TP6 JAVA – Programmation Orientée Objets

Rafael COLARES

Maitre de Conférences ISIMA

Bureau D104 – email:rafael.colares_borges@uca.fr

Leçons apprises du TP5

- EXCEPTION HANDLING
- CHECKED VS. UNCHECKED EXCEPTIONS
- CUSTOMIZED EXCEPTIONS

Exception handling

- Exemple: Lecture de données

Class Integer

java.lang.Object
    java.lang.Number
        java.lang.Integer

All Implemented Interfaces:

Serializable, Comparable<Integer>

```
static int    parseInt(String s)
              Parses the string argument as a signed decimal integer.
```

```
int myInt = Integer.parseInt(« NaN »);
```

NumberFormatException est levée

Si pas traité,

Exception handling

- Bonne pratique: prévoir les scénarios possibles

```
try {
    int myInt = Integer.parseInt(« NaN »);
}
catch (NumberFormatException e){
    System.out.println(« [Warning] Could not convert to int »);
}
```

Exception handling

- Bonne pratique: prévoir les scénarios possibles

```
try {
    int myInt = Integer.parseInt(« NaN »);
}
catch (NumberFormatException e){
    System.out.println(« [Warning] Could not convert to int »);
}
```

Code qui pourrait lever une exception

Code à exécuter au cas où l'exception est levée

Exception handling

- Pas besoin de traiter l'exception tout de suite
  - Important de l'attraper à un moment (**`catch`**)

L'exception lancée va monter la pile d'appels (*call stack*) jusqu'à ce qu'elle soit attrapée

- Si traitement est fait plus tard:
  - Bonne pratique: annoncer la possibilité d'exceptions (**`throws`**)

App.java

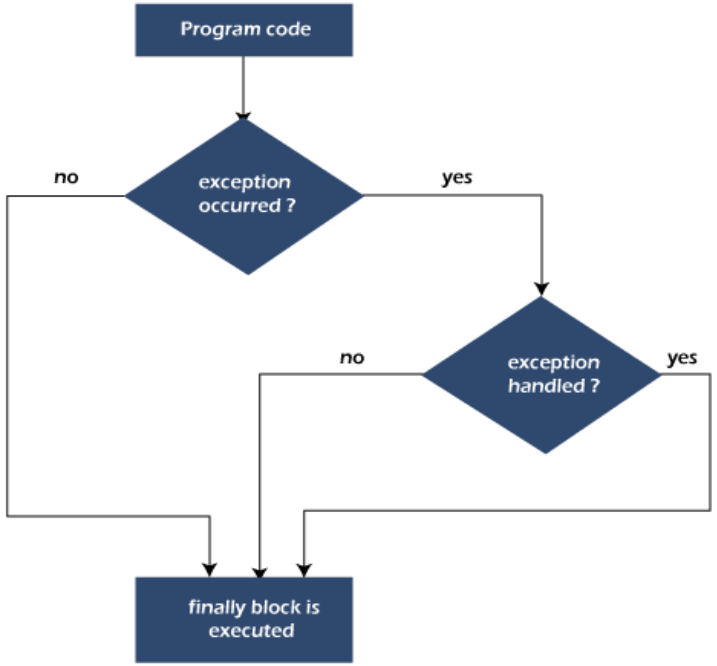
```
public class App{
    public static void main(String[] args) {
        try {
            int myInt = getInt();
        }
        catch (IllegalArgumentException e){
            System.out.println(« [Warning] Could not get myInt »);
        }
    }
    public int getInt(String str) throws IllegalArgumentException {
        int myInt = Integer.parseInt(str);
        ... // do other stuff
        return myInt;
    }
}
```

Exception handling

- **`finally`** block
  - Code qui est toujours exécuté (avec ou sans exception)

FileManip.java

```
public class FileManip{
   public static void main(String[] args) {
      try {
         ... // Open files and manipulate them
      }
      catch (Exception e){
         ... // Log the exceptions
      }
      finally {
         ... // Close files
      }
   }
}
```



