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

COMPTE RENDU

POO - TP3 : Manipulation des classes en JAVA
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 : 17 octobre 2025

Table des matières

1 Partie 1 :

1.1 Classe *CompteBancaire*

```java
package ex2;

public class CompteBancaire {
    private int id_client;
    private String nom_client;
    private int solde_client;

    public CompteBancaire(int id_client, String nom_client, int
       solde_client) {
        this.id_client = id_client;
        this.nom_client = nom_client;
        this.solde_client = solde_client;
    }

    public int getId_client() {
        return id_client;
    }

    public void setId_client(int id_client) {
        this.id_client = id_client;
    }

    public String getNom_client() {
        return nom_client;
    }

    public void setNom_client(String nom_client) {
        this.nom_client = nom_client;
    }

    public int getSolde_client() {
        return solde_client;
    }

    public void setSolde_client(int solde_client) {
        this.solde_client = solde_client;
    }

    @Override
    public String toString() {
        return "CompteBancaire [id_client = " + id_client + ",
           nom_client = " + nom_client + ", solde_client = "
                + solde_client + "]";
    }

    public boolean retrait(int montant) {
        if (montant > 0 && montant <= solde_client) {
```

```
            solde_client -= montant;
            return true;
        }
        return false;
    }

    public void depot(int montant) {
        if (montant > 0) {
            solde_client += montant;
        }
    }
}
```

1.2 Classe *Banque*

```
package ex2;

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

@SuppressWarnings("unused")
public class Banque {
    PrintStream ps = System.out;
    private int id_banque;
    private String nom_banque;
    private ArrayList<CompteBancaire> tblCmp;

    public Banque(int id_banque, String nom_banque) {
        this.id_banque = id_banque;
        this.nom_banque = nom_banque;
        this.tblCmp = new ArrayList<CompteBancaire>();
    }

    @Override
    public String toString() {
        return "Banque [ps=" + ps + ", id_banque=" + id_banque +
           ", nom_banque=" + nom_banque + ", tblCmp=" + tblCmp
                + "]";
    }

    public void creerCompte(int id, String nom, int solde) {
        tblCmp.add(new CompteBancaire(id, nom, solde));
        ps.println("Compte  c r    avec  succ s  !");
    }

    private CompteBancaire rechercherCompte(int id) {
        for (CompteBancaire compte : tblCmp) {
            if(compte.getId_client() == id) {
                return compte;
```

```
35              }
36          }
37
38          return null;
39      }
40
41      public void virement(int idSource, int idDest, int montant) {
42          CompteBancaire source = rechercherCompte(idSource);
43          CompteBancaire dest = rechercherCompte(idDest);
44
45          if(source == null) {
46              ps.println("Compte source introuvable !");
47              return;
48          }
49
50          if(dest == null) {
51              ps.println("Compte destination introuvable !");
52              return;
53          }
54
55          if (source.retrait(montant)) {
56              dest.depot(montant);
57              ps.println("Virement d'un montant de " + montant + "
                        a   t   effectu  !");
58          } else {
59              ps.println("Solde insuffisant pour effectuer votre
                    op ration !");
60          }
61      }
62
63      public void retraitArgent(int id, int montant) {
64          CompteBancaire compte = rechercherCompte(id);
65
66          if (compte == null) {
67              ps.println("Compte introuvable !");
68              return;
69          }
70
71          if (compte.retrait(montant)) {
72              ps.println("Retrait d'un montant de " + montant + "
                        a   t   effectu  !");
73              ps.println("Actualisation de votre solde !");
74              ps.println("Nouveau Solde : " + compte.
                    getSolde_client() + "   ");
75          } else {
76              ps.println("Solde insuffisant pour effectuer votre
                    op ration !");
77          }
78      }
79
80      public void supprimerCompte(int id) {
```

```
81          CompteBancaire compte = rechercherCompte(id);
82
83          if (compte != null) {
84              tblCmp.remove(compte);
85              ps.println("Compte supprim  !");
86          } else {
87              ps.println("Compte introuvable !");
88          }
89      }
90
91      public void afficherTousLesComptes() {
92          if (tblCmp.isEmpty()) {
93              ps.println("Aucun compte bancaire.");
94              return;
95          }
96          ps.println("\n========== Liste des comptes ==========");
97          for (CompteBancaire compte : tblCmp) {
98              ps.println(compte.toString());
99          }
100         ps.println("========================================\n");
101     }
102
103     public void consulterSolde(int id) {
104         CompteBancaire compte = rechercherCompte(id);
105
106         if (compte != null) {
107             ps.println("Solde du compte " + id + " : " + compte.
                    getSolde_client() + "    ");
108         } else {
109             ps.println("Compte introuvable !");
110         }
111     }
112
113 }
```

2 Partie 2 :

2.1 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 {
```

```
        InputStreamReader r = new InputStreamReader(in);
        BufferedReader br = new BufferedReader(r);
        String line = br.readLine();
        String[] values = line.split(" ");
        return values;
    }
}
```

2.2 Classe *Main*

