

# Cours Programmation Orientée Objets avec Java

---

## Gestion des exceptions

Stephane MALANDAIN, repris de Dr. Yassin REKIK

L'avenir est à créer

# Principe

---

- Une exception est une erreur se produisant dans un programme qui conduit le plus souvent à l'arrêt de celui-ci. Il vous est sûrement déjà arrivé d'obtenir un gros message affiché en rouge dans la console d'Eclipse : eh bien, cela a été généré par une exception... qui n'a pas été capturée .
- Le fait de gérer les exceptions s'appelle aussi « la capture d'exception ». Le principe consiste à repérer un morceau de code (par exemple, une division par zéro) qui pourrait générer une exception, de capturer l'exception correspondante et enfin de la traiter, c'est-à-dire d'afficher un message personnalisé et de continuer l'exécution.

# Exemple

## Code : Java

```
int j = 20, i = 0;  
System.out.println(j/i);  
System.out.println("coucou toi !");
```

Problems @ Javadoc Declaration Console X

```
<terminated> Test [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.03/bin/java (25 févr  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
at Test.main(Test.java:10)
```

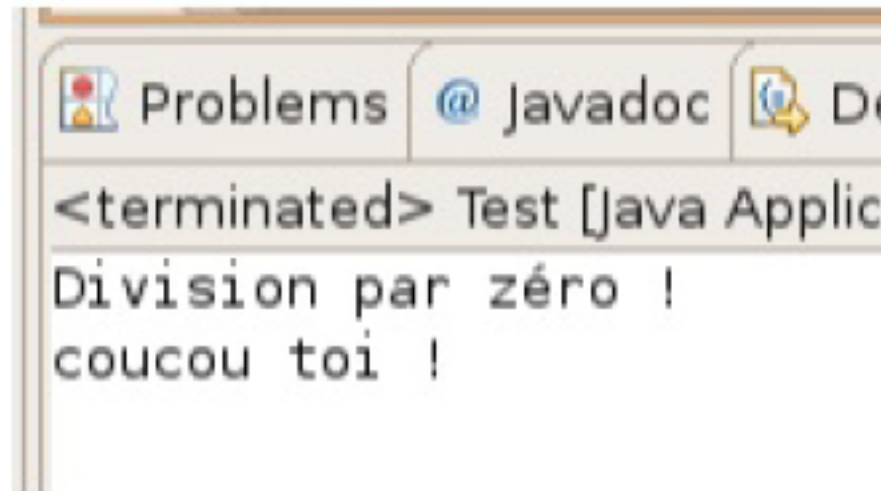
# Correction

---

- Lorsque l'exception a été levée, le programme s'est arrêté ! D'après le message affiché dans la console, le nom de l'exception qui a été déclenchée est `ArithmeticException` .
- Nous savons donc maintenant qu'une division par zéro est une `ArithmeticException` .
- Nous allons pouvoir la capturer, avec un bloc `try{...}` `catch{...}` , puis réaliser un traitement en conséquence.

## Code : Java

```
public static void main(String[] args) {  
  
    int j = 20, i = 0;  
    try {  
        System.out.println(j/i);  
    } catch (ArithmeticException e) {  
        System.out.println("Division par zéro !");  
    }  
    System.out.println("coucou toi !");  
}
```



# Explication

---

- Voyons un peu ce qui se passe :
  - Nous initialisons deux variables de type int , l'une à 0 et l'autre à un nombre quelconque.
  - Nous isolons le code susceptible de lever une exception :  
`System.out.println(j/i);` .
  - Une exception de type `ArithmeticException` est levée lorsque le programme atteint cette ligne.
  - Notre bloc `catch` contient justement un objet de type `ArithmeticException` en paramètre. Nous l'avons appelé `e` .
  - L'exception étant capturée, l'instruction du bloc `catch` s'exécute !
  - Notre message d'erreur personnalisé s'affiche alors à l'écran.

