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

COMPTE RENDU

POO - TP5 : Utilisation des interfaces, classes abstraites et les listes
(Partie 2)
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 : 14 novembre 2025

Table des matières

1 Exercice 2 :

1.1 Classe *Etudiant*

```java
package ex2;

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

public class Etudiant {
    public static PrintStream ps = System.out;

    public int code;
    public String nom, prenom, filiere;

    public Etudiant() {
        super();

    }

    public Etudiant(int code, String nom, String prenom, String
       filiere) {
        super();
        this.code = code;
        this.nom = nom;
        this.prenom = prenom;
        this.filiere = filiere;
    }

    @Override
    public String toString() {
        return "Etudiant [code = " + code + ", nom = " + nom + ",
           prenom = " + prenom + ", filiere = " + filiere + "]";
    }

    public static boolean recherche(List<Etudiant> liste, int code)
        {

        for (Etudiant etudiant : liste) {
            if (etudiant.code == code) {
                return true;
            }
        }
        return false;
    }

    public static void affichage(List<Etudiant> liste) {
        if (liste.isEmpty()) {
            ps.println("La liste est vide");
            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      public static void supprimerEtudiant(List<Etudiant> l, int code
            ) {
65          boolean supprime = false;
66
67          for (int i = 0; i < l.size(); i++) {
68              if (l.get(i).code == code) {
69                  Etudiant etudiantSupprime = l.get(i);
70                  l.remove(i);
71                  ps.println("   tudiant   supprim   : " +
                        etudiantSupprime);
72                  supprime = true;
73                  break;
74              }
75          }
76
77          if (!supprime) {
78              ps.println("  Aucun   tudiant   trouv   avec le code " +
                    code);
79          }
80      }
81
82      @SuppressWarnings("rawtypes")
83      public static List listeSpecialite(List<Etudiant> l, String spe
            ) {
84          List<Etudiant> listeFiliere = new ArrayList<>();
85
86          for(Etudiant etudiant : l) {
87              if (etudiant.filiere.equalsIgnoreCase(spe)) {
88                  listeFiliere.add(etudiant);
89              }
90          }
```

```
91          return listeFiliere;
92      }
93
94  }
```

