UPMC
SORBONNE UNIVERSITÉS

SORBONNE UNIVERSITÉS

Rappels, les exceptions en Java

F. Kordon - Université P. & M. Curie - CC2016

Fabrice.Kordon@lip6.fr

LIP6

Exceptions en Java

Prédéfinies

- Signification liée à la sémantique du langage
- Pas besoin de déclaration
- Deux types d'exceptions (via deux classes)
  - **Exceptions (java.lang.Exception)**
    - ➥ **«catch or specify» obligatoire**
  - **RuntimeExceptions (java.lang.RuntimeException)**
    - ➥ **inutile de spécifier qu'elle peut-être levée**

Rattrapage sous la forme

```
try {
// séquence d'instruction pouvant lever l'exception
}
catch (monException instanceException) {
// séquence d'instruction si l'exception est levée
} // On peut "cascader" les "catch" qui se rapportent au dernier "try"
```

Définies par le programmeur

- Extension de la classe dédiée : **Exception**

Quelques exceptions prédéfinies en Java

RuntimeException

- NullPointerException
  - Accès via une référence nulle (accès à un membre)
- ArrayIndexOutOfBoundsException / StringIndexOutOfBounds
  - Indice de tableau ou de chaîne en dehors des bornes
- NegativeArraySizeException
  - Création d'un tableau de taille négative
- ArithmeticException
  - Division par zéro
- IllegalThreadStateException
  - Opération incompatible avec l'état de la thread courante

java.lang

- ClassNotFoundException
  - Classe à instancier non trouvée
- InterruptedException
  - Une thread a été interrompue (par une autre)

Exemple d'utilisation d'une Exception


```
class HttpException extends Exception {
    private int code;
    public HttpException(int code,String message) {
        super(""+code+" "+message);
        this.code=code;
    }
    public int getHttpCode() {
        return code;
    }
}
```

Exemple d'utilisation d'une Exception


```java
class HttpException extends Exception {
    private int code;
    public HttpException(int code,String message) {
        super(""+code+" "+message);
        this.code=code;
    }
    public int getHttpCode() {
        return code;
    }
}
```

```java
public class ExempleException1 {
    public void maMethode() throws HttpException {
        int x;

        x = 3;
        throw new HttpException(404, "Fichier non trouvé");
    }
    public static void main(String []args) {
        ExempleException1 e;

        e = new ExempleException1();
        try {
            e.maMethode();
        }
        catch (HttpException exc) {
            System.err.println("Une erreur a été déclarée de code " +
                               exc.getHttpCode());
        }
    }
}
```

```
$ java ExempleException1
Une erreur a été déclarée de code 404
```

En guise de conclusion…

Les exceptions sont des classes

Toutes les exceptions dérivent de

- Exception ou RuntimeException
- Rattrapage obligatoire ou pas
  - **En fonction de la classe parente**

Séquence protégée type

- Située dans un «bloc try»
- «Handlers» situés dans des «blocs catch» ou des «blocs finally»
  - **Catch, 0 ou 1 exécution**
  - **Finally, toujours une exécution**

Démarche requise «catch or specify»

- On traite l'exception ou on déclare qu'elle est propagée