# Objet Exception e

---

## Code : Java

```
System.out.println("Division par zéro !" + e.getMessage());
```

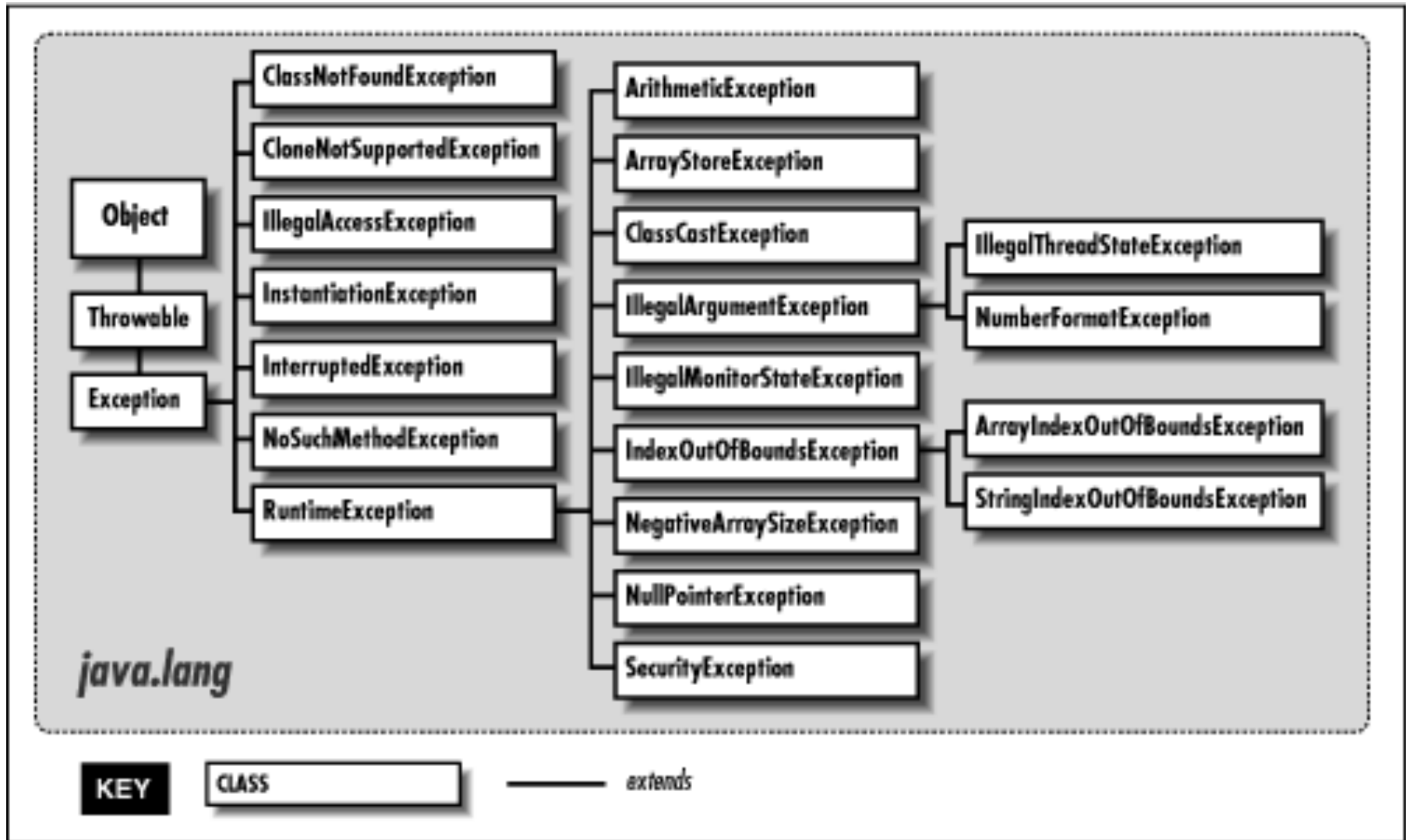
# La clause « finally »

## Code : Java

```
public static void main(String[] args) {  
    try {  
        System.out.println(" =>" + (1/0));  
    } catch (ClassCastException e) {  
        e.printStackTrace();  
    }  
    finally{  
        System.out.println("action faite systématiquement");  
    }  
}
```



