Java : les exceptions

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Introduction

Une exception, c'est quoi ?

- C'est une erreur qui se produit pendant l'exécution de notre programme
- Une exception dans un programme implique généralement son arrêt d'exécution

Introduction

Comment faire pour poursuivre l'exécution ?

- Repérer les blocs pouvant générer une exception
- Capturer l'exception correspondante
- Afficher un message relatif à cette exception
- Continuer l'exécution

Introduction

`Exception` : exemple

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    System.out.print(x/y);
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Introduction

`Exception` : exemple

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    System.out.print(x/y);
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception in thread "main" java.lang.ArithmeticException : / by zero at test.FirstClass.main(FirstClass.java :4)

Introduction

`Exception` : exemple

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    System.out.print(x/y);
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception in thread "main" java.lang.ArithmeticException : / by zero at test.FirstClass.main(FirstClass.java :4)

Constatation

- Le message `Fin de calcul` n'a pas été affiché
- La division par zéro déclenche une exception `ArithmeticException`

Capture d'exception

Comment faire pour capturer une exception ?

- Utiliser un bloc `try { ... } catch { ... }`
- Le `try { ... }` pour entourer une instruction susceptible de déclencher une exception
- Le `catch { ... }` pour capturer l'exception et afficher un message qui lui correspond

Capture d'exception

`Exception` : **exemple**

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (ArithmeticException e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Capture d'exception

`Exception` **: exemple**

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (ArithmeticException e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

```
Exception : Division par zéro
Fin de calcul
```

Capture d'exception

`Exception` **: exemple**

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (ArithmeticException e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception : Division par zéro
Fin de calcul

Constatation

- L'exception a été capturée
- Le message `Fin de calcul` a été affiché

Capture d'exception

Et si je ne connais pas le type d'exception

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Capture d'exception

Et si je ne connais pas le type d'exception

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Le même message sera affiché

Exception : Division par zéro
Fin de calcul

Capture d'exception

Et si je ne connais pas le type d'exception

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Le même message sera affiché

```
Exception : Division par zéro
Fin de calcul
```

Constatation

- La classe `Exception` peut être utilisée

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Exception : " + e.getMessage());
    }
    System.out.println("Fin de calcul");
  }
}
```

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Exception : " + e.getMessage());
    }
    System.out.println("Fin de calcul");
  }
}
```

Le même message sera affiché

Exception : / by zero
Fin de calcul

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    System.out.println("Fin de calcul");
  }
}
```

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    System.out.println("Fin de calcul");
  }
}
```

Le message affiché est :

```
java.lang.ArithmeticException : / by zero
at test.FirstClass.main( FirstClass.java :49 )
Fin de calcul
```

Les exceptions personnalisées

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `ArithmeticException`
- `NullPointerException`

Les exceptions personnalisées

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `ArithmeticException`
- `NullPointerException`

On peut aussi définir nos exceptions personnalisées

Les exceptions personnalisées

Considérons la classe `Adresse` suivante

```
public class Adresse {
  private String rue;
  private String ville;
  private String codePostal;
  public Adresse(String rue, String ville, String
     codePostal) {
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
  // ensuite les getters/setters et autres méthodes
  ...
}
```

Les exceptions personnalisées

Supposons que

- `codePostal` doit contenir exactement 5 chiffres

Les exceptions personnalisées

Supposons que

- `codePostal` doit contenir exactement 5 chiffres

Démarche à faire

- Créer notre propre exception (qui doit étendre la classe `Exception`)
- Dans le constructeur de `Adresse`, on lance une exception si `codePostal` ne contient pas 5 chiffres

Les exceptions personnalisées

Créons l'exception `IncorrectCodePostalException` **dans un package** `org.eclipse.exceptions`

```
public class IncorrectCodePostalException extends
   Exception {

  // le constructeur de cette nouvelle exception
  public IncorrectCodePostalException(){
    System.out.println("Le code postal doit contenir
        exactement 5 chiffres");
  }
}
```

