Les Exceptions Java

Ahcene.Bounceur@univ-brest.fr

Notion d'exception

En Java, les erreurs se propagent
sous la forme d'exceptions

Une exception :

- est un objet, instance d'une classe d'exception
- provoque la sortie d'une méthode
- correspond à un type d'erreur
- contient des informations sur cette erreur

Classes d'exceptions

Une classe d'exception hérite de **java.lang.Throwable**

correspond à un type d'erreur
et encapsule les informations relatives à l'erreur

La méthode **public String getMessage()** retourne une chaîne de caractères décrivant l'exception

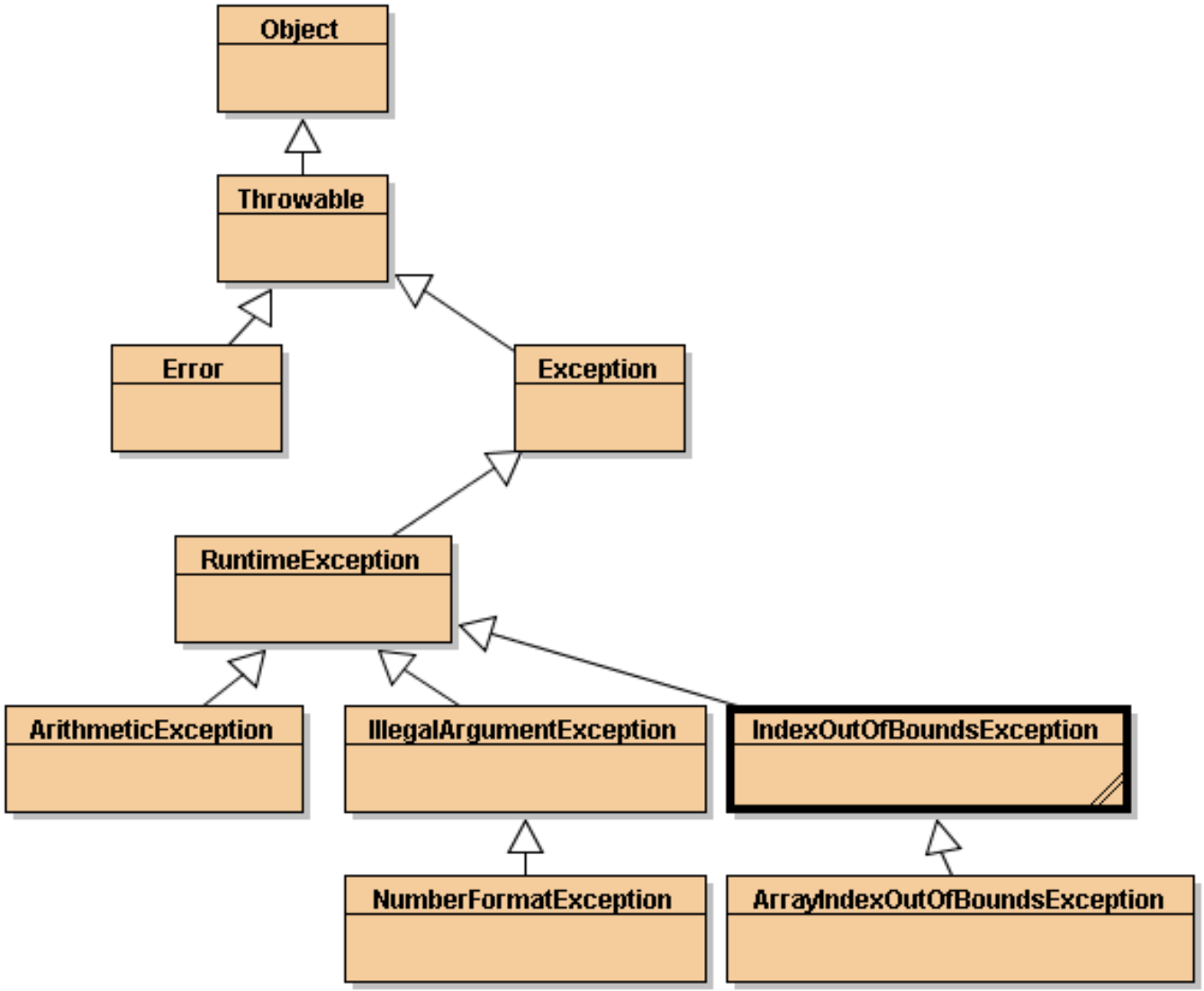
IndexOutOfBoundsException

NullPointerException

NumberFormatException

ClassCastException

Graphe d'héritage (partiel)



Anselme.Bouniol@univ-brest.fr

Déclaration des exceptions possibles

Chaque méthode doit déclarer les exceptions qu'elle peut émettre

La liste des exceptions possible est indiquée par **throws** suivi des classes d'exceptions concernées

```
public static int parseInt(String s) throws NumberFormatException {
        …
}
```