# Exception de base en Java

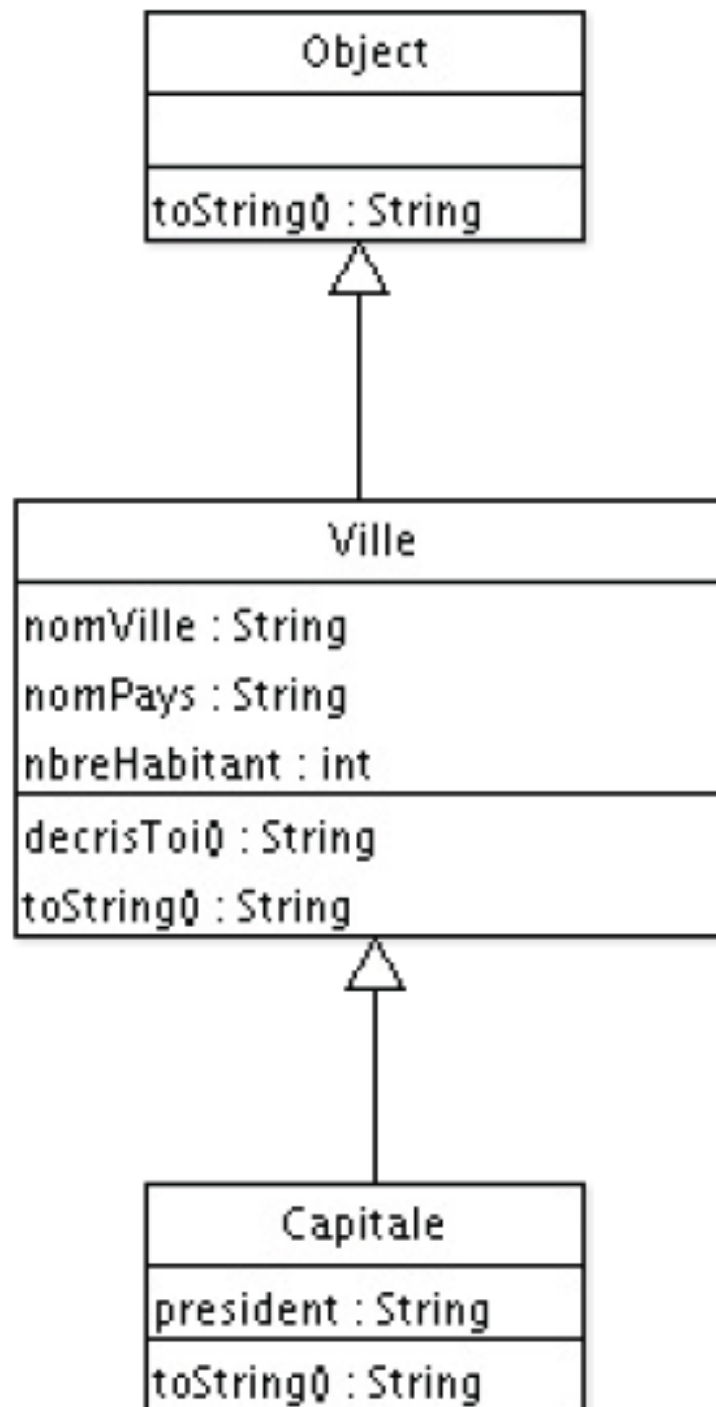


# Exceptions personnalisées

---

- En cas de nécessité, on peut créer ses propres exceptions.
- Elles descendent des classes Exception ou RuntimeException mais pas de la classe Error.
- Il est préférable (par convention) d'inclure le mot « Exception » dans le nom de la nouvelle classe

# Exemple



# Gestion d'exception personnalisée

---

- Nous allons perfectionner un peu la gestion de nos objets Ville et Capitale :
  - Nous mettre en œuvre une exception de notre cru afin d'interdire l'instanciation d'un objet Ville ou Capitale présentant un nombre négatif d'habitants.
- La procédure pour faire ça :
  - 1. créer une classe héritant de la classe Exception :  
NombreHabitantException (par convention, les exceptions ont un nom se terminant par « Exception ») ;
  - 2. renvoyer l'exception levée à notre classe  
NombreHabitantException ;
  - 3. ensuite, gérer celle-ci dans notre classe  
NombreHabitantException .

