



JAVA – EXCEPTIONS

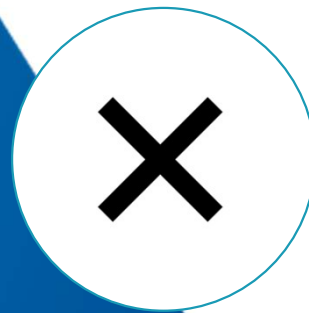
Wijin
v1.0 2024



MENU

- Gestion des exceptions dans Java
- Création de ses propres exceptions
- Utilisation du try with resource

JAVA - GESTION DES EXCEPTIONS DANS JAVA



Java – Exception ?

- Sujet **important** permettant la **gestion des erreurs**
- Dans un langage non objet, on va traiter les erreurs de cette façon (par exemple) :

```
retour = fonction1(parametreY);  
si (retour == erreur) {  
    traiter l'erreur;  
}
```

On traite l'erreur après l'instruction
qui peut poser problème



Java – Exception ?

- Sujet **important** permettant la **gestion des erreurs**
- En langage objet, c'est différent :
 - Gestion des **exceptions**



Java – Exception ?

```
1 package exceptions;
2
3 public class Tableaux {
4
5     private String[] tableau = {"Ain", "Aisne", "Allier"};
6
7     public void afficheTableau() {
8         for (int i = 0; i < 5; i++) {
9             System.out.println(tableau[i]);
10        }
11    }
12 }
```

Va provoquer une exception

```
1 package exceptions;
2
3 public class Programme {
4
5     public static void main(String[] args) {
6
7         Tableaux tableau1 = new Tableaux();
8         tableau1.afficheTableau();
9
10    }
11
12 }
```

Faites le test !



Java – Exception ?

Exception « levée »