Émission et création d'exceptions

Lorsqu'une erreur est détectée dans une méthode
il faut :

- Créer un objet (de classe d'exception)
- Émettre cette exception

```
int getValue(int pos) throws IndexOutOfBoundsException {
    if (pos > tab.length)
      throw new IndexOutOfBoundsException("trop grand");
    else return tab[pos];
}
```

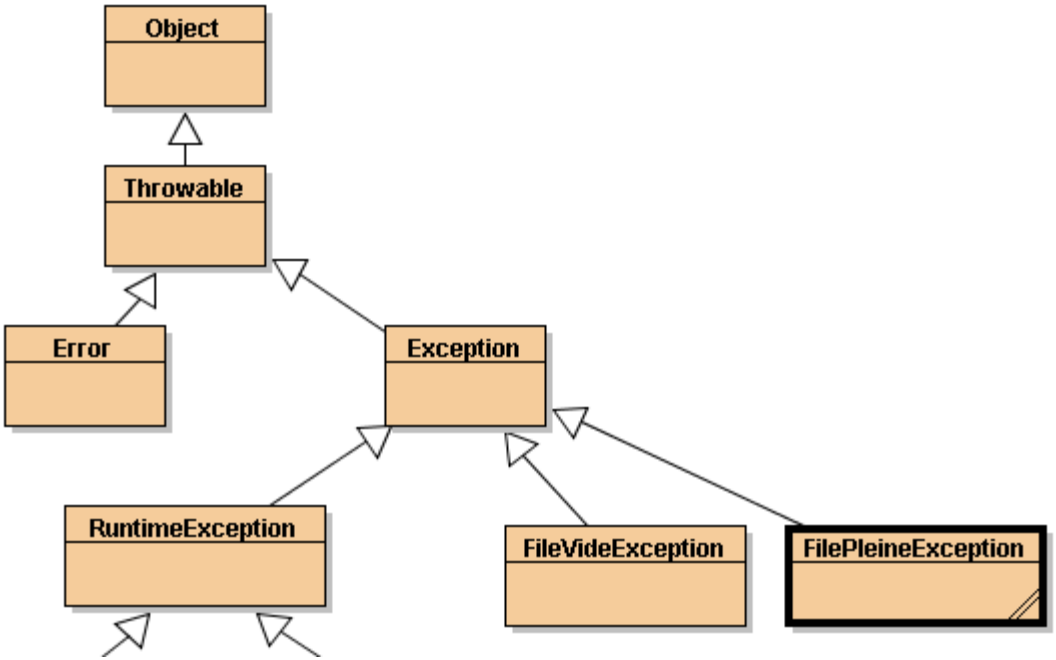
Autre exemple

```
class MonException extends Exception {
        public MonException() { }
        public MonException(String msg) { super(msg); }
}
…
public static void g() throws MonException {
        System.out.println("MonException levée dans g()");
        if (x=0)
                throw new MonException("démarrée en g()");
}
```

Nouvelles exceptions

public class FilePleineException extends Exception { … }

public class FileVideException extends Exception { … }



Traitement des exceptions

Si une méthode peut émettre une exception (ou appelle une autre méthode qui peut en émettre une)

il faut :

- soit **propager** l'exception

  (la méthode doit l'avoir déclarée)

- soit **intercepter** et traiter l'exception

Propagation d'exceptions

```
public int ajouter(int a, String str) throws NumberFormatException
    int b = Integer.parseInt(str);
    a = a + b;
    return a;
}
```

Throws précise les exceptions susceptibles d'être levées (ou propagées) par cette méthode

Interception d'exceptions

```
public int ajouter(int a, String str) {
   try {
       int b = Integer.parseInt(str);
       a = a + b;
   }
   catch (NumberFormatException e) {
       System.out.println(e.getMessage());
   }
   Finally {
       // bloc toujours exécuté
   }
}
```

try/catch n’est pas obligatoire

(Heureusement !)

pour toutes les classes dérivées

ArithmeticException, ArrayStoreException,
BufferOverflowException, BufferUnderflowException,
CannotRedoException, CannotUndoException,
ClassCastException, CMMException,
ConcurrentModificationException, DOMException,
EmptyStackException, IllegalArgumentException,
IllegalMonitorStateException, IllegalPathStateException,
IllegalStateException, ImagingOpException,
IndexOutOfBoundsException, MissingResourceException,
NegativeArraySizeException, NoSuchElementException,
NullPointerException, ProfileDataException,
ProviderException, RasterFormatException, SecurityException,
SystemException, UndeclaredThrowableException,
UnmodifiableSetException, UnsupportedOperationException