Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

COMPTE RENDU

POO - TP5 : Utilisation des interfaces, classes abstraites et les listes
3e année Cybersécurité - École Supérieure d'Informatique et du Numérique (ESIN)
Collège d'Ingénierie & d'Architecture (CIA)

Étudiant : HATHOUTI Mohammed Taha

Filière : Cybersecurité

Année : 2025/2026

Enseignants : M.NAJIB

Date : 9 novembre 2025

Table des matières

1 Exercice 1 :

1.1 Version Interface :

1.1.1 Interface *ICotisation*

```java
package ex1;

public interface ICotisation {

    double calculerCotisation();
}
```

1.1.2 Classe *EmployeServiceTechnique*

```java
package ex1;

public class EmployeServiceTechnique implements ICotisation{

    private String nom;
    private String prenom;
    private double salaire;

    private static final double TAUX_COTISATION = 0.06;


    public EmployeServiceTechnique(String nom, String prenom,
       double salaire) {
        this.nom = nom;
        this.prenom = prenom;
        this.salaire = salaire;
    }

    public String getNom() {
        return nom;
    }

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

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public double getSalaire() {
        return salaire;
```

```
36      }
37
38      public void setSalaire(double salaire) {
39          this.salaire = salaire;
40      }
41
42      @Override
43      public double calculerCotisation() {
44          // TODO Auto-generated method stub
45          return salaire * TAUX_COTISATION;
46      }
47
48      @Override
49      public String toString() {
50          return "EmployeServiceTechnique [nom = " + nom + ", prenom
                = " + prenom + ", salaire = " + salaire
51                  + ", calculerCotisation() = " + calculerCotisation
                    () + "    ]";
52      }
53
54  }
```

1.1.3 Classe *EmployeServiceAdministratif*

```
1   package ex1;
2
3   public class EmployeServiceAdministratif implements ICotisation{
4
5       private String nom;
6       private String prenom;
7       private double salaire;
8
9       private static final double TAUX_COTISATION = 0.07;
10
11
12      public EmployeServiceAdministratif(String nom, String prenom,
            double salaire) {
13          this.nom = nom;
14          this.prenom = prenom;
15          this.salaire = salaire;
16      }
17
18      public String getNom() {
19          return nom;
20      }
21
22      public void setNom(String nom) {
23          this.nom = nom;
24      }
25
26      public String getPrenom() {
```

```
27          return prenom;
28      }
29
30      public void setPrenom(String prenom) {
31          this.prenom = prenom;
32      }
33
34      public double getSalaire() {
35          return salaire;
36      }
37
38      public void setSalaire(double salaire) {
39          this.salaire = salaire;
40      }
41
42      @Override
43      public double calculerCotisation() {
44          // TODO Auto-generated method stub
45          return salaire * TAUX_COTISATION;
46      }
47
48      @Override
49      public String toString() {
50          return "EmployeServiceAdministratif [nom = " + nom + ",
                prenom = " + prenom + ", salaire = " + salaire
51                  + ", calculerCotisation() = " + calculerCotisation
                        () + "    ]";
52      }
53
54  }
```

1.1.4 Classe *Keyboard*

```
1  package ex1;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.io.InputStreamReader;
7
8  public class Keyboard {
9
10     public static String[] readValues(InputStream in) throws
           IOException {
11         InputStreamReader r = new InputStreamReader(in);
12         BufferedReader br = new BufferedReader(r);
13         String line = br.readLine();
14         String[] values = line.split(" ");
15         return values;
16     }
17 }
```

1.1.5 Classe *MainIterface*

```java
package ex1;

import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.List;

public class MainInterface {
    private static PrintStream ps = System.out;
    private static InputStream is = System.in;
    private static List<ICotisation> employes = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        ps.println("
           ================================================");
        ps.println("|    SYST ME DE GESTION DES COTISATIONS AMELI
              |");
        ps.println("
           ================================================\n");

        String choix;

        ps.println("Op rations disponibles :");
        ps.println(" - ajouter : ajouter un nouvel employ ");
        ps.println(" - afficher : afficher tous les employ s");
        ps.println(" - calculer : calculer les cotisations totales"
           );
        ps.println(" - rechercher : rechercher un employ  par nom"
           );
        ps.println(" - supprimer : supprimer un employ ");
        ps.println(" - quit : quitter le programme");

        do {
            ps.print("\n--> Que voulez-vous faire ? ");
            String[] input = Keyboard.readValues(is);
            choix = input[0].toLowerCase().trim();

            switch (choix) {
            case "ajouter":
                ajouterEmploye();
                break;

            case "afficher":
                afficherEmployes();
                break;

