

```
1  /*
2  -----
3  Laboratoire : 04
4  Fichier : Compte.java
5  Auteur(s) : Samuel Darcey & Yves Athanasiadès
6  Date : 08.11.2015
7
8  But : Classe permettant de gérer un compte bancaire
9      et de faire des manipulations dessus.
10
11 Remarque(s) : <à compléter>
12
13 Compilateur : jdk1.8.0_60
14 -----
15 */
16
17 package Compte;
18
19 /**
20  * Classe permettant la gestion d'un compte bancaire.
21  *
22  * Author Samuel Darcey, Yves Athanasiadès
23  */
24 public class Compte {
25     /**
26      * Permet de définir avec quel solde par défaut le compte sera ouvert.
27      */
28     private final static double DEFAULT_SOLDE = 0.0;
29
30     /**
31      * Permet de définir avec quel retrait maximum le compte sera paramétré.
32      */
33     private static double defaultRetraitMax = 1000.0;
34
35     /**
36      * Permet de définir avec quel seuil de découvert le compte sera paramétré.
37      */
38     private static double defaultDecouvertMax = 800.0;
39
40     /**
41      * Défini l'id du prochain compte créé.
42      */
43     private static int currentId;
44
45     /**
46      * L'id utilisé par le compte.
47      */
48     private final int ID;
49
50     /**
51      * Le nom du titulaire du compte.
52      */
53     private final String TITULAIRE;
54
55     /**
56      * Le solde actuel du compte.
57      */
58     private double solde;
59
60     /**
61      * Le découvert actuel du compte (en valeur absolue).
62      */
63     private double decouvert;
64
65     /**
66      * Le découvert maximum autorisé sur ce compte.
67      */
68     private double decouvertMax;
69
```

```
70  /**
71   * Le retrait maximum autorisé sur ce compte.
72   */
73  private double retraitMax;
74
75  /**
76   * Crée une instance de la classe Compte.
77   *
78   * @param nom Le nom du titulaire du compte
79   * @throws IllegalArgumentException
80   */
81 public Compte(String nom) throws IllegalArgumentException {
82     this(nom, DEFAULT_SOLDE, defaultDecouvertMax, defaultRetraitMax);
83 }
84
85 /**
86  * Crée une instance de la classe Compte.
87  *
88  * @param nom Le nom du titulaire du compte
89  * @param depot Le dépôt initial du compte
90  * @throws IllegalArgumentException
91  */
92 public Compte(String nom, double depot) throws IllegalArgumentException {
93   this(nom, depot, defaultDecouvertMax, defaultRetraitMax);
94 }
95
96 /**
97  * Crée une instance de la classe Compte.
98  *
99  * @param nom      Le nom du titulaire du compte
100 * @param depot    Le dépôt initial du compte
101 * @param decouvert Le découvert maximum autorisé sur le compte
102 * @throws IllegalArgumentException
103 */
104 public Compte(String nom, double depot, double decouvert)
105   throws IllegalArgumentException {
106   this(nom, depot, decouvert, defaultRetraitMax);
107 }
108
109 /**
110  * Crée une instance de la classe Compte.
111  *
112  * @param nom      Le nom du titulaire du compte
113  * @param depot    Le dépôt initial du compte
114  * @param decouvert Le découvert maximum autorisé sur le compte
115  * @param retrait   Le retrait maximum autorisé sur le compte
116  * @throws IllegalArgumentException
117 */
118 public Compte(String nom, double depot, double decouvert, double retrait)
119   throws IllegalArgumentException {
120   //Vérifie que les entrées utilisateurs soient bonnes
121   if (nom == null || -depot > decouvert || decouvert < 0 || retrait < 0) {
122     throw new IllegalArgumentException();
123   }
124   TITULAIRE = nom;
125   ID = currentId++;
126   solde = depot;
127   decouvertMax = decouvert;
128   retraitMax = retrait;
129   actualiserDecouvert();
130 }
131
132 /**
133  * Convertit le Compte en format affichable.
134  *
135  * @return Une chaîne de caractère contenant le compte
136  * sous format affichable.
137  */
138 public String toString() {
```

```

Compte.java                                     Yves Athanasiades & Samuel Darcey - Heig-vd
139     return "\nId: " + ID + "\nTitulaire: " + TITULAIRE + "\nDecouvert Max: "
140         + decouvertMax + "\nRetrait Max: " + retraitMax + "\nSolde: "
141         + solde + "\nDecouvert?: " + (estDecouvert() ? "Oui" : "Non") + "\n";
142 }
143 /**
144 * Obtient l'id du compte courant.
145 *
146 * @return L'id du compte
147 */
148 public int getId() {
149     return ID;
150 }
151
152 /**
153 * Obtient le titulaire du compte.
154 *
155 * @return Le titulaire du compte
156 */
157 public String getTitulaire() {
158     return TITULAIRE;
159 }
160
161 /**
162 * Obtient le solde du compte.
163 *
164 * @return Le solde actuel du compte
165 */
166 public double getSolde() {
167     return solde;
168 }
169
170 /**
171 * Obtient le découvert actuel du compte.
172 *
173 * @return Le découvert actuel du compte
174 */
175 public double getDecouvert() {
176     return decouvert;
177 }
178
179 /**
180 * Obtient le découvert maximum autorisé sur le compte.
181 *
182 * @return Le découvert maximum du compte
183 */
184 public double getDecouvertMax() {
185     return decouvertMax;
186 }
187
188 /**
189 * Obtient le retrait maximum autorisé sur le compte.
190 *
191 * @return Le retrait maximum
192 */
193 public double getRetraitMax() {
194     return retraitMax;
195 }
196
197 /**
198 * Obtient le solde par défaut sur un compte.
