBMAXBAT)
            System.out.println("La liste des bateaux est pleine");
        else {
            participants.add(b);
            b.affecterEquipage(e);
            nbBateaux ++;
        }
    }

    private float tempsMin(){
        float min=participants.get(0).getTemps();
        for (Bateau b:participants){
            if (b.getTemps()<min)
                min=b.getTemps();
        }
        return min;
    }
    private void trouverGagnants(){
        for (Bateau b:participants){
            if (b.getTemps()==tempsMin()) {
                gagnants.add(b);
            }
        }
    }
    public String toString(){
        String mes="Liste des participants de la " + titre+"\n";

```

```

        if (participants.size()==0)
            mes+= "Erreur: liste vide";
        else
            for (Bateau b:participants)
                mes+= " - " + b + "\n";
        return mes;
    }

    public void afficherGagnants() {
        String mes="RESULTATS : \n";
        if (participants.size()==0)
            mes+= "Erreur: liste vide";
        else {
            trouverGagnants();
            if (gagnants.size()==1)
                mes+= "Le grand gagnant est " + gagnants.get(0);
            else{
                mes= "Il y a " + gagnants.size() + " gagnants ex-aequo :
";
                for (Bateau b:gagnants)
                    mes+= b.getNom() + " ";
            }
        }
        System.out.println(mes);
    }
}

public class GestionCourse {

    public static void main(String[] args) {
        Monocoque b1=new Monocoque("Java");
        Multicoque b2=new Multicoque("Tango");
        Equipage eq1=new Equipage("FineEquipe", new
Equipier("Jean", "barreur"));
        eq1.ajoutEquipier(new Equipier("Marie", "mousse"));
        Equipage eq2=new Equipage("Solitaire", new
Equipier("Pierre", "barreur"));
        Course c=new Course("Course du vent");
        c.inscrireBateau(b1,eq1);
        c.inscrireBateau(b2,eq2);
        System.out.println(c);
        b1.enregistrerTemps(120);
        b2.enregistrerTemps(150);
        c.afficherGagnants();
    }
}

```

Exercice 8 - Classes et Relations

Définissez les diagrammes de classes modélisant les situations décrites dans les expressions suivantes :

1. Un pays a une capitale
2. Une transaction boursière est un achat ou une vente
3. Une pièce contient des murs
4. Un répertoire contient des fichiers
5. Les feutres et les stylos permettent d'écrire. Ils appartiennent à une personne (leur propriétaire). Ils ont une couleur et une marque mais les feutres ont un bouchon alors que les stylos ont une pointe rétractable.
6. Des employés ont chacun, une fonction, un grade, et peuvent percevoir des primes.

7. Un timbre, une adresse, une enveloppe
8. Des poupées russes
9. Un club de football, des équipes et des joueurs (titulaires et remplaçants)
10. Un document, un paragraphe, une phrase

Exercice 9 - Bateau

Dessinez le diagramme de classes correspondant à la situation suivante :

Un bateau contient des cabines, occupées par des personnes qui effectuent des activités. Les personnes sont ou bien des guides, ou bien des animateurs, ou bien des passagers. Les guides expliquent des visites aux passagers et les animateurs animent des animations pour les passagers.