QCM0

QCM1.java

```
public class QCM1{
   public static void main(String[] args) {
      try {
         double d = Double.parseDouble(« 12.5 »);
      }
      catch (Exception e){
         System.out.print(« 1 »);
      }
      finally {
         System.out.print(« 2 »);
      }
      System.out.print(« 3 »);
   }
}
```

```
$ javac QCM1.java
$ java QCM1
```

a)
b) 1
c) 2
d) 3
e) 12
f) 13
g) 23
h) 123

QCM1

QCM1.java

```
public class QCM1{
    public static void main(String[] args) {
        try {
            double d = Double.parseDouble(« 12.5 »);
            return;
        }
        catch (Exception e){
            System.out.print(« 1 »);
        }
        finally {
            System.out.print(« 2 »);
        }
        System.out.print(« 3 »);
    }
}
```

```
$ javac QCM1.java
$ java QCM1
```

a)
b) 1
c) 2
d) 3
e) 12
f) 13
g) 23
h) 123

QCM2

QCM2.java

```
public class QCM2{
    public static void main(String[] args) {
        System.out.print(quizz());
    }

    public static String quizz() {
        try {
            return « a »;
        }
        catch (Exception e){
            return « b »;
        }
        finally {
            return « c »;
        }
        return « d »;
    }
}
```

```
$ javac QCM2.java
$ java QCM2
```

a) a
b) b
c) c
d) d

Checked vs. Unchecked Exceptions

- Checked : L'exception est vérifié à la compilation

FileManip.java

```
import java.io.FileReader;

public class FileManip{
    public static void main(String[] args) {
        readFile(« myFile.txt »);
    }

    private static void readFile(String fileName) {
        FileReader reader = new FileReader(fileName);
    }
}
```

Unhandled exception
type FileNotfound

Deux façons de remédier l'erreur:

Checked vs. Unchecked Exceptions

- Checked : L'exception est vérifié à la compilation

FileManip.java

```
import java.io.FileReader;

public class FileManip{
   public static void main(String[] args) {
       readFile(« myFile.txt »);
   }

   private static void readFile(String fileName) {
      try {
          FileReader reader = new FileReader(fileName);
      }
      catch (FileNotFoundException e){
          System.out.print(« Hey, I cannot find this file »);
      }
   }
}
```

Deux façons de remédier l'erreur:

1. **try/catch**

Checked vs. Unchecked Exceptions

- Checked : L'exception est vérifié à la compilation

FileManip.java

```
import java.io.FileReader;

public class FileManip{
   public static void main(String[] args) {
       readFile(« myFile.txt »);
   }

   private static void readFile(String fileName) throws
FileNotFoundException {
       FileReader reader = new FileReader(fileName);
   }
}
```

Unhandled exception
type FileNotfound

Deux façons de remédier l'erreur:

1. try/catch
2. **throws**

Checked vs. Unchecked Exceptions

- Checked : L'exception est vérifié à la compilation

FileManip.java

```
import java.io.FileReader;

public class FileManip{
   public static void main(String[] args) throws FileNotFoundException
   {
       readFile(« myFile.txt »);
   }

   private static void readFile(String fileName) throws
FileNotFoundException {
       FileReader reader = new FileReader(fileName);
   }
}
```

Erreur de compilation corrigé, mais quand même pas top…

Deux façons de remédier l'erreur:

1. try/catch
2. **throws**

Checked vs. Unchecked Exceptions

- Checked : L'exception est vérifié à la compilation

FileManip.java

```
import java.io.FileReader;

public class FileManip{
   public static void main(String[] args) {
      try {
         readFile(« myFile.txt »);
      }
      catch (FileNotFoundException e){
         System.out.print(« Hey, I cannot find this file »);
      }
   }

   private static void readFile(String fileName) throws
FileNotFoundException {
       FileReader reader = new FileReader(fileName);
   }
}
```

Deux façons de remédier l'erreur:

1. try/catch
2. **throws**

Checked vs. Unchecked Exceptions

- Unchecked : L'exception est vérifié seulement à l'exécution

Unchecked.java

```
public class Unchecked{
    public static void main(String[] args) {
        String str = null;
        printLength(str);
    }

    private static void printLength(String str) {
        System.out.println(str.length());
    }
}
```