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
    at exceptions.Tableaux.afficheTableau(Tableaux.java:9)
    at exceptions.Programme.main(Programme.java:8)
```

On peut voir la « stack trace » qui a conduit à l'exception (pile des instructions exécutées) :
À lire du bas vers le haut

Arrêt du programme ! ☹



Java – Exception ?

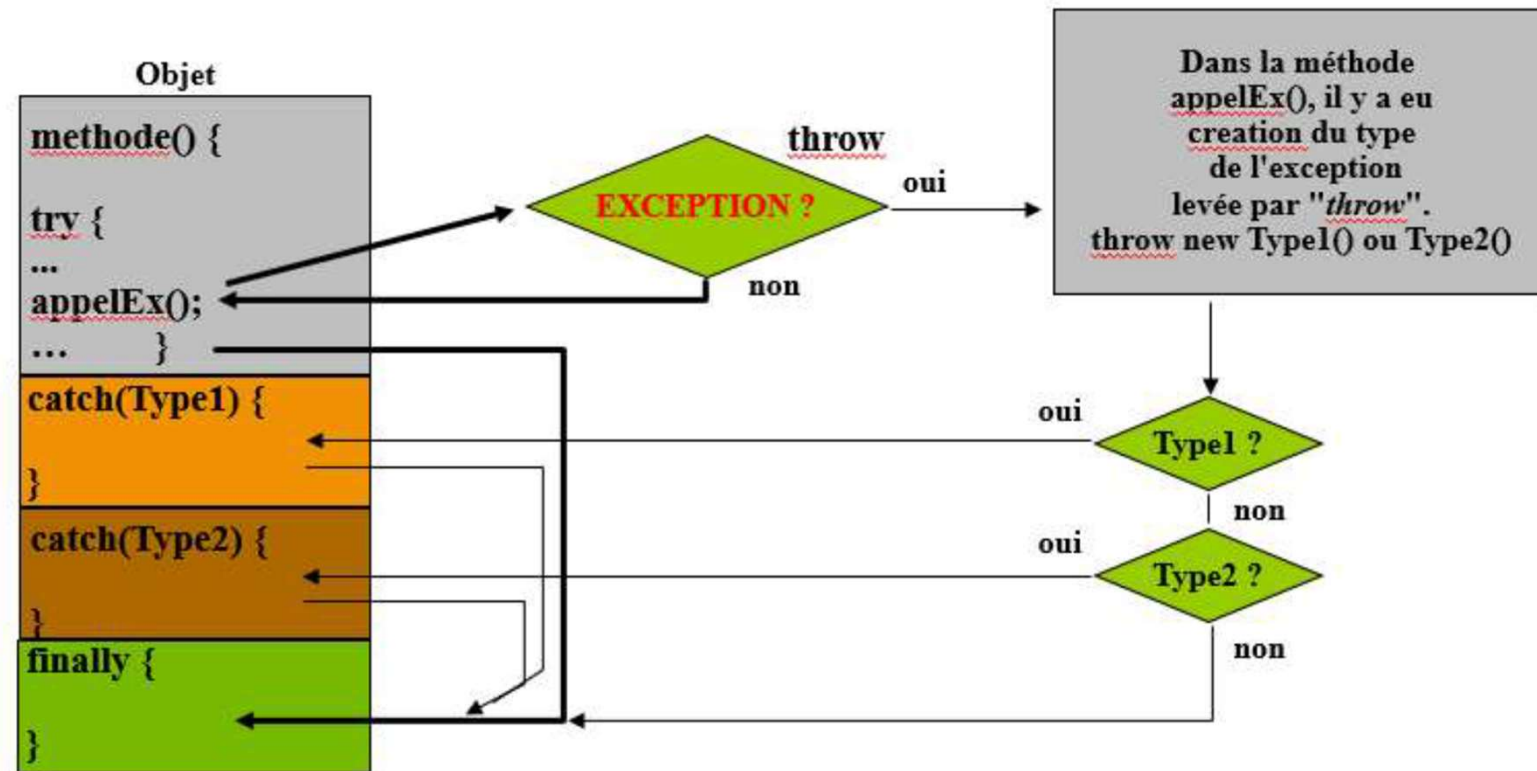
- **Objectif** : gérer l'exception, afin de terminer proprement le programme
- Trois choses :
 - Une exception est une **classe**, instanciée pour produire un **objet** exception
 - Il faut informer le système de ce qui **déclenche** la levée d'une exception
 - Il faut savoir **recupérer** et **traiter** une exception

Mot-clé throw

Bloc *try-catch*



Java – Exception ?



Java – Exception

- **Important : TOUTES** les exceptions levées dans une méthode, et qui n'ont pas été traitées localement, **DOIVENT** être ajoutées à la signature de la méthode
- Cela oblige l'appelant à **traiter** ces exceptions ou les **faire remonter** également à l'appelant



Mot-clé **throws**



Java – Exception

```
1 package exceptions;
2
3 public class Tableaux {
4
5     private String[] tableau = {"Ain", "Aisne", "Allier"};
6
7     public void afficheTableau() throws ArrayIndexOutOfBoundsException {
8         for (int i = 0; i < 5; i++) {
9             System.out.println(tableau[i]);
10        }
11    }
12 }
```

Mot-clé throws

```
1 package exceptions;
2
3 public class Programme {
4
5     public static void main(String[] args) {
6
7         Tableaux tableau1 = new Tableaux();
8         try {
9             tableau1.afficheTableau();
10        } catch (ArrayIndexOutOfBoundsException e) {
11            System.out.println("Erreur lors de l'affichage du tableau : " + e.getMessage());
12        }
13    }
14 }
15
16 }
```

Bloc try-catch

Il s'agit d'une exception levée directement par le système



Java – Exception

```
1 package exceptions;
2
3 public class Tableaux {
4
5     private String[] tableau = {"Ain", "Aisne", "Allier"};
6
7     public void afficheTableau() throws Exception {
8         for (int i = 0; i < 5; i++) {
9             if (i > tableau.length - 1) {
10                 throw new Exception("Indice en dehors des bornes du tableau : " + i);
11             }
12             System.out.println(tableau[i]);
13         }
14     }
15 }
```

Il est possible
pour le
programmeur
de lever une
exception