```

```
43              case "calculer":
44                  calculerCotisations();
45                  break;
46
47              case "rechercher":
48                  rechercherEmploye();
49                  break;
50
51              case "supprimer":
52                  supprimerEmploye();
53                  break;
54
55              case "quit":
56                  ps.println("\n  Au revoir !");
57                  break;
58
59              default:
60                  ps.println("\n  Op ration non reconnue !");
61                  break;
62              }
63          } while (!choix.equals("quit"));
64      }
65
66      private static void ajouterEmploye() throws IOException {
67          ps.println("\n=== AJOUTER UN EMPLOY  ===");
68
69          ps.println("\nType d'employ  :");
70          ps.println(" 1 - Service Technique (6%)");
71          ps.println(" 2 - Service Administratif (7%)");
72          ps.print("-->Votre choix : ");
73
74          String[] typeInput = Keyboard.readValues(is);
75          int type = Integer.parseInt(typeInput[0]);
76
77          if (type != 1 && type != 2) {
78              ps.println("  Type invalide !");
79              return;
80          }
81
82          ps.print("\n--> Nom : ");
83          String[] nomInput = Keyboard.readValues(is);
84          String nom = String.join(" ", nomInput);
85
86          ps.print("--> Pr nom : ");
87          String[] prenomInput = Keyboard.readValues(is);
88          String prenom = String.join(" ", prenomInput);
89
90          ps.print("--> Salaire (en    ) : ");
91          String[] salaireInput = Keyboard.readValues(is);
92          double salaire = Double.parseDouble(salaireInput[0]);
93
```

```
94          if (type == 1) {
95              employes.add(new EmployeServiceTechnique(nom, prenom,
                    salaire));
96              ps.println("\n  Employ  Service Technique ajout  !");
97          } else {
98              employes.add(new EmployeServiceAdministratif(nom,
                    prenom, salaire));
99              ps.println("\n  Employ  Service Administratif ajout
                    !");
100         }
101
102         ps.printf("  Cotisation : %.2f    \n", employes.get(
                employes.size() - 1).calculerCotisation());
103     }
104
105     private static void afficherEmployes() {
106         ps.println("\n=== LISTE DES EMPLOY S ===");
107
108         if (employes.isEmpty()) {
109             ps.println("Aucun employ  enregistr .");
110             return;
111         }
112
113         for (int i = 0; i < employes.size(); i++) {
114             ps.println("\n[" + (i + 1) + "] " + employes.get(i));
115         }
116     }
117
118     private static void calculerCotisations() {
119         ps.println("\n=== CALCUL DES COTISATIONS ===");
120
121         if (employes.isEmpty()) {
122             ps.println("Aucun employ  enregistr .");
123             return;
124         }
125
126         double total = 0;
127         for (ICotisation emp : employes) {
128             total += emp.calculerCotisation();
129         }
130
131         ps.printf("\n--> Nombre d'employ s : %d\n", employes.size
                ());
132         ps.printf("--> Total des cotisations : %.2f    \n", total);
133         ps.printf("--> Moyenne par employ  : %.2f    \n", total /
                employes.size());
134     }
135
136     private static void rechercherEmploye() throws IOException {
137         ps.print("\n--> Nom    rechercher : ");
138         String[] nomInput = Keyboard.readValues(is);
```

```
139         String nomRecherche = String.join(" ", nomInput).
               toLowerCase();
140
141         boolean trouve = false;
142         ps.println("\n=== R SULTATS ===");
143
144         for (ICotisation emp : employes) {
145             String empStr = emp.toString().toLowerCase();
146
147             if (empStr.contains(nomRecherche)) {
148                 ps.println(emp);
149                 trouve = true;
150             }
151         }
152
153         if (!trouve) {
154             ps.println("  Aucun employ  trouv  avec ce nom.");
155         }
156     }
157
158     private static void supprimerEmploye() throws IOException {
159         if (employes.isEmpty()) {
160             ps.println("\n  Aucun employ     supprimer.");
161             return;
162         }
163
164         afficherEmployes();
165
166         ps.print("\n--> Num ro de l'employ     supprimer : ");
167         String[] numInput = Keyboard.readValues(is);
168         int num = Integer.parseInt(numInput[0]);
169
170         if (num > 0 && num <= employes.size()) {
171             employes.remove(num - 1);
172             ps.println("\n  Employ  supprim  !");
173         } else {
174             ps.println("\n  Num ro invalide !");
175         }
176     }
177 }
```

