TD 5 : Contrats, exceptions

Objectifs pédagogiques :

- **exceptions**
- **contrats**

5.1 Gestion des exceptions

Soit le programme Java de l'annexe 1.

Question 1. Donnez les affichages du programme exécuté par `java ExempleSimple`.

Question 2. Même question en ajoutant les sources de l'annexe 2 pour l'exécution du programme par `java ExempleMultiple`.

Question 3. Une partie du programme est commentée, elle ne compile pas. Expliquez pourquoi.

5.2 Contrats pour piles

Dans cet exercice, nous allons utiliser l'approche dite de *conception par contrats* pour concevoir et implémenter une classe simple pour les structures de données de piles bornées.

Le squelette de la classe est le suivant :

```
public class BoundedStack {
  private Object[] content;
  private int size;

  public BoundedStack(int limit) {
    content = new Object[(limit < 0) ? 0 : limit];
  }

  public int getSize() { return size; }
  public int getLimit() { return content.length; }

  // suite de la classe à compléter...
}
```

Une pile p est considérée pleine quand $p.getSize() == p.getLimit()$.

Elle est vide quand $p.getSize() == 0$.

Nous souhaitons implémenter trois méthodes supplémentaires :

+ `top() : Object` qui retourne l'élément en haut de la pile sans changer la pile,
+ `push(Object o) : void` qui ajoute un élément en haut de la pile,
+ `pop() : Object` qui dépile l'élément en haut de la pile et le retourne.

5.2.1 Contractualisation

Question 4. Quels invariants doit respecter l'état de la pile pour rester cohérent ?

Question 5. Écrivez, de façon informelle, les contrats pour chacune de ces méthodes en précisant :

- le fournisseur des contrats,
- les clients,
- les pré-requis du client (ou pré-conditions) sur l'état de la pile (taille, état plein ou vide),
- les garanties du fournisseur (i.e., les mises à jour de l'état).

5.2.2 Implémentation

La *programmation défensive* consiste à tester les pré-requis au début de l'invocation d'une méthode. S'ils sont invalides, il faut signaler l'erreur au client (c'est de *sa* faute). La façon propre de le faire en Java est d'utiliser des exceptions. Nous allons donc prévoir des exceptions qui couvrent les mauvais usages potentiels d'une pile par le client.

Question 6. Définissez une hiérarchie de classes d'exceptions permettant de prendre en compte :

- les erreurs génériques sur les piles (`StackException`) ;
- les erreurs spécifiques : pile vide (`StackEmptyException`) ou pile pleine (`StackFullException`).

Question 7. Donnez le code des méthodes `top`, `push` et `pop` de `BoundedStack` en générant correctement les exceptions en cas de non-respect des pré-requis.

5.2.3 Tests unitaires

Pour tester qu'une implémentation se conforme à une spécification, comme les contrats que nous avons définis sur la pile, nous utilisons des tests. Les tests cherchent à vérifier que les garanties posées dans la spécification sont bien honorées par l'implémentation. En général, un test est une *séquence* d'invocations de méthodes sur un objet. Nous attendons des résultats précis, que nous vérifions à l'aide d'assertions ; si elles sont violées le test échoue. Ici, nous souhaitons tester que :

- l'implémentation satisfait ses invariants ;
- l'implémentation satisfait les garanties identifiées sur chaque méthode ;
- l'implémentation lève des exceptions adaptées quand les pré-requis ne sont pas respectés.

Question 8. Définissez une classe de test unitaire avec déclaration et initialisation d'une pile vide, d'une pile à moitié pleine et d'une pile pleine (vous définirez pour cela une méthode annotée avec `@Before`).

Question 9. Pour chaque méthode de `BoundedStack`, écrivez un test unitaire permettant de :

- tester les pré-requis (au moins un cas pré-requis assuré et un autre cas pré-requis faux) ;
- tester les garanties (si les pré-requis sont vérifiés).

Annexe 1

MonException.java

```java
public class MonException extends Exception {
   public MonException(String message) {
      super(message);
   }
}
```

Generateur.java

```java
public class Generateur {
   private String genstr;
   public Generateur(String genstr) {
      this.genstr = genstr;
   }

   public void tryMe(String str) throws MonException {
      if(str.equals(genstr))
         throw new MonException("dans tryMe");
   }
}
```