```
1 package exceptions;
2
3 public class Programme {
4
5     public static void main(String[] args) {
6
7         Tableaux tableau1 = new Tableaux();
8         try {
9             tableau1.afficheTableau();
10        } catch (Exception e) {
11            System.out.println("Erreur lors de l'affichage du tableau : " + e.getMessage());
12        }
13
14    }
15
16 }
```

Bloc try-catch

Il s'agit d'une
exception levée par
la méthode
afficheTableau



Java – Exception

- **Synthèse :**

- La signature d'une méthode doit inclure la déclaration des diverses exceptions levées dans le code, et non récupérées localement, et ceci se réalise avec l'instruction **throws**
- Lever une Exception se fait avec l'instruction **throw new <Type-Exception>**
- La récupération se fait en encapsulant le code risquant de monter une Exception avec **try/catch (Exception)**



Java – Exception

Synthèse :

TRAITEMENT D'UNE EXCEPTION :

```
try
{
    // instructions
}
catch(ExceptionType1 e) {
    // actions si une exception a été lancée
}
finally
{
    // nettoyage si nécessaire
}
```

LEVÉE D'UNE EXCEPTION :

```
public String methode() throws ExceptionType1{
    if (...) {
        throw new ExceptionType1("Message de l'exception");
    }
}
```



Java – Exception

- **Synthèse :**
 - Le bloc **try** peut être suivi d'un bloc **finally**, qui permet d'exécuter du code dans tous les cas, qu'il y ait eu exception ou non
 - Extrêmement utile pour **libérer des ressources** par exemple mémoire ou fichier, que l'on doit libérer dans tous les cas



Java – Exception

- **Cas avec levée de plusieurs exceptions :**
 - Il est possible de déclarer la levée de plusieurs exceptions dans une méthode
 - Plusieurs **throw new** ...
 - Signature de méthode avec plusieurs exception (**throws** Exception1, Exception 2, ...)



Java – Exception

- **Cas avec levée de plusieurs exceptions :**
 - Il est possible de traiter plusieurs exceptions dans un même bloc **try-catch**

```
try {  
    Integer retour = faireQuiMontePlusieursException();  
}  
catch (ExceptionType1 e) {  
    // Faire traitement 1  
}  
catch (ExceptionType2 e){  
    // Faire traitement 2  
}
```



Java – Exception

- **Cas avec levée de plusieurs exceptions :**
 - Il est possible de traiter plusieurs exceptions dans un même bloc try-catch
 - Alternative (avec *instanceof*) :

```
    } catch (Exception e) {  
        if(e instanceof ExceptionType1){  
            // Traitement  
        } else if (e instanceof ExceptionType2){  
            //Autre traitement  
        }  
    }
```



JAVA - CRÉER SES PROPRES EXCEPTIONS



Java – Créer ses propres exceptions

- Pour des besoins applicatifs **spécifiques**, Il est possible de créer sa propre gestion d'exceptions en Java
- Les classes d'exception personnalisées doivent hériter de la classe **Exception**, ou d'une classe qui **hérite** de la classe **Exception**
- La gestion de ces exceptions est **identique**



Java – Créer ses propres exceptions

```
1 package exceptions;
2
3 public class ExceptionDecrochage extends Exception {
4
5     private Integer vitesse;
6
7     public ExceptionDecrochage(Integer vitesse) {
8         this.vitesse = vitesse;
9     }
10
11     public String getRaison() {
12         return "Changez l'inclinaison ou augmentez la vitesse. Vitesse actuelle = " + this.vitesse;
13     }
14
15 }
```

Exception personnalisée
pour décrire le décrochage
d'un avion

Constructeur permettant de
stocker la vitesse de l'avion

Méthode retournant un
message d'alerte



Java – Créer ses propres exceptions

