

# Programmation objet

Gestion des exceptions

A decorative graphic element consisting of several horizontal lines of varying lengths and colors (red, white, and light blue) extending from the right side of the slide towards the center.

Principe

# Gestion des exceptions

- Une erreur déclenche une exception
- Une exception doit être gérée afin d'éviter la fin brutale du programme

# Intérêt de la gestion des erreurs

- Permet à l'utilisateur de savoir pourquoi le programme n'a pas fonctionné correctement
- Permet de personnaliser les messages d'erreurs affichés
- Fait partie de la création d'un programme de qualité

# Quels erreurs ?

- Erreur de programmation (boucle infinie)
- Erreur de saisie de l'utilisateur
- Erreur d'accès à une ressource (fichier, imprimante...)
- L'utilisation de certaines ressources oblige l'application à gérer les exceptions

# Quel impact ?

- Provoque un arrêt du programme
- Provoque un résultat faux dans le programme

# Exemple

- Programme qui
  - demande la saisie de 2 nombres
  - divise le premier par le deuxième
  - affiche le résultat.
- Si l'utilisateur entre 0 comme diviseur, le programme s'arrête sur une exception.
- Ce cas particulier peut être prévu et géré par le développeur

# Test du programme

- Créer un nouveau projet GestionException
- Intégrer le fichier GestionException.java
- Tester le programme avec un diviseur égal à 0

# La gestion des erreurs

- Ancienne méthode : tests dans le programme avec la structure if... else...

- Exemple :

```
If (nb != 0)
```

```
{
```

```
Operations suivantes à réaliser
```

```
}
```

```
Else
```

```
{
```

```
    Message d'erreur à afficher
```

```
    Autres opérations
```

```
}
```

# Les exceptions

# La gestion des exceptions

- Une erreur est une exception dans l'exécution du programme
  - Utilisation de structures permettant de capter une exception
  - Possibilité de définir une fin d'exécution du programme en cas d'exception
  - Possibilité de créer une exception
  - Possibilité de propager une exception

# Class Exception

- Une exception est un objet d'une classe Exception qui hérite obligatoirement de la classe Exception

## Method Summary

### Methods inherited from class `java.lang.Throwable`

`addSuppressed`, `fillInStackTrace`, `getCause`, `getLocalizedMessage`, `getMessage`, `getStackTrace`, `getSuppressed`, `initCause`, `printStackTrace`, `printStackTrace`, `printStackTrace`, `setStackTrace`, `toString`

### Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

# La gestion des erreurs en Java

- Plusieurs structures :
- Throw
  - Permet de déclencher une exception
- Try catch
  - Permet d'essayer le programme et réaliser un traitement en cas d'exception
- Try finally
  - Permet d'essayer le programme et de définir une fin différente en cas d'exception
- Try catch finally
  - Permet d'essayer le programme, de réaliser un traitement et de définir une fin différente en cas d'exception

# Principe

```
try
{
    Programme qui doit être exécuté
}
catch (nature de l'exception variableException)
{
    Gestion de l'exception
}
finally
{
    Programme qui doit être exécuté dans tous les cas
}
```

# Exemple d'exceptions

- `NullPointerException` : accès à un objet null
- `NumberFormatException` : erreur lors de la conversion d'une chaîne de caractère en nombre
- `ArrayOutOfBoundsException` : accès à une case inexistante d'un tableau

# Pour connaître l'exception levée

Déclencher l'exception lors de l'exécution

OU

Catch (Exception e)

{

*// affichage du nom de l'exception*

System.out.println(e);

*// affichage du message d'erreur de l'exception*

System.out.println(e.getMessage());

}

# Exercice

- Intégrer un bloc try-catch dans le programme pour afficher : Division par 0 interdite lorsque l'exception correspondante est levée

# Fin d'exécution du programme

- Pour définir des instructions à exécuter qu'il y ait eu ou non exception, on utilise le bloc finally
- Exemple :
  - affichage d'un message de fin du programme
  - Fermeture d'une ressource...

