IUT de Paris Descartes – Base de la Programmation Objet

Travaux Dirigés – Sujet n°8

Objectifs

— Définir une hiérarchie de classes
— Définir des constructeurs dans une telle hiérarchie
— Spécialiser et employer des méthodes héritées

Un compte bancaire sécurisé

À la demande de certains de ses clients, notre banque suisse souhaite proposer des comptes sécurisés. Ce nouveau type de comptes dispose des mêmes fonctionnalités que les autres. Toutefois, pour limiter les pertes en cas d'accès frauduleux, le montant cumulé des retraits sur les 5 derniers jours ne pourra jamais dépasser une somme fixée par le client.

Les éléments publics de la classe `Compte` (vue lors d'une séance de travaux dirigés précédente) sont rappelés en annexe.

1. Déclarez une classe nommée `CompteSécurisé` héritant de la classe `Compte`. Faites le minimum pour que cette nouvelle classe puisse être compilée sans erreur.

   Solution: Le solde initial d'un compte doit toujours être précisé lorsqu'un compte est créé (i.e. la classe `Compte` ne dispose que d'un constructeur prenant un entier en paramètre). Il est donc obligatoire de d'ajouter au moins un constructeur à la classe `CompteSécurisé` en précisant comment initialiser le solde du compte correspondant.
   On note que l'appel au constructeur de la classe mère (i.e. l'instruction '`super(soldeInitial)`') doit être la première instruction du constructeur.

   ```
   public class CompteSécurisé extends Compte {
     public CompteSécurisé(long soldeInitial) {
       super(soldeInitial);
     }
   }
   ```

2. Un compte sécurisé doit pouvoir être paramétré par le montant maximal des retraits cumulés sur 5 jours que le client autorise. Ce montant doit pouvoir être précisé à la construction. Déclarez un attribut permettant de stocker ce montant et modifiez le constructeur en conséquence.

   Solution:

   ```
   public class CompteSécurisé extends Compte {
     private long cumulMax;

     public CompteSécurisé(long soldeInitial, long cumulMax) {
       super(soldeInitial);
       assert(cumulMax > 0);
   ```

```
    this.cumulMax = cumulMax;
  }
}
```

3. La règle de sécurité devant être mise en place impose de stocker au sein d'un compte sécurisé le montant et la date des retraits ayant eu lieu. Seuls les retraits vieux d'au plus 5 jours doivent être conservés.
   Pour faciliter cette tâche, une classe nommée `Retrait` a été développée. Les éléments publics qu'elle propose sont les suivants :

```
public class Retrait {
  /** Construit un retrait daté à la date d'aujourd'hui */
  public Retrait(long montant) { ... }
  /** Retourne le nombre de jours séparant le retrait de
   *  la date d'aujourd'hui */
  public long age() { ... }
  /** Retourne le montant du retrait */
  public long montant() { ... }
}
```

   Pour les curieux, le code complet de cette classe est sur le disque `COMMUN`.
   Les derniers retraits ayant été effectués sur un compte sécurisé doivent être stockés dans une liste de type `LinkedList<Retrait>`. Une liste vide peut être obtenue par l'expression **new** `LinkedList<Retrait>()`. Déclarez un attribut de ce type et initialisez le dans le constructeur.

Solution:

```
public class CompteSécurisé extends Compte {
  private long cumulMax;
  private LinkedList<Retrait> derniersRetraits;

  public CompteSécurisé(long soldeInitial, long cumulMax) {
    super(soldeInitial);
    assert(cumulMax > 0);
    this.cumulMax = cumulMax;
    this.derniersRetraits = new LinkedList<Retrait>();
  }
}
```

4. Sachant qu'une liste peut être parcourue avec une boucle '**foreach**', spécialisez la méthode `retraitPossible` d'un compte sécurisé. Vous devez vous assurer que le montant cumulé des retraits datant d'au plus 5 jours ne dépassera pas le maximum autorisé.

Solution: Il ne faut pas oublier de vérifier que le retrait est possible au sens du compte normal.

```
public class CompteSécurisé extends Compte {
  private static final int duréePériode = 5;
  ...

  @Override
  public boolean retraitPossible(long val) {
    if (!super.retraitPossible(val))
      return false;
    long cumul = val;
    for (Retrait r : derniersRetraits)
      if (r.age() <= duréePériode)
        cumul += r.montant();
```

```
    return cumul <= cumulMax;
  }
}
```

5. Tout nouveau retrait doit être mémorisé dans la liste. De plus, les anciens retraits de plus de 5 jours peuvent être supprimés. Pour ce faire, la liste doit être gérée comme une file. L'extrait de code suivant illustre les différentes opérations qui vous sont nécessaires sur une liste contenant des chaînes de caractères :

```
LinkedList<String> liste;
liste = new LinkedList<String>(); // liste = {}
liste.add("hello");               // liste = {"hello"}
liste.add("world");               // liste = {"hello", "world"}
String s = liste.element();       // s = "hello"
liste.remove();                   // liste = {"world"}
if (liste.isEmpty())
  System.out.println("bizarre");
```

Spécialisez la méthode `retirer` de façon à mettre à jour la liste des retraits.

Solution: La précondition de cette méthode est que le retrait soit possible :

```
assert(retraitPossible(val)
```

Toutefois, la méthode `retirer` de la classe Compte (`super.retirer(val)`) à la même précondition (et le même `assert`). Comme nous avons spécialisé la méthode `retraitPossible` et que c'est la plus spécialisée qui sera invoqué, il n'est pas nécessaire de répéter le test ici.

```
public class CompteSécurisé extends Compte {
  private static final int duréePériode = 5;
  ...

  @Override
  public void retirer(long val) {
    super.retirer(val);
    while (!derniersRetraits.isEmpty()
        && derniersRetraits.element().age() > duréePériode)
      derniersRetraits.remove();
    derniersRetraits.add(new Retrait(val));
  }
}
```

Annexe – Éléments publics de la classe `Compte`

```
public class Compte {
  public static void setDécouvertMax(long val) { ... }
  public static long getDécouvertMax() { ... }

  public Compte(long val) { ... }
  public void déposer(long val) { ... }
  public boolean retraitPossible(long val) { ... }
  public void retirer(long val) { ... }
  public String toString() { ... }
}
```