1.2 Version Classe Abstraite :

1.2.1 Classe Abstraite *Employe*

```java
package ex1;

public abstract class Employe {

    protected String nom;
    protected String prenom;
    protected double salaire;

    public Employe(String nom, String prenom, double salaire) {
        this.nom = nom;
        this.prenom = prenom;
        this.salaire = salaire;
    }

    public abstract double calculerCotisation();

    public String getNom() {
        return nom;
    }

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

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public double getSalaire() {
        return salaire;
    }

    public void setSalaire(double salaire) {
        this.salaire = salaire;
    }

    @Override
    public String toString() {
        return "Employe [nom = " + nom + ", prenom = " + prenom + "
            , salaire = " + salaire + "   ]";
    }
}
```

1.2.2 Classe *EmpServiceTech*

```java
package ex1;

public class EmpServiceTech extends Employe {

    private static final double TAUX_COTISATION = 0.07;

    public EmpServiceTech(String nom, String prenom, double salaire) {
        super(nom, prenom, salaire);
    }

    @Override
    public double calculerCotisation() {
        return salaire * TAUX_COTISATION;
    }

    @Override
    public String toString() {
        return "EmployeServiceTechnique [nom = " + nom + ", prenom = " + prenom + ", salaire = " + salaire
                + ", calculerCotisation() = " + calculerCotisation() + "    ]";
    }
}
```

1.2.3 Classe *EmpServiceAdmin*

```java
package ex1;

public class EmpServiceAdmin extends Employe {

    private static final double TAUX_COTISATION = 0.07;

    public EmpServiceAdmin(String nom, String prenom, double salaire) {
        super(nom, prenom, salaire);
    }

    @Override
    public double calculerCotisation() {
        return salaire * TAUX_COTISATION;
    }

    @Override
    public String toString() {
        return "EmployeServiceAdministratif [nom = " + nom + ", prenom = " + prenom + ", salaire = " + salaire
                + ", calculerCotisation() = " + calculerCotisation() + "    ]";
```

```
    }
}
```

1.2.4 Classe *Keyboard*

```
package ex1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class Keyboard {

    public static String[] readValues(InputStream in) throws
       IOException {
        InputStreamReader r = new InputStreamReader(in);
        BufferedReader br = new BufferedReader(r);
        String line = br.readLine();
        String[] values = line.split(" ");
        return values;
    }
}
```

1.2.5 Classe *MainAbstract*

```
package ex1;

import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.List;

public class MainAbstract {
    private static PrintStream ps = System.out;
    private static InputStream is = System.in;
    private static List<Employe> employes = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        ps.println("
           ================================================");
        ps.println("|    SYST ME DE GESTION DES COTISATIONS AMELI
              |");
        ps.println("|           Version Classe Abstraite
                    |");
        ps.println("
           ================================================\n");