Les exceptions personnalisées

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {
  // après les attributs
  public Adresse(String rue, String ville, String
     codePostal) throws IncorrectCodePostalException
     {
    if (codePostal.length()!=5)
      throw new IncorrectCodePostalException();
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
}
// il faut faire pareil dans setCodePostal()
```

Les exceptions personnalisées

Testons tout cela dans le `main()`

```
public static void main(String[] args) {
  Adresse a = null;
  try {
    a = new Adresse ("rue de paradis", "Marseille",
       "1300");
  }
  catch(IncorrectCodePostalException icpe) {
    icpe.printStackTrace();
  }
}
```

Les exceptions personnalisées

Testons tout cela dans le `main()`

```
public static void main(String[] args) {
  Adresse a = null;
  try {
    a = new Adresse ("rue de paradis", "Marseille",
       "1300");
  }
  catch(IncorrectCodePostalException icpe) {
    icpe.printStackTrace();
  }
}
```

Le message affiché est :

Le code postal doit contenir exactement 5 chiffres

Les instructions multi-catch

On peut rajouter une deuxième condition

- `codePostal` doit contenir exactement 5 chiffres
- `rue` doit être une chaîne en majuscule

Les instructions multi-catch

Créons une deuxième exception `IncorrectStreetNameException` **dans le package** `org.eclipse.exceptions`

```
public class IncorrectStreetNameException extends
   Exception {

  public IncorrectStreetNameException() {
    System.out.print("Le nom de la rue doit être en
       majuscule");
  }

}
```

Les instructions multi-catch

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {
  // après les attributs
  public Adresse(String rue, String ville, String
     codePostal) throws IncorrectCodePostalexception
     , IncorrectStreetNameException {
    if (codePostal.length()!=5)
      throw new IncorrectCodePostalException();
    if(!rue.equals(rue.toUpperCase()))
      throw new IncorrectStreetNameException();
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
}
```

Les instructions multi-catch

Re-testons tout cela dans le `main()`

```
public static void main(String[] args) {
  try {
    Adresse a = new Adresse ("paradis", "Marseille",
        "1300");
  }
  catch(IncorrectCodePostalException icp) {
    icp.printStackTrace();
  }
  catch(IncorrectStreetNameException isn) {
    isn.printStackTrace();
  }
}
```

Les instructions multi-catch

Depuis Java 7, on peut écrire :

```
public static void main(String[] args) {
  try {
    Adresse a = new Adresse ("paradis", "Marseille",
        "1300");
  }
  catch(IncorrectCodePostalException |
     IncorrectStreetNameException e) {
    e.printStackTrace();
  }
}
```

Les exceptions paramétrées

Question

Comment faire si on veut afficher les valeurs qui ont déclenché l'exception dans le message ?

Les exceptions paramétrées

Modifions la première exception

`IncorrectCodePostalException`

```
public class IncorrectCodePostalException extends
   Exception {
  // le constructeur de cette nouvelle exception
  public IncorrectCodePostalException(String cp){
    System.out.println("Le code postal '" + cp + "'
      doit contenir exactement 5 chiffres");
                    }
}
```

Les exceptions paramétrées

Modifions la deuxième exception

`IncorrectStreetNameException`

```
public class IncorrectStreetNameException extends
   Exception {
  public IncorrectStreetNameException(String rue) {
    System.out.print("Le nom de la rue '" + rue + "'
        doit être en majuscule");
  }
}
```

Les exceptions paramétrées

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {
  // après les attributs
  public Adresse(String rue, String ville, String
     codePostal) throws IncorrectCodePostalexception
     , IncorrectStreetNameException {
    if (codePostal.length()!=5)
      throw new IncorrectCodePostalException(
         codePostal);
    if(!rue.equals(rue.toUpperCase()))
      throw new IncorrectStreetNameException(rue);
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
}
```

Les exceptions paramétrées

Pour tester