# Exercice

- Intégrer un bloc try-catch-finally pour afficher le message Fin du programme dans tous les cas (qu'il y ait eu une exception ou non)

# Options des exceptions

# Utilisation de plusieurs catch

- Lorsque l'on souhaite une gestion particulière pour plusieurs types d'exceptions, on peut enchaîner les catch

```
catch (ArithmeticException e)
{
    System.out.println("Division par zéro interdite");
}
catch (Exception e)
{
    System.out.println(e.getMessage());
}
```

# Utilisation de plusieurs catch

- L'ordre d'écriture des catch a une importance.
- Les Exceptions sont traitées dans l'ordre séquentiel. Dès qu'une exception est trouvée, elle est levée.

```
catch (ArithmeticException e)
{
    System.out.println("Division par zéro interdite");
}
catch (Exception e)
{
    System.out.println(e.getMessage());
}
```

# Exercice

- Ajouter dans le programme la gestion de l'exception levée lorsque l'utilisateur n'entre pas une valeur entière pour afficher le message :  
Veuillez saisir un entier
- Ajouter dans le programme la gestion de toute exception catch (Exception e) pour afficher l'exception (e)

# Catch de plusieurs exceptions

- Permet de réaliser le même traitement pour plusieurs exceptions

```
catch (ArithmeticException |  
    InputMismatchException e)  
{  
    System.out.println(e.getMessage());  
}
```

# Création d'exceptions personnalisées

- Possibilité de créer de nouvelles exceptions
- Création d'une classe qui hérite d'Exception
  - Le nom de la classe doit se terminer par Exception
- Le constructeur de la classe contient les actions à réaliser dans le cas de l'exception

# Déclencher une exception

- Throw permet de déclencher une exception
  - De java
  - Personnalisée
- A tout moment dans le programme, la commande `throw new nomException();` déclenche l'exception indiquée

# Exercice

- Créer une classe `NegatifException`. Son constructeur appellera le constructeur suivant de la classe `Exception` pour initialiser le message liée à l'exception avec la valeur : Nombres négatifs interdits

## Exception

```
public Exception(String message)
```

Constructs a new exception with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to `Throwable.initCause(java.lang.Throwable)`.

### Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `Throwable.getMessage()` method.

# Exercice

- Dans le programme, déclencher cette exception dans le cas où un des nombres entrés est négatif et faire afficher le message associé à l'exception

# Exception non gérées

- Une méthode qui peut déclencher une exception mais qui ne la gère pas (pas de bloc try... catch) doit le déclarer
- On rajoute le mot clé throws avec la liste des exceptions possibles lors de la création de la méthode

# Exception non gérée

- Création de la fonction

```
Public void nomFonction () throws nomException  
{  
}
```

- L'exception doit être gérée dans le programme qui appelle la fonction

# Exercice

- Créer une fonction void `divise()` qui demande la saisie de 2 nombres et affiche le résultat. Si un nombre est négatif, la fonction déclenche l'exception `NegatifException`
- Cette fonction ne gère pas la levée de l'exception mais la propage (`throws NegatifException`)

# Exercice

- Dans le main, appeler la fonction `divise()` dans un bloc `try-catch` qui gère l'exception `NegatifException`

# Jeu des héros - Exception

# Exercice

- Dans les classes du jeu des héros, ajouter la gestion des exceptions suivantes :
  - `NullPointerException` dans la méthode `afficheResultat` de la classe `Heros`
  - `NullPointerException` dans la méthode `attaque` de la classe `Joueur`
  - Toutes les exception dans les méthodes de création des héros ou des joueurs.

# Exercice

- Créer une classe d'exception HerosException
- L'exception HerosException sera déclenchée dans la méthode combat si le héros attaquant n'a plus de point de vie et propagée
- L'exception sera gérée dans le main avec l'affichage du message : Attaque impossible