# Clauses throws et throw

---

- throws : ce mot clé permet de signaler à la JVM qu'un morceau de code, une méthode, une classe... est potentiellement dangereux et qu'il faut utiliser un bloc `try{...} catch{...}`. Il est suivi du nom de la classe qui va gérer l'exception.
- throw : celui-ci permet tout simplement de lever une exception manuellement en instanciant un objet de type **Exception** (ou un objet hérité). Dans l'exemple de notre **ArithmeticException**, il y a quelque part dans les méandres de Java un `throw new ArithmeticException()`.

# Déclarer l'exception

## Code : Java

```
class NombreHabitantException extends Exception{  
    public NombreHabitantException() {  
        System.out.println("Vous essayez d'instancier une classe Ville  
avec un nombre d'habitants négatif !");  
    }  
}
```

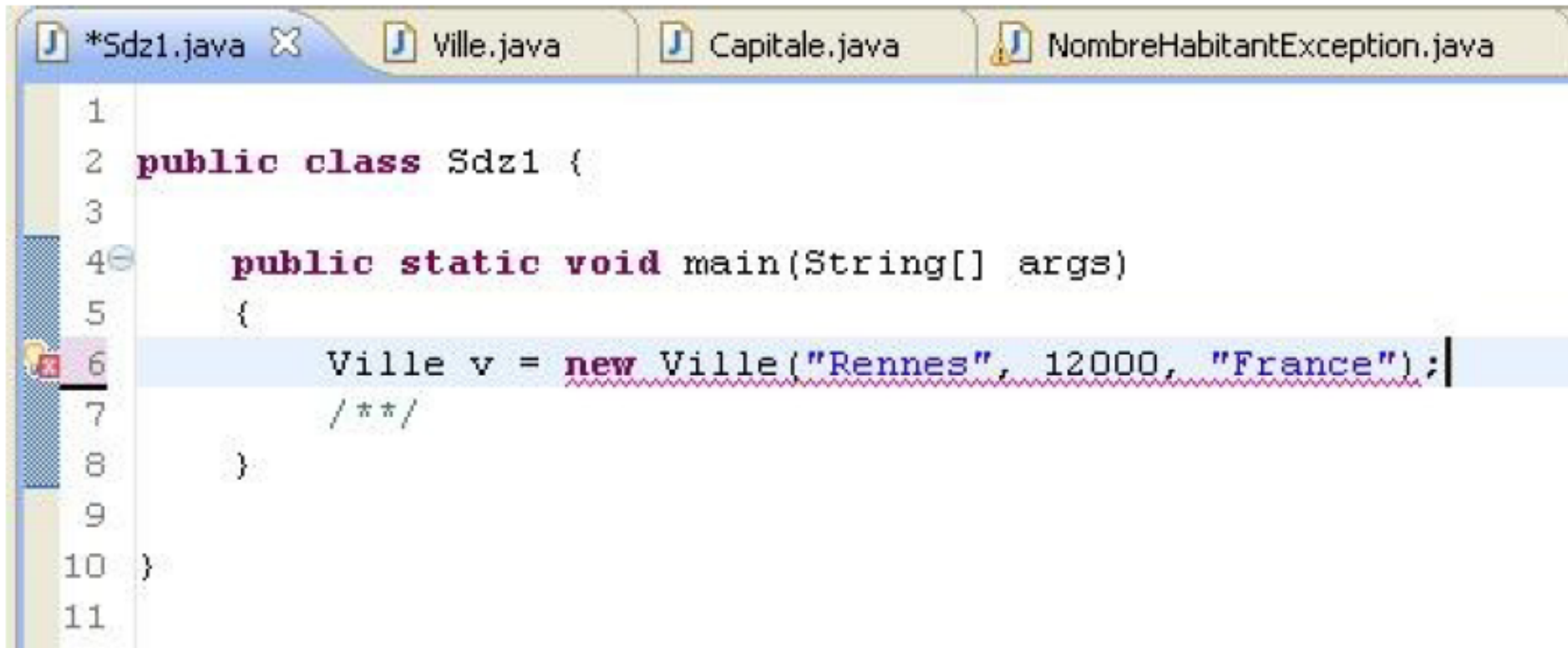
# Lever l'exception

## Code : Java

```
public Ville(String pNom, int pNbre, String pPays)
    throws NombreHabitantException
{
    if(pNbre < 0)
        throw new NombreHabitantException();
    else
    {
        nbreInstance++;
        nbreInstanceBis++;

        nomVille = pNom;
        nomPays = pPays;
        nbreHabitant = pNbre;
        this.setCategorie();
    }
}
```