Ce code compile

… mais crash à l'exécution

Checked vs. Unchecked Exceptions

- Unchecked : L'exception est vérifié seulement à l'exécution

Unchecked.java

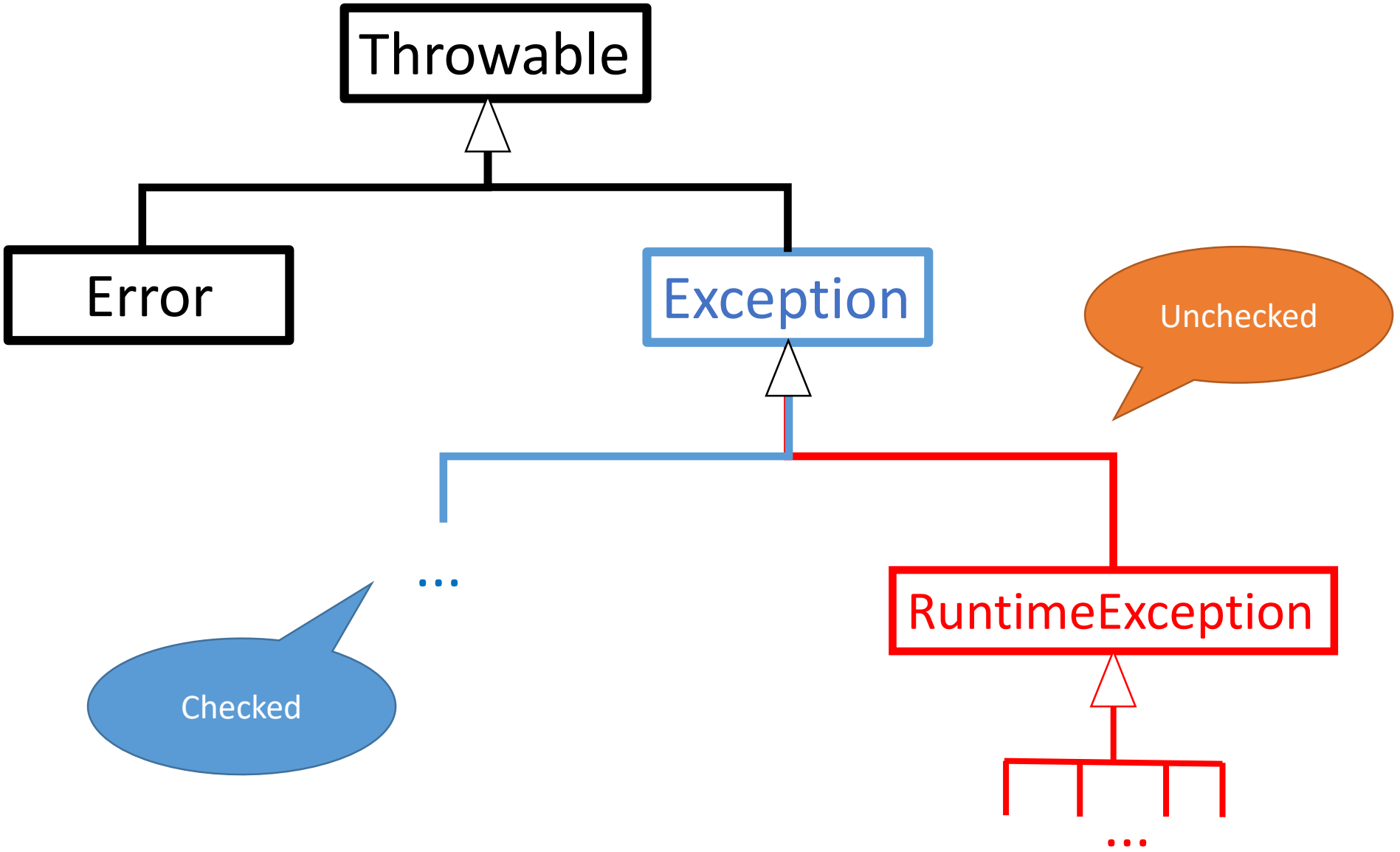
```
public class Unchecked{
    public static void main(String[] args) {
        String str = null;
        printLength(str);
    }

    private static void printLength(String str) {
        try {
            System.out.println(str.length());
        }
        catch (NullPointerException e){
            System.out.print(« Hey, String cannot be null here. »);
        }
    }
}
```

Ce code compile

… et se comporte comme attendu à l'exécution

Hiérarchie des exceptions en java



Exceptions customisées

Person.java

```
public class Person {
  int age;

  public void setAge(int newAge) throws NegativeAgeException {
    if (newAge < 0.0) {
      throw new NegativeAgeException(«Age cannot be negative.»);
    }
    age = newAge;
  }
}
```

Checked exception → Doit être traité quelque part !

Que se passe t-il si aucun message n'est passé en argument ?

NegativeAgeException.java

```
class NegativeAgeException extends Exception {
  public NegativeAgeException (String msg) {
     super(msg);
  }
}
```

Comment rendre cette exception « unchecked »?

TP 6

- Contenu:
  - ✓ Tout

- Aller voir *https://perso.isima.fr/loic/java/tp_06.php*