```

```
20          String choix;
21
22          ps.println("Op rations disponibles :");
23          ps.println(" - ajouter : ajouter un nouvel employ ");
24          ps.println(" - afficher : afficher tous les employ s");
25          ps.println(" - calculer : calculer les cotisations totales"
                );
26          ps.println(" - rechercher : rechercher un employ  par nom"
                );
27          ps.println(" - supprimer : supprimer un employ ");
28          ps.println(" - quit : quitter le programme");
29
30          do {
31              ps.print("\n--> Que voulez-vous faire ? ");
32              String[] input = Keyboard.readValues(is);
33              choix = input[0].toLowerCase().trim();
34
35              switch (choix) {
36              case "ajouter":
37                  ajouterEmploye();
38                  break;
39
40              case "afficher":
41                  afficherEmployes();
42                  break;
43
44              case "calculer":
45                  calculerCotisations();
46                  break;
47
48              case "rechercher":
49                  rechercherEmploye();
50                  break;
51
52              case "supprimer":
53                  supprimerEmploye();
54                  break;
55
56              case "quit":
57                  ps.println("\n  Au revoir !");
58                  break;
59
60              default:
61                  ps.println("\n  Op ration non reconnue !");
62                  break;
63              }
64          } while (!choix.equals("quit"));
65      }
66
67      private static void ajouterEmploye() throws IOException {
68          ps.println("\n=== AJOUTER UN EMPLOY  ===");
```

```
69
70          ps.println("\nType d'employ  :");
71          ps.println(" 1 - Service Technique (6%)");
72          ps.println(" 2 - Service Administratif (7%)");
73          ps.print("--> Votre choix : ");
74
75          String[] typeInput = Keyboard.readValues(is);
76          int type = Integer.parseInt(typeInput[0]);
77
78          if (type != 1 && type != 2) {
79              ps.println("  Type invalide !");
80              return;
81          }
82
83          ps.print("\n--> Nom : ");
84          String[] nomInput = Keyboard.readValues(is);
85          String nom = String.join(" ", nomInput);
86
87          ps.print("--> Pr nom : ");
88          String[] prenomInput = Keyboard.readValues(is);
89          String prenom = String.join(" ", prenomInput);
90
91          ps.print("--> Salaire (en    ) : ");
92          String[] salaireInput = Keyboard.readValues(is);
93          double salaire = Double.parseDouble(salaireInput[0]);
94
95          if (type == 1) {
96              employes.add(new EmpServiceTech(nom, prenom, salaire));
97              ps.println("\n  Employ  Service Technique ajout  !");
98          } else {
99              employes.add(new EmpServiceAdmin(nom, prenom, salaire))
                    ;
100             ps.println("\n  Employ  Service Administratif ajout
                    !");
101         }
102
103         ps.printf("  Cotisation : %.2f    \n", employes.get(
                employes.size() - 1).calculerCotisation());
104     }
105
106     private static void afficherEmployes() {
107         ps.println("\n=== LISTE DES EMPLOY S ===");
108
109         if (employes.isEmpty()) {
110             ps.println("Aucun employ  enregistr .");
111             return;
112         }
113
114         for (int i = 0; i < employes.size(); i++) {
115             ps.println("\n[" + (i + 1) + "] " + employes.get(i));
116         }
```

```
117     }
118
119     private static void calculerCotisations() {
120         ps.println("\n=== CALCUL DES COTISATIONS ===");
121
122         if (employes.isEmpty()) {
123             ps.println("Aucun employ  enregistr .");
124             return;
125         }
126
127         double total = 0;
128         double totalSalaires = 0;
129
130         for (Employe emp : employes) {
131             total += emp.calculerCotisation();
132             totalSalaires += emp.getSalaire();
133         }
134
135         ps.printf("\n--> Nombre d'employ s : %d\n", employes.size
                ());
136         ps.printf("--> Total des salaires : %.2f    \n",
                totalSalaires);
137         ps.printf("--> Total des cotisations : %.2f    \n", total);
138         ps.printf("--> Moyenne par employ  : %.2f    \n", total /
                employes.size());
139     }
140
141     private static void rechercherEmploye() throws IOException {
142         ps.print("\n--> Nom    rechercher : ");
143         String[] nomInput = Keyboard.readValues(is);
144         String nomRecherche = String.join(" ", nomInput).
                toLowerCase();
145
146         boolean trouve = false;
147         ps.println("\n=== R SULTATS ===");
148
149         for (Employe emp : employes) {
150             String empStr = emp.toString().toLowerCase();
151
152             if (empStr.contains(nomRecherche)) {
153                 ps.println(emp);
154                 trouve = true;
155             }
156         }
157
158         if (!trouve) {
159             ps.println("  Aucun employ  trouv  avec ce nom.");
160         }
161     }
162
163     private static void supprimerEmploye() throws IOException {
```

```
164         if (employes.isEmpty()) {
165             ps.println("\n  Aucun employ     supprimer.");
166             return;
167         }
168 
169         afficherEmployes();
170 
171         ps.print("\n--> Num ro de l'employ     supprimer : ");
172         String[] numInput = Keyboard.readValues(is);
173         int num = Integer.parseInt(numInput[0]);
174 
175         if (num > 0 && num <= employes.size()) {
176             employes.remove(num - 1);
177             ps.println("\n  Employ  supprim  !");
178         } else {
179             ps.println("\n  Num ro invalide !");
180         }
181     }
182 }
```