199 *
200 * @return Le solde par défaut
201 */
202 public static double getDefaultSolde() {
203     return DEFAULT_SOLDE;
204 }
205
206 /**
207 */

```

```
208     * Obtient le découvert maximum autorisé par défaut sur un compte.  
209     *  
210     * @return Le découvert maximum par défaut  
211     */  
212     public static double getDefaultDecouvertMax() {  
213         return defaultDecouvertMax;  
214     }  
215  
216     /**  
217     * Obtient le retrait maximum autorisé par défaut sur un compte.  
218     *  
219     * @return Le retrait maximum par défaut  
220     */  
221     public static double getDefaultRetraitMax() {  
222         return defaultRetraitMax;  
223     }  
224  
225     /**  
226     * Permet de définir le découvert maximum autorisé sur le compte.  
227     *  
228     * @param montant Le montant du découvert maximum.  
229     * @throws IllegalArgumentException  
230     */  
231     public void setDecouvertMax(double montant) throws IllegalArgumentException {  
232         //Vérifie que le montant est correct  
233         if (montant < 0 || solde < -montant) {  
234             throw new IllegalArgumentException();  
235         }  
236         decouvertMax = montant;  
237     }  
238  
239     /**  
240     * Permet de définir le retrait maximum autorisé sur le compte.  
241     *  
242     * @param montant Le montant du retrait maximum  
243     * @throws IllegalArgumentException  
244     */  
245     public void setRetraitMax(double montant) throws IllegalArgumentException {  
246         //Vérifie que le montant est correct  
247         if (montant < 0) {  
248             throw new IllegalArgumentException();  
249         }  
250         retraitMax = montant;  
251     }  
252  
253     /**  
254     * Permet de définir le découvert maximum autorisé par défaut sur un compte.  
255     *  
256     * @param montant Le montant du découvert maximum  
257     * @throws IllegalArgumentException  
258     */  
259     public static void setDefaultDecouvertMax(double montant)  
260         throws IllegalArgumentException {  
261         //Vérifie que le montant est correct  
262         if (montant < 0) {  
263             throw new IllegalArgumentException();  
264         }  
265         defaultDecouvertMax = montant;  
266     }  
267  
268     /**  
269     * Permet de définir le retrait maximum autorisé par défaut sur un compte.  
270     *  
271     * @param montant Le montant du retrait maximum  
272     * @throws IllegalArgumentException  
273     */  
274     public static void setDefaultRetraitMax(double montant)  
275         throws IllegalArgumentException {  
276         //Vérifie que le montant est correct
```

```
277     if (montant < 0) {
278         throw new IllegalArgumentException();
279     }
280     defaultRetraitMax = montant;
281 }
282
283 /**
284 * Vérifie si le compte est à découvert
285 *
286 * @return Si le compte est à découvert ou pas
287 */
288 public boolean estDecouvert() {
289     return decouvert > 0;
290 }
291
292 /**
293 * Vérifie quel est le retrait maximum pouvant être effectué sur le compte.
294 *
295 * @return Le montant du retrait maximum
296 */
297 public double debitMaxAutorise() {
298     //Vérifie si on peut retirer le montant maximum ou non
299     if (Math.abs(solde - retraitMax) > decouvertMax) {
300         return solde + decouvertMax;
301     }
302     return retraitMax;
303 }
304
305 /**
306 * Crédite sur le compte une somme donnée.
307 *
308 * @param montant Le montant à créditer
309 * @throws IllegalArgumentException
310 */
311 public void crediter(double montant) throws IllegalArgumentException {
312     //Vérifie que le montant est correct
313     if (montant < 0) {
314         throw new IllegalArgumentException();
315     }
316     solde += montant;
317     actualiserDecouvert();
318 }
319
320 /**
321 * Débite sur le compte une somme donnée.
322 *
323 * @param montant Le montant à débiter
324 * @return Le montant débité
325 * @throws IllegalArgumentException
326 */
327 public double debiter(double montant) throws IllegalArgumentException {
328     //Vérifie que le montant est correct
329     if (montant < 0) {
330         throw new IllegalArgumentException();
331     }
332     //Vérifie que le montant n'excède pas le retrait max
333     if (montant > retraitMax) {
334         montant = retraitMax;
335     }
336     //Vérifie que le montant demandé ne dépasse pas le découvert max
337     if (Math.abs(solde - montant) > decouvertMax) {
338         montant = solde + decouvertMax;
339     }
340     solde -= montant;
341     actualiserDecouvert();
342     return montant;
343 }
344
345 /**
```

```
346     * Effectue un virement d'un compte à un autre.  
347     *  
348     * @param debit Le compte à débiter  
349     * @param credit Le compte à créditer  
350     * @param montant Le montant à débiter/créditer  
351     * @return Le montant débité/crédié  
352     * @throws IllegalArgumentException  
353     */  
354     public static double virer(Compte debit, Compte credit, double montant)  
355         throws IllegalArgumentException {  
356             //Vérifie que les comptes existent  
357             //Le test du montant est fait dans debiter() et crediter()  
358             if (debit == null || credit == null) {  
359                 throw new IllegalArgumentException();  
360             }  
361             montant = debit.debiter(montant);  
362             credit.crediter(montant);  
363             return montant;  
364         }  
365         /**  
366             * Met le découvert à jour en fonction du solde du compte.  
367             */  
368         private void actualiserDecouvert() {  
369             if (solde < 0) {  
370                 decouvert = -solde;  
371             }  
372             else {  
373                 decouvert = 0;  
374             }  
375         }  
376     }  
377 }  
378 }
```