ExempleSimple.java

```
public class ExempleSimple {
   public void premierTest(Generateur gen) {
      System.out.println("Premier test :");
      String chaine = "Alea jacta est";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MonException e) {
         System.out.println("trois");
      }
      System.out.println("quatre");
   }

   public void secondTest(Generateur gen) {
      System.out.println("Second test :");
      String chaine = "Tu quoque fili";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MonException e) {
         System.out.println("trois");
      }
      System.out.println("quatre");
   }

   public void troisiemeTest(Generateur gen) throws MonException {
      System.out.println("Troisieme test :");
      String chaine = "Alea jacta est";
      System.out.println("un");
      gen.tryMe(chaine);
      System.out.println("deux");
   }

   public void quatriemeTest(Generateur gen) throws MonException {
      System.out.println("Quatrieme test :");
      String chaine = "Tu quoque fili";
      System.out.println("un");
      gen.tryMe(chaine);
      System.out.println("deux");
   }

   public void cinquiemeTest(Generateur gen) throws MonException {
      System.out.println("Cinquieme test :");
      String chaine = "Alea jacta est";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MonException e) {
         System.out.println("trois");
         throw e;
      } finally {
         System.out.println("trois bis");
      }
      System.out.println("quatre");
   }

   public void sixiemeTest(Generateur gen) throws MonException {
```

```
      System.out.println("Sixieme test :");
      String chaine = "Tu quoque fili";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MonException e) {
         System.out.println("trois");
         throw e;
      } finally {
         System.out.println("trois bis");
      }
      System.out.println("quatre");
   }

   public static void main(String[] args) {
      Generateur gen = new Generateur("Alea jacta est");
      ExempleSimple exemple = new ExempleSimple();
      exemple.premierTest(gen);
      exemple.secondTest(gen);
      try {
         exemple.troisiemeTest(gen);
      } catch(MonException e) {
         System.out.println("cinq");
      }
      try {
         exemple.quatriemeTest(gen);
      } catch(MonException e) {
         System.out.println("cinq");
      }
      try {
         exemple.cinquiemeTest(gen);
      } catch(MonException e) {
         System.out.println("cinq");
      }
      try {
         exemple.sixiemeTest(gen);
      } catch(MonException e) {
         System.out.println("cinq");
      }
   }
}
```

Annexe 2

MaSousException1.java

```
public class MaSousException1 extends MonException {
   public MaSousException1(String message) {
      super("Sous exception 1:" + message);
   }
}
```

MaSousException2.java

```
public class MaSousException2 extends MonException {
   public MaSousException2(String message) {
      super("Sous exception 2:" + message);
   }
```

```
}
```

GenMultiple.java

```
public class GenMultiple extends Generateur {
   private String genstr1;
   private String genstr2;

   public GenMultiple(String gen, String gen1, String gen2) {
      super(gen);
      genstr1 = gen1;
      genstr2 = gen2;
   }

   public void tryMe(String str) throws MonException {
      super.tryMe(str);
      if(str.equals(genstr1))
         throw new MaSousException1("Cas 1");
      else if(str.equals(genstr2))
         throw new MaSousException2("Cas 2");
   }
}
```

ExempleMultiple.java

```
public class ExempleMultiple {
   public void premierTest(Generateur gen) {
      System.out.println("Premier test multiple :");
      String chaine = "Alea jacta est";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MaSousException1 e) {
         System.out.println("trois");
      } catch(MaSousException2 e) {
         System.out.println("trois bis");
      } catch(MonException e) {
         System.out.println("trois ter");
      }
      System.out.println("quatre");
   }

   public void secondTest(Generateur gen) {
      System.out.println("Second test multiple :");
      String chaine = "Veni, vidi";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MaSousException1 e) {
         System.out.println("trois");
      } catch(MaSousException2 e) {
         System.out.println("trois bis");
      } catch(MonException e) {
         System.out.println("trois ter");
      }
      System.out.println("quatre");
   }

```

```
   public void troisiemeTest(Generateur gen) {
      System.out.println("Troisieme test multiple :");
      String chaine = "Cogito ergo sum";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MaSousException1 e) {
         System.out.println("trois");
      } catch(MaSousException2 e) {
         System.out.println("trois bis");
      } catch(MonException e) {
         System.out.println("trois ter");
      }
      System.out.println("quatre");
   }

   public void quatriemeTest(Generateur gen) {
      System.out.println("Quatrieme test multiple :");
      String chaine = "Fluctua nec mergitur";
      try {
         System.out.println("un");
         gen.tryMe(chaine);
         System.out.println("deux");
      } catch(MaSousException1 e) {
         System.out.println("trois");
      } catch(MaSousException2 e) {
         System.out.println("trois bis");
      } catch(MonException e) {
         System.out.println("trois ter");
      }
      System.out.println("quatre");
   }

/* NE COMPILE PAS :
  public void cinquiemeTest(Generateur gen) {
    System.out.println("Cinquieme test multiple :");
    String chaine = "Veni, vidi";
    try {
      gen.tryMe(chaine);
     } catch(MonException e) {
       System.out.println("un");
     } catch(MaSousException1 e) {
       System.out.println("deux");
     } catch(MaSousException2 e) {
       System.out.println("trois");
     }
     System.out.println("quatre");
    }
*/

   public static void main(String[] args) {
      GenMultiple gen = new GenMultiple("Alea jacta est","Cogito ergo sum","Veni, vidi");
      ExempleMultiple exemple=new ExempleMultiple();
      exemple.premierTest(gen);
      exemple.secondTest(gen);
      exemple.troisiemeTest(gen);
      exemple.quatriemeTest(gen);
      //exemple.cinquiemeTest(gen);
   }
}
```