```
package ex2;

import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;

public class Main {
    private static PrintStream ps = System.out;
    private static InputStream is = System.in;
    private static Banque banque;

    public static void main(String[] args) throws IOException {
        // Authentification
        if (!authentifier()) {
            ps.println("\nNombre maximum de tentatives atteint.
               Programme termine.");
            return;
        }

        // Initialisation de la banque
        banque = new Banque(1, "UIR Bank");

        String choix;

        ps.println("\n=== SYSTEME DE GESTION BANCAIRE ===");
        ps.println("Operations disponibles :");
        ps.println(" - creer : creer un nouveau compte bancaire")
           ;
        ps.println(" - afficher : afficher tous les comptes
           bancaires");
        ps.println(" - supprimer : supprimer un compte bancaire")
           ;
        ps.println(" - virement : effectuer un virement entre
           deux comptes");
        ps.println(" - retrait : retrait d'argent d'un compte");
        ps.println(" - consulter : consulter le solde d'un compte
           ");
        ps.println(" - quit : quitter le programme");

        do {
```

```
35              ps.print("\nQue voulez-vous faire ? ");
36              String[] input = Keyboard.readValues(is);
37              String ligne = String.join(" ", input).toLowerCase().
                   trim();
38              choix = ligne.split("\\s+")[0];
39
40              switch (choix) {
41              case "creer":
42                  int idCompte = 0;
43                  String nomClient = "";
44                  int soldeInitial = 0;
45                  boolean donneesValides = false;
46
47                  do {
48                      ps.print("\nEntrez l'ID du compte : ");
49                      String[] idInput = Keyboard.readValues(is);
50
51                      ps.print("Entrez le nom du client : ");
52                      String[] nomInput = Keyboard.readValues(is);
53                      nomClient = String.join(" ", nomInput);
54
55                      ps.print("Entrez le solde initial : ");
56                      String[] soldeInput = Keyboard.readValues(is)
                           ;
57
58                      try {
59                          idCompte = Integer.parseInt(idInput[0]);
60                          soldeInitial = Integer.parseInt(
                               soldeInput[0]);
61
62                          if (idCompte <= 0) {
63                              ps.println("Erreur: l'ID doit etre un
                                   nombre positif");
64                              continue;
65                          }
66
67                          if (nomClient.trim().isEmpty()) {
68                              ps.println("Erreur: le nom du client
                                   ne peut pas etre vide");
69                              continue;
70                          }
71
72                          if (soldeInitial < 0) {
73                              ps.println("Erreur: le solde initial
                                   ne peut pas etre negatif");
74                              continue;
75                          }
76
77                          donneesValides = true;
78                      } catch (NumberFormatException e) {
```

```
79                      ps.println("Erreur: entrez uniquement des
                            nombres pour l'ID et le solde !");
80                  }
81              } while (!donneesValides);
82
83              ps.println("\n========== RESULTAT ==========");
84              banque.creerCompte(idCompte, nomClient,
                    soldeInitial);
85              ps.println("==============================");
86              break;
87
88          case "afficher":
89              ps.println("\n========== RESULTAT ==========");
90              banque.afficherTousLesComptes();
91              ps.println("==============================");
92              break;
93
94          case "supprimer":
95              int idSupprimer = 0;
96              boolean idValideSupp = false;
97
98              do {
99                  ps.print("\nEntrez l'ID du compte a supprimer
                        : ");
100                 String[] idInput = Keyboard.readValues(is);
101
102                 try {
103                     idSupprimer = Integer.parseInt(idInput
                            [0]);
104                     if (idSupprimer > 0) {
105                         idValideSupp = true;
106                     } else {
107                         ps.println("Erreur: l'ID doit etre un
                                nombre positif");
108                     }
109                 } catch (NumberFormatException e) {
110                     ps.println("Erreur: entrez uniquement des
                            nombres !");
111                 }
112             } while (!idValideSupp);
113
114             ps.println("\n========== RESULTAT ==========");
115             banque.supprimerCompte(idSupprimer);
116             ps.println("==============================");
117             break;
118
119         case "virement":
120             int idSource = 0, idDest = 0, montantVirement =
                    0;
121             boolean virementValide = false;
122
```