1.2 Classe *Keyboard*

```
1  package ex2;
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 }
```

1.3 Classe *Main*

```
1  package ex2;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.PrintStream;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  public class Main {
10     private static PrintStream ps = System.out;
11     private static InputStream is = System.in;
12     private static List<Etudiant> listeEtudiants = new ArrayList
           <>();
13
14     public static void main(String[] args) throws IOException {
15         ps.println("
               ================================================");
16         ps.println("|      SYST ME DE GESTION DES   TUDIANTS   - UIR
                   |");
17         ps.println("
               ================================================\n");
```

```
18
19          String choix = "";
20
21          ps.println("\n\n========== MENU PRINCIPAL ==========");
22          ps.println("Op rations disponibles :");
23          ps.println(" - ajouter : ajouter un nouvel  tudiant ");
24          ps.println(" - afficher : afficher tous les  tudiants ");
25          ps.println(" - rechercher : rechercher un  tudiant  par
                code");
26          ps.println(" - trier : trier la liste par code");
27          ps.println(" - supprimer : supprimer un  tudiant ");
28          ps.println(" - filtrer : filtrer par fili re/sp cialit "
                );
29          ps.println(" - vider : supprimer tous les  tudiants ");
30          ps.println(" - quit : quitter le programme");
31          ps.println("====================================");
32
33          do {
34              boolean choixValide = false;
35
36              do {
37                  ps.print("\n--> Que voulez-vous faire ? ");
38
39                  try {
40                      String[] input = Keyboard.readValues(is);
41                      choix = input[0].toLowerCase().trim();
42
43                      if (estChoixValide(choix)) {
44                          choixValide = true;
45                      } else {
46                          ps.println("    Op ration non reconnue !")
                                ;
47                          ps.println("  Veuillez choisir parmi :
                                ajouter, afficher, rechercher, trier,
                                supprimer, filtrer, vider, quit");
48                      }
49
50                  } catch (Exception e) {
51                      ps.println("    Erreur de saisie ! R essayez."
                            );
52                  }
53
54              } while (!choixValide);
55
56              switch (choix) {
57                  case "ajouter":
58                      ajouterEtudiant();
59                      break;
60
61                  case "afficher":
62                      afficherEtudiants();
```

```
63                      break;
64
65                  case "rechercher":
66                      rechercherEtudiant();
67                      break;
68
69                  case "trier":
70                      trierEtudiants();
71                      break;
72
73                  case "supprimer":
74                      supprimerEtudiant();
75                      break;
76
77                  case "filtrer":
78                      filtrerParSpecialite();
79                      break;
80
81                  case "vider":
82                      viderListe();
83                      break;
84
85                  case "quit":
86                      ps.println("\n  Au revoir !");
87                      break;
88              }
89
90          } while (!choix.equals("quit"));
91      }
92
93      private static boolean estChoixValide(String choix) {
94          String[] choixValides = {"ajouter", "afficher", "rechercher
               ", "trier",
95                                   "supprimer", "filtrer", "vider", "
                                      quit"};
96
97          for (String c : choixValides) {
98              if (c.equals(choix)) {
99                  return true;
100             }
101         }
102         return false;
103     }
104
105     private static void ajouterEtudiant() throws IOException {
106         ps.println("\n=== AJOUTER UN   TUDIANT   ===");
107
108         int code = 0;
109         String nom = "";
110         String prenom = "";
111         String filiere = "";
```

```
112         boolean donneesValides = false;
113
114         do {
115             try {
116                 ps.print("\n--> Code  tudiant  : ");
117                 String[] codeInput = Keyboard.readValues(is);
118                 code = Integer.parseInt(codeInput[0]);
119
120                 ps.print("--> Nom : ");
121                 String[] nomInput = Keyboard.readValues(is);
122                 nom = String.join(" ", nomInput);
123
124                 ps.print("--> Pr nom : ");
125                 String[] prenomInput = Keyboard.readValues(is);
126                 prenom = String.join(" ", prenomInput);
127
128                 ps.print("--> Fili re : ");
129                 String[] filiereInput = Keyboard.readValues(is);
130                 filiere = String.join(" ", filiereInput);
131
132                 if (code <= 0) {
133                     ps.println("  Erreur : le code doit  tre  un
                        nombre positif");
134                     continue;
135                 }
136
137                 if (Etudiant.recherche(listeEtudiants, code)) {
138                     ps.println("  Erreur : un  tudiant  avec ce
                        code existe d j  !");
139                     continue;
140                 }
141
142                 if (nom.trim().isEmpty() || prenom.trim().isEmpty()
                        || filiere.trim().isEmpty()) {
143                     ps.println("  Erreur : tous les champs doivent
                            tre  remplis");
144                     continue;
145                 }
146
147                 donneesValides = true;
