Java : 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

Java

Exception

- Erreur qui se produit pendant l'exécution du programme
- Impliquant, généralement, l'arrêt d'exécution

Java

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

`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");
  }
}
```

`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)

`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`

Java

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

Java

`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");
  }
}
```

Java

`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");
  }
}
```


Message affiché à l'exécution (exception capturée)

Exception : Division par zéro
Fin de calcul

Java

Question

Et si on connaissait pas la classe d'exception ?

Java

Question

Et si on connaissait pas la classe d'exception ?

Réponse

On peut utiliser la classe `Exception` : classe mère de toute classe d'exception.

Java

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("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

Java

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("Exception : Division par zéro ");
    }
    System.out.println("Fin de calcul");
  }
}
```

même message affiché

Exception : Division par zéro
Fin de calcul

Java

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");
  }
}
```

Java

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");
  }
}
```

Message affiché

Exception : / by zero
Fin de calcul

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");
  }
}
```

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");
  }
}
```


Message affiché

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

Java

On a utilisé quelques exceptions prédéfinies

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

Java

On a utilisé quelques exceptions prédéfinies

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

On peut aussi définir nos exceptions personnalisées.

Java

Considérons la classe `Adresse` suivante

```
package org.eclipse.model;

public class Adresse {

  private String rue;
  private String ville;
  private String codePostal;

  public Adresse(String rue, String codePostal, String ville) {
    this.rue = rue;
    this.ville = ville;
    this.codePostal = codePostal;
  }

  // ensuite les getters/setters et autres méthodes

}
```

Java

Hypothèse

`codePostal` doit contenir exactement 5 chiffres.

Java

Hypothèse

`codePostal` doit contenir exactement 5 chiffres.

Comment faire

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

Java

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

```
public class IncorrectCodePostalException extends Exception {

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

Java

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {

  private String rue;
  private String ville;
  private String codePostal;

  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;
  }

  // + les autres méthodes
}
```

Java

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();
  }
}
```

Java

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();
  }
}
```


Message affiché

Le code postal doit contenir exactement 5 chiffres.

Java

Deuxième hypothèse

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

Java

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

```
public class IncorrectStreetNameException extends Exception {

    public IncorrectStreetNameException() {
        super("La rue doit être en majuscule");
    }

}
```

Java

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {

  private String rue;
  private String ville;
  private String codePostal;

  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;
  }
}
```

Java

Pour tester dans `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();
  }
}
```

Java

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();
  }
}
```

Java

Question

Comment faire si on voulait afficher les valeurs ayant déclenché l'exception dans le message ?

Java

Modifions la première exception `IncorrectCodePostalException`

```
public class IncorrectCodePostalException extends Exception {

  public IncorrectCodePostalException(String cp) {
    super("Le code postal '" + cp + "' doit contenir 5 chiffres");
  }

}
```

Les exceptions paramétrées

Modifions la deuxième exception `IncorrectStreetNameException`

```
public class IncorrectStreetNameException extends Exception {

  public IncorrectStreetNameException(String rue) {
    System.out.print("La rue '" + rue + "' doit être en majuscule");
  }

}
```

Java

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {

  private String rue;
  private String ville;
  private String codePostal;

  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;
  }
}
```

Java

Pour tester

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

Java

Pour tester

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


Message affiché

Le code postal ’1300’ doit contenir exactement 5 chiffres.