```
public static void main(String[] args) {
  try {
    Adresse a = new Adresse ("paradis", "Marseille",
        "1300");
  }
  catch(IncorrectCodePostalException |
     IncorrectStreetNameException e) {
    e.printStackTrace();
  }
}
```

Les exceptions paramétrées

Pour tester

```
public static void main(String[] args) {
  try {
    Adresse a = new Adresse ("paradis", "Marseille",
        "1300");
  }
  catch(IncorrectCodePostalException |
     IncorrectStreetNameException e) {
    e.printStackTrace();
  }
}
```

Le message affiché est :

Le code postal ’1300’ doit contenir exactement 5 chiffres

Les exceptions paramétrées

Exercice

Créer une nouvelle classe d'exception `AdresseException` pour fusionner et remplacer les deux exceptions `IncorrectCodePostalException` et `IncorrectStreetNameException`

Les exceptions paramétrées

Première solution : contenu de la classe `AdresseException`

```
public class AdresseException {
  public AdresseException(String cp, String rue) {
    if (cp.length() != 5) {
      System.out.println("Le code postal '" + cp + "
        ' doit contenir exactement 5 chiffres ");
    }
    if (!rue.equals(rue.toUpperCase())) {
      System.out.print("Le nom de la rue '" + rue +
        "' doit etre en majuscule ");
    }
  }
}
```

Les exceptions paramétrées

Contenu de la classe `Adresse`

```
public class Adresse {
  // après les attributs

  public Adresse(String rue, String ville, String codePostal)
     throws AdresseException {

    if (codePostal.length() != 5 || !rue.equals(rue.toUpperCase
       ())) {
      throw new AdresseException(codePostal, rue);
    }
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
  // + getters + setters + toString()
}
```

Les exceptions paramétrées

Remarque

Cette classe exception ne peut pas être appelé par `setRue()` ni par `setCodePostal()`. Seul le constructeur de la classe `Adresse` vérifie les deux et appelle l'exception en lui envoyant les deux paramètres `rue` et `codePostal`.

Les exceptions paramétrées

Deuxième solution : contenu de la classe `AdresseException`

```
public class AdresseException {
  public AdresseException(String cp, String rue) {
    if (cp.length() != 5) {
      System.out.println("Le code postal '" + cp + "
        ' doit contenir exactement 5 chiffres ");
    }
    if (!rue.equals(rue.toUpperCase())) {
      System.out.print("Le nom de la rue '" + rue +
        "' doit etre en majuscule ");
    }
  }
}
```

Les exceptions paramétrées

Contenu de la classe `Adresse`

```
public class Adresse {
  // après les attributs
  public Adresse(String rue, String ville, String codePostal) throws
     AdresseException {

    if (codePostal.length() != 5 && !rue.equals(rue.toUpperCase())) {
      throw new AdresseException(3, codePostal, rue);
    }
    else {
      if (codePostal.length() != 5) {
        throw new AdresseException(1, codePostal);
      }
      if (!rue.equals(rue.toUpperCase())) {
        throw new AdresseException(2, rue);
      }
    }
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }
  // + getters + setters + toString()
}
```

Le bloc `finally`

- À utiliser quand on veut exécuter une instruction qu'une exception soit levée ou non

Le bloc finally

Exemple

```
public class Test {
  public static void main(String[] args) {
    int x = 5, y = 0;
    try {
      System.out.println(x/y);
    }
    catch (Exception e) {
      System.out.println("Division par zéro");
    }
    finally{
      System.out.println("Instruction exécutée systé
        matiquement");
    }
  }
}
```

Le bloc `finally`

Remarque

- Le bloc `finally` peut s'avérer intéressant si le `catch` contiens un `return` qui forcera l'arrêt de l'exécution du code. Malgré cela, ce bloc (`finally`) sera exécuté.