```
123                 do {
124                     ps.print("\nEntrez l'ID du compte source : ")
                           ;
125                     String[] sourceInput = Keyboard.readValues(is
                           );
126 
127                     ps.print("Entrez l'ID du compte destination :
                            ");
128                     String[] destInput = Keyboard.readValues(is);
129 
130                     ps.print("Entrez le montant du virement : ");
131                     String[] montantInput = Keyboard.readValues(
                           is);
132 
133                     try {
134                         idSource = Integer.parseInt(sourceInput
                               [0]);
135                         idDest = Integer.parseInt(destInput[0]);
136                         montantVirement = Integer.parseInt(
                               montantInput[0]);
137 
138                         if (idSource <= 0 || idDest <= 0) {
139                             ps.println("Erreur: les IDs doivent
                                   etre des nombres positifs");
140                             continue;
141                         }
142 
143                         if (idSource == idDest) {
144                             ps.println("Erreur: le compte source
                                   et destination doivent etre
                                   differents");
145                             continue;
146                         }
147 
148                         if (montantVirement <= 0) {
149                             ps.println("Erreur: le montant doit
                                   etre un nombre positif");
150                             continue;
151                         }
152 
153                         virementValide = true;
154                     } catch (NumberFormatException e) {
155                         ps.println("Erreur: entrez uniquement des
                                nombres !");
156                     }
157                 } while (!virementValide);
158 
159                 ps.println("\n========== RESULTAT ==========");
160                 banque.virement(idSource, idDest, montantVirement
                       );
161                 ps.println("==============================");
```

```
162                 break;
163
164             case "retrait":
165                 int idRetrait = 0, montantRetrait = 0;
166                 boolean retraitValide = false;
167
168                 do {
169                     ps.print("\nEntrez l'ID du compte : ");
170                     String[] idInput = Keyboard.readValues(is);
171
172                     ps.print("Entrez le montant du retrait : ");
173                     String[] montantInput = Keyboard.readValues(
                            is);
174
175                     try {
176                         idRetrait = Integer.parseInt(idInput[0]);
177                         montantRetrait = Integer.parseInt(
                                montantInput[0]);
178
179                         if (idRetrait <= 0) {
180                             ps.println("Erreur: l'ID doit etre un
                                    nombre positif");
181                             continue;
182                         }
183
184                         if (montantRetrait <= 0) {
185                             ps.println("Erreur: le montant doit
                                    etre un nombre positif");
186                             continue;
187                         }
188
189                         retraitValide = true;
190                     } catch (NumberFormatException e) {
191                         ps.println("Erreur: entrez uniquement des
                                nombres !");
192                     }
193                 } while (!retraitValide);
194
195                 ps.println("\n========== RESULTAT ==========");
196                 banque.retraitArgent(idRetrait, montantRetrait);
197                 ps.println("==============================");
198                 break;
199
200             case "consulter":
201                 int idConsulter = 0;
202                 boolean idValideConsult = false;
203
204                 do {
205                     ps.print("\nEntrez l'ID du compte : ");
206                     String[] idInput = Keyboard.readValues(is);
207
```

```
208                     try {
209                         idConsulter = Integer.parseInt(idInput
                                [0]);
210                         if (idConsulter > 0) {
211                             idValideConsult = true;
212                         } else {
213                             ps.println("Erreur: l'ID doit etre un
                                    nombre positif");
214                         }
215                     } catch (NumberFormatException e) {
216                         ps.println("Erreur: entrez uniquement des
                                nombres !");
217                     }
218                 } while (!idValideConsult);
219 
220                 ps.println("\n========== RESULTAT ==========");
221                 banque.consulterSolde(idConsulter);
222                 ps.println("==============================");
223                 break;
224 
225             case "quit":
226                 ps.println("\nAu revoir !");
227                 break;
228 
229             default:
230                 ps.println("\nOperation non reconnue. Utilisez:
                        creer, afficher, supprimer, virement, retrait,
                        consulter ou quit");
231                 break;
232             }
233         } while (!choix.equals("quit"));
234     }
235 
236     private static boolean authentifier() throws IOException {
237         ps.println("=== AUTHENTIFICATION ===");
238         ps.println("Veuillez vous authentifier pour acceder au
                systeme");
239 
240         int tentatives = 0;
241         final int MAX_TENTATIVES = 3;
242         final String LOGIN_CORRECT = "Admin";
243         final String MDP_CORRECT = "1234";
244 
245         while (tentatives < MAX_TENTATIVES) {
246             ps.print("\nLogin : ");
247             String[] loginInput = Keyboard.readValues(is);
248             String login = loginInput[0];
249 
250             ps.print("Mot de passe : ");
251             String[] mdpInput = Keyboard.readValues(is);
252             String mdp = mdpInput[0];
```

```
253
254             if (login.equals(LOGIN_CORRECT) && mdp.equals(
                    MDP_CORRECT)) {
255                 ps.println("\n========== AUTHENTIFICATION REUSSIE
                        ==========");
256                 ps.println("Bienvenue " + login + " !");
257                 ps.println("
                        ==============================================="
                        );
258                 return true;
259             } else {
260                 tentatives++;
261                 int restantes = MAX_TENTATIVES - tentatives;
262
263                 if (restantes > 0) {
264                     ps.println("\nErreur: login ou mot de passe
                            incorrect !");
265                     ps.println("Il vous reste " + restantes + "
                            tentative(s)");
266                 }
267             }
268         }
269
270         return false;
271     }
272 }
```