

**Compte Rendu TP Java**

**Réalisé par : Khadija Kriaa**

**1 IDSD 1**

- La classe parent Accompagnant :
  - ◆ **Classe de base** pour tous les types d'accompagnants.
  - ◆ Contient des informations générales : **nom et prénom**.
  - ◆ Redéfinition de `toString()` pour afficher ses infos.

```

1 package exerc;
2
3 public class accompagnant {
4     private String nom;
5     private String prenom;
6     public accompagnant(String nom, String prenom) {
7         this.nom = nom;
8         this.prenom = prenom;
9     }
10    public String getNom() {
11        return nom;
12    }
13    public String getPrenom() {
14        return prenom;
15    }
16    @Override
17    public String toString() {
18        return "accompagnant:nom=" + nom + ", prenom=" + prenom;
19    }
20
21 }
22

```

- La Classe enseignant :
  - ◆ Hérite de `accompagnant` : a donc aussi nom et prenom.
  - ◆ Ajoute cin et grade.
  - ◆ Redéfinit `toString()` pour ajouter plus d'infos.
  - ◆ a une fonction `getGrade` qui retourne la grade de l'enseignant.

```

1 package exerc;
2
3 public class enseignant extends accompagnant {
4     private int cin;
5     private String grade;
6     public enseignant(String nom, String prenom, int cin, String grade) {
7         super(nom, prenom);
8         this.cin = cin;
9         this.grade = grade;
10    }
11    @Override
12    public String toString() {
13        return "enseignant:"+ super.toString()+"grade: "+ grade + "CIN: "+ cin;
14    }
15    public String getGrade() {
16        return grade;
17    }
18
19 }

```

- La Classe doctorant :
  - ◆ Sous-classe de `accompagnant`.
  - ◆ Ajoute cin et matricule.
  - ◆ Redéfinit aussi `toString()` pour ses propres infos.
  - ◆ a une fonction `getMatricule` qui retourne la matricule du doctorant.

```

package exerc;

public class doctorant extends accompagnant{
    private int cin;
    private int matricule ;
    public doctorant(String nom, String prenom, int cin, int matricule) {
        super(nom, prenom);
        this.cin = cin;
        this.matricule = matricule;
    }
    public int getMatricule() {
        return matricule;
    }
    public String toString(){
        return "doctorant: "+super.toString()+"cin:"+ " " +cin + "matricule: "+matricule;
    }
}

```

- La Classe Conference :
  - ◆ Une conférence a un nom, des frais de participation et une capacité.
  - ◆ listP contient tous les accompagnants inscrits (enseignants, doctorants, accompagnants).
  - ◆ La methode inscrire ajoute un accompagnant à la liste **s'il reste de la place**.
  - ◆ La methode recette :
 

Calcule la recette totale :

    - 30% des frais pour un **enseignant**
    - 70% pour un **doctorant**
    - 100% pour les **autres accompagnants**
  - ◆ La methode toString :
    - Affiche les infos générales + les participants .

```

package exerc;
import java.util.ArrayList;
public class conference {
    private String nom;
    private float frais;
    private int capacite;
    ArrayList<accompagnant>listP= new ArrayList<>(100);
    public conference(String nom, float frais){
        this.nom=nom;
        this.frais=frais;
        this.capacite=0;
    }
    public int getNb(){
        return listP.size();
    }
    public ArrayList<accompagnant> grtlist(){
        return listP;
    }
    public float getfrais(){
        return frais;
    }
    public String toString(){
        System.out.println("conference :"+"nom="+ nom +",frais="+ frais + ",capacite="+ capacite);
        for (accompagnant a: listP){
            if (a instanceof enseignant){
                System.out.println(a.toString());
            }
            else if (a instanceof doctorant){
                System.out.println(a.toString());
            }
            else{
                System.out.println(a.toString());
            }
        }
        return " ";
    }
    public void inscrire(accompagnant A){
        if(listP.size()<100){
            listP.add(A);
            capacite++;
        }
        else{
            System.out.println("la liste est pleine");
        }
    }
    public float recette (){
        float recette=0;
        for (accompagnant a: listP){
            if (a instanceof enseignant){
                recette+=frais*0.3;
            }
            else if (a instanceof doctorant){
                recette+=frais*0.7;
            }
            else{
                recette+=frais;
            }
        }
        return recette;
    }
}

```

- La classe gestionConf :
  - ◆ Crée une conférence.
  - ◆ Crée trois accompagnants (un de chaque type).
  - ◆ Les inscrit à la conférence.
  - ◆ Affiche les infos de la conférence + la recette totale.

```
package exerc;
xerc/src/exerc/accompagnant.java
public class gestionConf {
    public static void main(String[] args) {
        conference C = new conference("conf1", 100);
        accompagnant A = new accompagnant("Ali", "Ben");
        enseignant E = new enseignant("khadija", "kriaa", 114321, "professeur");
        doctorant D = new doctorant("sami", "ghorbel", 1113456, 712);

        C.inscrire(A);
        C.inscrire(E);
        C.inscrire(D);

        System.out.println(C);
        System.out.println("Recette totale: " + C.recette() );
    }
}
```

- L'affichage :

```
conference :nom=conf1 ,frais=100.0,capacite=3
accompagnant:nom=Ali, prenom=Ben
enseignant:accompagnant:nom=khadija, prenom=kriaagrade: professeurCIN: 114321
doctorant: accompagnant:nom=sami, prenom=ghorbelcin: 1113456matricule: 712

Recette totale: 200.0
```