148
149             } catch (NumberFormatException e) {
150                 ps.println("  Erreur : entrez un nombre valide pour
                        le code !");
151             }
152         } while (!donneesValides);
153
154         listeEtudiants.add(new Etudiant(code, nom, prenom, filiere)
               );
155
156         ps.println("\n     tudiant  ajout  avec succ s !");
```

```
157         ps.println("  Code : " + code + " | Nom : " + nom + " " +
                prenom + " | Fili re : " + filiere);
158     }
159
160     private static void afficherEtudiants() {
161         ps.println("\n=== AFFICHAGE (M THODE STATIQUE Q2) ===");
162
163         Etudiant.affichage(listeEtudiants);
164     }
165
166     private static void rechercherEtudiant() throws IOException {
167         ps.println("\n=== RECHERCHER PAR CODE ===");
168
169         if (listeEtudiants.isEmpty()) {
170             ps.println("  Aucun  tudiant  dans la liste.");
171             return;
172         }
173
174         ps.print("\n--> Code  rechercher : ");
175         try {
176             String[] codeInput = Keyboard.readValues(is);
177             int code = Integer.parseInt(codeInput[0]);
178
179             boolean trouve = Etudiant.recherche(listeEtudiants,
                    code);
180
181             if (trouve) {
182                 ps.println("\n  tudiant  avec le code " + code +
                        " trouv  !");
183
184                 for (Etudiant e : listeEtudiants) {
185                     if (e.code == code) {
186                         ps.println("  " + e);
187                         break;
188                     }
189                 }
190             } else {
191                 ps.println("\n  Aucun  tudiant  trouv  avec le
                        code " + code);
192             }
193
194         } catch (NumberFormatException e) {
195             ps.println("  Erreur : entrez un nombre valide !");
196         }
197     }
198
199     @SuppressWarnings("unchecked")
200     private static void trierEtudiants() {
201         ps.println("\n=== TRIER LA LISTE ===");
202
203         if (listeEtudiants.isEmpty()) {
```

```
204             ps.println("  Aucun   tudiant   dans la liste.");
205             return;
206         }
207 
208         ps.println("\nListe AVANT tri :");
209         for (int i = 0; i < listeEtudiants.size(); i++) {
210             Etudiant e = listeEtudiants.get(i);
211             ps.println("  [" + (i+1) + "] Code " + e.code + " : " +
                    e.nom + " " + e.prenom);
212         }
213 
214         List<Etudiant> listeTriee = Etudiant.trierListe(
               listeEtudiants);
215 
216         ps.println("\nListe APR S tri (ordre croissant des codes)
               :");
217         for (int i = 0; i < listeTriee.size(); i++) {
218             Etudiant e = listeTriee.get(i);
219             ps.println("  [" + (i+1) + "] Code " + e.code + " : " +
                    e.nom + " " + e.prenom);
220         }
221 
222         ps.println("\n    Liste tri e ! (l'originale reste
               inchang e)");
223 
224         ps.print("\n--> Voulez-vous remplacer la liste originale
               par la version tri e ? (oui/non) : ");
225 
226         try {
227             String[] reponse = Keyboard.readValues(is);
228             if (reponse[0].equalsIgnoreCase("oui")) {
229                 listeEtudiants = listeTriee;
230                 ps.println("    Liste originale remplac e !");
231             }
232         } catch (IOException e) {
233         }
234     }
235 
236     private static void supprimerEtudiant() throws IOException {
237         ps.println("\n=== SUPPRIMER UN   TUDIANT   ===");
238 
239         if (listeEtudiants.isEmpty()) {
240             ps.println("  Aucun   tudiant      supprimer.");
241             return;
242         }
243 
244         Etudiant.affichage(listeEtudiants);
245 
246         ps.print("\n--> Code de l' tudiant      supprimer : ");
247         try {
248             String[] codeInput = Keyboard.readValues(is);
```

```
249             int code = Integer.parseInt(codeInput[0]);
250
251             Etudiant.supprimerEtudiant(listeEtudiants, code);
252
253         } catch (NumberFormatException e) {
254             ps.println("  Erreur : entrez un nombre valide !");
255         }
256     }
257
258     @SuppressWarnings("unchecked")
259     private static void filtrerParSpecialite() throws IOException {
260         ps.println("\n=== FILTRER PAR FILI RE ===");
261
262         if (listeEtudiants.isEmpty()) {
263             ps.println("  Aucun  tudiant  dans la liste.");
264             return;
265         }
266
267         ps.print("\n--> Nom de la fili re/sp cialit  : ");
268         String[] filiereInput = Keyboard.readValues(is);
269         String filiere = String.join(" ", filiereInput);
270
271         List<Etudiant> listeFiltre = Etudiant.listeSpecialite(
                listeEtudiants, filiere);
272
273         if (listeFiltre.isEmpty()) {
274             ps.println("\n  Aucun  tudiant  trouv  dans la
                    fili re : " + filiere);
275         } else {
276             ps.println("\n===  tudiants  en " + filiere + " ===");
277             ps.println("Nombre total : " + listeFiltre.size());
278
279             for (int i = 0; i < listeFiltre.size(); i++) {
280                 ps.println("[ " + (i + 1) + " ] " + listeFiltre.get
                        (i));
281             }
282         }
283     }
284
285     private static void viderListe() {
286         if (listeEtudiants.isEmpty()) {
287             ps.println("\n  La liste est d  j  vide.");
288             return;
289         }
290
291         int taille = listeEtudiants.size();
292         listeEtudiants.clear();
293
294         ps.println("\n    Liste vid e ! (" + taille + "  tudiant (
                s) supprim (s))");
295         ps.println("  Taille actuelle : " + listeEtudiants.size());
```

```
296         }
297 }
```