# Tenir compte de l'exception



```
1  
2 public class Sdz1 {  
3  
4     public static void main(String[] args)  
5     {  
6         Ville v = new Ville("Rennes", 12000, "France");  
7         /**/  
8     }  
9  
10 }  
11
```



# Tenir compte de l'exception

```
1
2 public class Sdz1 {
3
4     public static void main(String[] args)
5     {
6         try {
7
8             Ville v = new Ville("Rennes", 12000, "France");
9             System.out.println(v.toString());
10
11         } catch (NombreHabitantException e) {}
12
13     }
14
15 }
```

Problems Javadoc Declaration Console

<terminated> sdz1 [Java Application] C:\Program Files\Java\jre1.6.0\_03\bin\javaw.exe (25 févr. 08 10:21:58)

Rennes est une ville de France, elle comporte : 12000 => elle est donc de catégorie : C

# Attention : visibilité

## Code : Java

```
public static void main(String[] args)
{
    try {
        Ville v = new Ville("Rennes", 12000, "France");
    } catch (NombreHabitantException e) { }

    System.out.println(v.toString());
}
```

# Solution intermédiaire

## Code : Java

```
public static void main(String[] args)
{
    Ville v = null;
    try {
        v = new Ville("Rennes", 12000, "France");
    } catch (NombreHabitantException e) { }

    System.out.println(v.toString());
}
```

Attention NullPointerException

# Solution complète

## Code : Java

```
public static void main(String[] args)
{
    Ville v = null;
    try {
        v = new Ville("Rennes", 12000, "France");
    } catch (NombreHabitantException e) { }
    finally{
        if(v == null)
            v = new Ville();
    }
    System.out.println(v.toString());
}
```

# Exception avec paramètres

## Code : Java

```
public NombreHabitantException(int nbre)
{
    System.out.println("Instanciation avec un nombre d'habitants négatif.");
    System.out.println("\t => " + nbre);
}
```

## Code : Java

```
public Ville(String pNom, int pNbre, String pPays)
    throws NombreHabitantException
{
    if(pNbre < 0)
        throw new NombreHabitantException(pNbre);
    else
    {
        //Le code est identique à précédemment
    }
}
```

# Gérer plusieurs exceptions

Code : Java

```
public class NomVilleException extends Exception {  
    public NomVilleException(String message) {  
        super(message);  
    }  
}
```

# Gérer plusieurs exceptions

## Code : Java

```
public Ville(String pNom, int pNbre, String pPays) throws
NombreHabitantException, NomVilleException
{
    if(pNbre < 0)
        throw new NombreHabitantException(pNbre);

    if(pNom.length() < 3)
        throw new NomVilleException("le nom de la ville est inférieur à
3 caractères ! nom = " + pNom);
    else
    {
        nbreInstance++;
        nbreInstanceBis++;

        nomVille = pNom;
        nomPays = pPays;
        nbreHabitant = pNbre;
        this.setCategorie();
    }
}
```

# Gérer plusieurs exceptions

## Code : Java

```
Ville v = null;

try {
    v = new Ville("Re", 12000, "France");
}

//Gestion de l'exception sur le nombre d'habitants
catch (NombreHabitantException e) {
    e.printStackTrace();
}

//Gestion de l'exception sur le nom de la ville
catch (NomVilleException e2) {
    System.out.println(e2.getMessage());
}
finally {
    if (v == null)
        v = new Ville();
}

System.out.println(v.toString());
```



# Le multi-catch

## Code : Java

```
public static void main(String[] args){
    Ville v = null;
    try {
        v = new Ville("Re", 12000, "France");
    }
    //Gestion de plusieurs exceptions différentes
    catch (NombreHabitantException | NomVilleException e2){
        System.out.println(e2.getMessage());
    }
    finally{
        if(v == null)
            v = new Ville();
    }
    System.out.println(v.toString());
}
```