2 Exercice 2 :

2.1 Classe *Etudiant*

```
1  package ex2;
2  
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.io.PrintStream;
6  
7  public class Etudiant {
8      public static PrintStream ps = System.out;
9  
10     public int code;
11     public String nom, prenom, filiere;
12 
13     public Etudiant() {
14         super();
15 
16     }
17 
18     public Etudiant(int code, String nom, String prenom, String
       filiere) {
19         super();
20         this.code = code;
21         this.nom = nom;
22         this.prenom = prenom;
23         this.filiere = filiere;
24     }
25 
```

```
26      @Override
27      public String toString() {
28          return "Etudiant [code = " + code + ", nom = " + nom + ",
                prenom = " + prenom + ", filiere = " + filiere + "]";
29      }
30
31      public static boolean recherche(List<Etudiant> liste, int code)
            {
32
33          for (Etudiant etudiant : liste) {
34              if (etudiant.code == code) {
35                  return true;
36              }
37          }
38          return false;
39      }
40
41      public static void affichage(List<Etudiant> liste) {
42          if (liste.isEmpty()) {
43              ps.println("La liste est vide");
44              return;
45          }
46
47          ps.println("\n===   Liste des Etudiants   ===");
48          ps.println("Nombre total : " + liste.size());
49
50          for (int i = 0; i < liste.size(); i++) {
51              ps.println("[ " + (i + 1) + " ] " + liste.get(i));
52          }
53      }
54
55      @SuppressWarnings("rawtypes")
56      public static List trierListe(List<Etudiant> liste) {
57          List<Etudiant> liste_triee = new ArrayList<>(liste);
58
59          liste_triee.sort((etudiant1, etudiant2) -> Integer.compare(
                etudiant1.code, etudiant2.code));
60
61          return liste_triee;
62      }
63
64  }
```

2.2 Classe *Keyboard*

```
1  package ex2;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStream;
```

```java
import java.io.InputStreamReader;

public class Keyboard {

    public static String[] readValues(InputStream in) throws
       IOException {
        InputStreamReader r = new InputStreamReader(in);
        BufferedReader br = new BufferedReader(r);
        String line = br.readLine();
        String[] values = line.split(" ");
        return values;
    }
}
```