Java

Exercice

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

Java

finally

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

Java

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 toujours exécutée");
    }
  }
}
```

Java

Question

Quelle différence entre le code de `finally` et le code après `try ... catch` ?

Java

Question

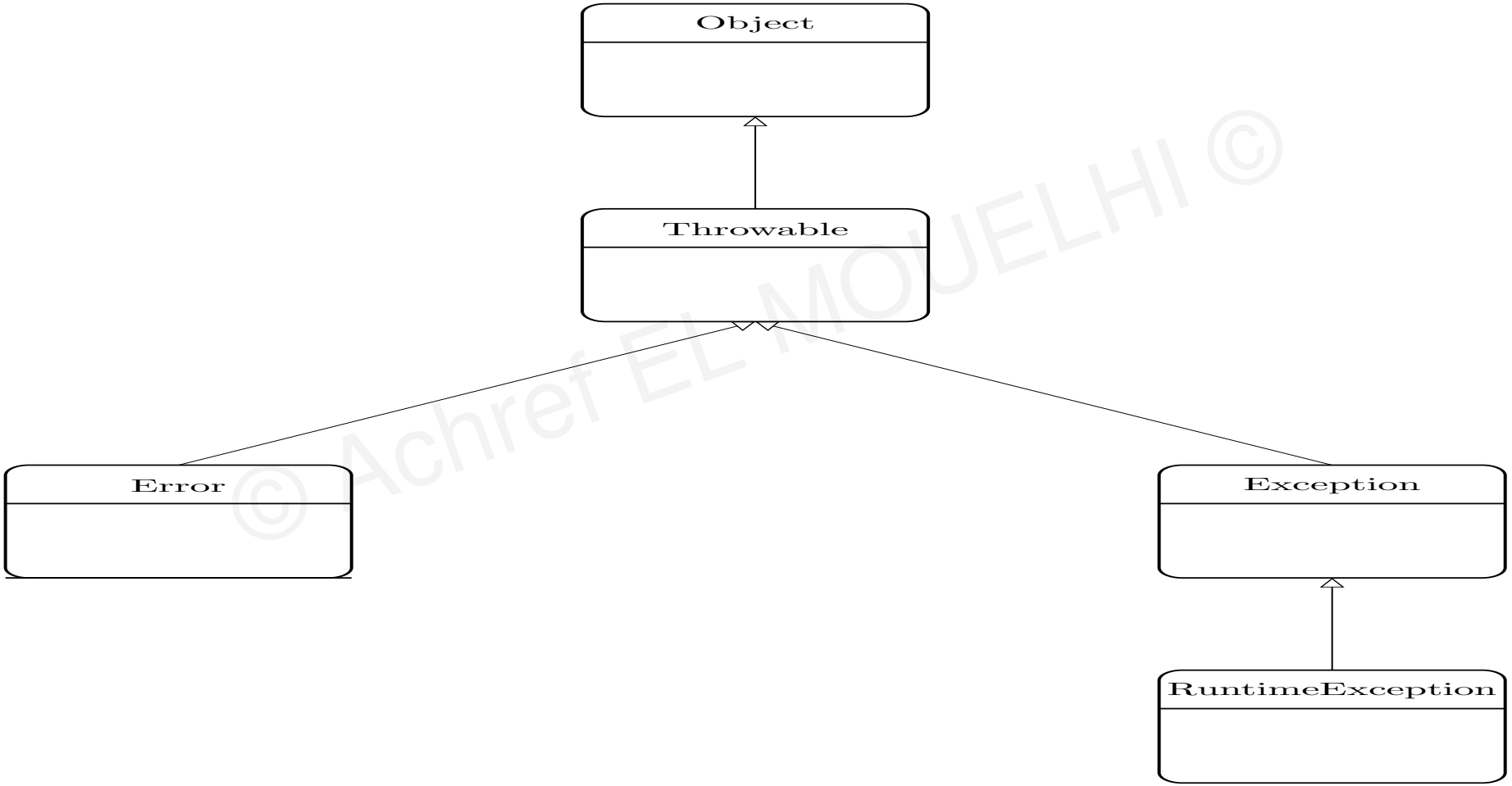
Quelle différence entre le code de `finally` et le code après `try ... catch` ?

Réponse

Le code de `finally` s'exécute même si `try` et/ou `catch` contiennent un `return` qui forcera l'arrêt de l'exécution du code (et donc le non-exécution du code situé après `try ... catch`).

UML

Hiérarchie de classes d'exceptions



Java

Quelle est la différence entre `Error` et `Exception` ?

- `Error`
  - Problème qui se produit principalement en raison du manque de ressources (mémoire...)
  - Pas besoin de capturer ce genre de problèmes.
- `Exception`
  - Problème qui se produit principalement dans le code écrit par les développeurs.
  - Certaines peuvent être vérifiées (**Checked Exception**) et certaines autres non (**Unchecked Exception**).

Java

Quelle est la différence entre `RuntimeException` et `Exception` ?

- `RuntimeException`
  - Exception non-vérifiée : exception détectée à l'exécution (Pas besoin de la capturer).
  - Ne peut être détectée par le compilateur.
  - Exemple : `IndexOutOfBoundsException`, `ArithmeticException`...
- `Exception`
  - Exception vérifiée : exception détectée à la compilation (le compilateur nous oblige à capturer l'exception).
  - Exemple : `SQLException`, `IOException`...