2.3 Classe *Main*

```java
package ex2;

import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.List;

public class Main {
    private static PrintStream ps = System.out;
    private static InputStream is = System.in;
    private static List<Etudiant> listeEtudiants = new ArrayList
       <>();

    public static void main(String[] args) throws IOException {
        ps.println("
           ================================================");
        ps.println("|     SYST ME DE GESTION DES   TUDIANTS   - UIR
                |");
        ps.println("
           ================================================\n");

        String choix = "";

        ps.println("\n\n========== MENU PRINCIPAL ==========");
        ps.println("Op rations disponibles :");
        ps.println(" - ajouter : ajouter un nouvel   tudiant ");
        ps.println(" - afficher : afficher tous les   tudiants ");
        ps.println(" - rechercher : rechercher un   tudiant   par
           code");
        ps.println(" - trier : trier la liste par code");
        ps.println(" - supprimer : supprimer un   tudiant ");
        ps.println(" - vider : supprimer tous les   tudiants ");
        ps.println(" - quit : quitter le programme");
```

```
30          ps.println("====================================");
31
32          do {
33              boolean choixValide = false;
34
35              do {
36                  ps.print("\n--> Que voulez-vous faire ? ");
37
38                  try {
39                      String[] input = Keyboard.readValues(is);
40                      choix = input[0].toLowerCase().trim();
41
42                      if (estChoixValide(choix)) {
43                          choixValide = true;
44                      } else {
45                          ps.println("    Op ration non reconnue !")
                                ;
46                          ps.println("  Veuillez choisir parmi :
                                ajouter, afficher, rechercher, trier,
                                supprimer, vider, demo, quit");
47                      }
48
49                  } catch (Exception e) {
50                      ps.println("    Erreur de saisie ! R essayez."
                            );
51                  }
52
53              } while (!choixValide);
54
55              switch (choix) {
56                  case "ajouter":
57                      ajouterEtudiant();
58                      break;
59
60                  case "afficher":
61                      afficherEtudiants();
62                      break;
63
64                  case "rechercher":
65                      rechercherEtudiant();
66                      break;
67
68                  case "trier":
69                      trierEtudiants();
70                      break;
71
72                  case "supprimer":
73                      supprimerEtudiant();
74                      break;
75
76                  case "vider":
```

```
77                      viderListe();
78                      break;
79
80                  case "quit":
81                      ps.println("\n  Au revoir !");
82                      break;
83              }
84
85          } while (!choix.equals("quit"));
86      }
87
88      private static boolean estChoixValide(String choix) {
89          String[] choixValides = {"ajouter", "afficher", "rechercher
               ", "trier",
90                                   "supprimer", "vider", "demo", "
                                      quit"};
91
92          for (String c : choixValides) {
93              if (c.equals(choix)) {
94                  return true;
95              }
96          }
97          return false;
98      }
99
100     private static void ajouterEtudiant() throws IOException {
101         ps.println("\n=== AJOUTER UN   TUDIANT   ===");
102
103         int code = 0;
104         String nom = "";
105         String prenom = "";
106         String filiere = "";
107         boolean donneesValides = false;
108
109         do {
110             try {
111                 ps.print("\n--> Code   tudiant   : ");
112                 String[] codeInput = Keyboard.readValues(is);
113                 code = Integer.parseInt(codeInput[0]);
114
115                 ps.print("--> Nom : ");
116                 String[] nomInput = Keyboard.readValues(is);
117                 nom = String.join(" ", nomInput);
118
119                 ps.print("--> Pr nom : ");
120                 String[] prenomInput = Keyboard.readValues(is);
121                 prenom = String.join(" ", prenomInput);
122
123                 ps.print("--> Fili re : ");
124                 String[] filiereInput = Keyboard.readValues(is);
125                 filiere = String.join(" ", filiereInput);
```

```
126
127                 if (code <= 0) {
128                     ps.println("  Erreur : le code doit  tre  un
                           nombre positif");
129                     continue;
130                 }
131
132                 if (Etudiant.recherche(listeEtudiants, code)) {
133                     ps.println("  Erreur : un  tudiant  avec ce
                           code existe d j  !");
134                     continue;
135                 }
136
137                 if (nom.trim().isEmpty() || prenom.trim().isEmpty()
                        || filiere.trim().isEmpty()) {
138                     ps.println("  Erreur : tous les champs doivent
                            tre  remplis");
139                     continue;
140                 }
141
142                 donneesValides = true;
143
144             } catch (NumberFormatException e) {
145                 ps.println("  Erreur : entrez un nombre valide pour
                        le code !");
146             }
147         } while (!donneesValides);
148
149         listeEtudiants.add(new Etudiant(code, nom, prenom, filiere)
               );
150
151         ps.println("\n     tudiant  ajout  avec succ s !");
152         ps.println("  Code : " + code + " | Nom : " + nom + " " +
               prenom + " | Fili re : " + filiere);
153     }
154
155     private static void afficherEtudiants() {
156         ps.println("\n=== AFFICHAGE (M THODE STATIQUE Q2) ===");
157
158         Etudiant.affichage(listeEtudiants);
159     }
160
161     private static void rechercherEtudiant() throws IOException {
162         ps.println("\n=== RECHERCHER PAR CODE (M THODE STATIQUE Q2
               ) ===");
163
164         if (listeEtudiants.isEmpty()) {
165             ps.println("  Aucun  tudiant  dans la liste.");
166             return;
167         }
168
```

```
169         ps.print("\n--> Code    rechercher : ");
170         try {
171             String[] codeInput = Keyboard.readValues(is);
172             int code = Integer.parseInt(codeInput[0]);
173 
174             boolean trouve = Etudiant.recherche(listeEtudiants,
                    code);
175 
176             if (trouve) {
177                 ps.println("\n      tudiant   avec le code " + code +
                        " trouv  !");
178 
179                 for (Etudiant e : listeEtudiants) {
180                     if (e.code == code) {
181                         ps.println("  " + e);
182                         break;
183                     }
184                 }
185             } else {
186                 ps.println("\n    Aucun   tudiant   trouv   avec le
                        code " + code);
187             }
188 
189         } catch (NumberFormatException e) {
190             ps.println("  Erreur : entrez un nombre valide !");
191         }
192     }
193 
194     @SuppressWarnings("unchecked")
195     private static void trierEtudiants() {
196         ps.println("\n=== TRIER LA LISTE (M THODE STATIQUE Q2) ===
                ");
197 
198         if (listeEtudiants.isEmpty()) {
199             ps.println("  Aucun   tudiant   dans la liste.");
200             return;
201         }
202 
203         ps.println("\nListe AVANT tri :");
204         for (int i = 0; i < listeEtudiants.size(); i++) {
205             Etudiant e = listeEtudiants.get(i);
206             ps.println("  [" + (i+1) + "] Code " + e.code + " : " +
                    e.nom + " " + e.prenom);
207         }
208 
209         List<Etudiant> listeTriee = Etudiant.trierListe(
                listeEtudiants);
210 
211         ps.println("\nListe APR S tri (ordre croissant des codes)
                :");
212         for (int i = 0; i < listeTriee.size(); i++) {
```

```
213             Etudiant e = listeTriee.get(i);
214             ps.println("  [" + (i+1) + "] Code " + e.code + " : " +
                    e.nom + " " + e.prenom);
215         }
216
217         ps.println("\n    Liste tri e ! (l'originale reste
                inchang e)");
218
219         ps.print("\n--> Voulez-vous remplacer la liste originale
                par la version tri e ? (oui/non) : ");
220
221         try {
222             String[] reponse = Keyboard.readValues(is);
223             if (reponse[0].equalsIgnoreCase("oui")) {
224                 listeEtudiants = listeTriee;
225                 ps.println("    Liste originale remplac e !");
226             }
227         } catch (IOException e) {
228         }
229     }
230
231     private static void supprimerEtudiant() throws IOException {
232         if (listeEtudiants.isEmpty()) {
233             ps.println("\n  Aucun  tudiant     supprimer.");
234             return;
235         }
236
237         Etudiant.affichage(listeEtudiants);
238
239         ps.print("\n--> Num ro de l' tudiant     supprimer : ");
240         try {
241             String[] numInput = Keyboard.readValues(is);
242             int num = Integer.parseInt(numInput[0]);
243
244             if (num > 0 && num <= listeEtudiants.size()) {
245                 Etudiant supprime = listeEtudiants.get(num - 1);
246                 listeEtudiants.remove(num - 1);
247                 ps.println("\n     tudiant  supprim  : " +
                        supprime.nom + " " + supprime.prenom);
248             } else {
249                 ps.println("\n    Num ro invalide !");
250             }
251         } catch (NumberFormatException e) {
252             ps.println("  Erreur : entrez un nombre valide !");
253         }
254     }
255
256     private static void viderListe() {
257         if (listeEtudiants.isEmpty()) {
258             ps.println("\n  La liste est d  j   vide.");
259             return;
```

```
260             }
261
262             int taille = listeEtudiants.size();
263             listeEtudiants.clear();
264
265             ps.println("\n    Liste vid e ! (" + taille + "   tudiant  (
                    s) supprim (s))");
266             ps.println("  Taille actuelle : " + listeEtudiants.size());
267         }
268 }
```