```
1 package exceptions;
2
3 public class Avion {
4
5     private Boolean enVol;
6     private Integer vitesseDecrochage;
7
8     public Avion(Boolean enVol, Integer vitesseDecrochage) {
9         this.enVol = enVol;
10        this.vitesseDecrochage = vitesseDecrochage;
11    }
12
13    public String voler(Integer vitesse) throws ExceptionDecrochage {
14
15        if (this.enVol && vitesse < this.vitesseDecrochage) {
16            throw new ExceptionDecrochage(vitesse);
17        }
18
19        return this.enVol?"L'avion vole à " + vitesse:"L'avion ne vole pas";
20    }
21
22 }
```

Classe Avion

2 propriétés :

- *enVol* pour l'état de l'avion
- *VitesseDecrochage* pour la vitesse minimal avant décrochage

Constructeur

Méthode « voler » avec levée de l'exception personnalisée



Java – Créer ses propres exceptions

```
1 package exceptions;
2
3 public class Pilote {
4
5     public static void main(String[] args) {
6         Avion monAvion = new Avion(true, 200);
7         String message = null;
8
9         try {
10             message = monAvion.voler(210);
11         } catch (ExceptionDecrochage exception) {
12             message = exception.getRaison();
13         } finally {
14             System.out.println(message);
15         }
16     }
17 }
18
19 }
```

Classe Pilote

On instancie un avion

On fait voler l'avion à une certaine vitesse

On attrape l'exception personnalisée :
Affichage du message (*raison*)

Affichage du message qu'il y ait eu une exception ou non



JAVA - TRY WITH RESOURCES



Java – try with resources

- Disponible depuis Java 7, il est possible de déclarer en paramètre du **try** les ressources à utiliser dans ce bloc **try**
- **Avantage** : ces ressources sont **libérées automatiquement** par Java en sortie du bloc **try** !
 - Et ce quelque soit le résultat du traitement dans le bloc **try** (levée d'exception ou non)



Java – try with resources

```
3 public class TestException {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         while (sc.hasNextLine()) {
7             String chaine = sc.nextLine();
8             String[] chaines = chaine.split(",");
9             Voiture v1 = new Voiture(chaines[0], Integer.parseInt(chaines[1]));
10            try {
11                System.out.println(v1.rouler(Integer.parseInt(chaines[2])));
12            } catch (GrandeVitesseException e) {
13                System.out.println(e.getRaison());
14            } finally {
15            }
16        }
17        sc.close();
18    }
19 }
```

Sans try with resources

Obligation de fermer explicitement la ressource (libération)



Java – try with resources

```
3 public class TestException {  
4     public static void main(String[] args) {  
5         try (Scanner sc = new Scanner(System.in)) {  
6             while (sc.hasNextLine()) {  
7                 String chaine = sc.nextLine();  
8                 String[] chaines = chaine.split(",");  
9                 Voiture v1 = new Voiture(chaines[0], Integer.parseInt(chaines[1]));  
10                try {  
11                    System.out.println(v1.rouler(Integer.parseInt(chaines[2])));  
12                } catch (GrandeVitesseException e) {  
13                    System.out.println(e.getRaison());  
14                } finally {  
15                }  
16            }  
17        }  
18    }  
19 }
```

Avec try with resources

Aucune libération explicite : Java s'en charge



JAVA - QUIZ



Java – Quiz

- Comment déclare t-on à la signature d'une méthode qu'elle peut lever une exception de type Exception ?
 - catch (Exception)
 - throws Exception
 - throws new Exception
 - rien n'est obligatoire
 - finally



Java – Quiz

- Comment fait-on en Java pour attraper une exception levée dans une méthode que l'on appelle ?

- `try{ objet.methode(); } catch (Exception e) {`

- `Objet o = objet.methode(); if (o instanceof(Exception) {`
- `try{ objet.methode(); } finally (Exception e) {`
- `on catch {objet.methode(); } {`



Java – Quiz

- Quelle est l'utilité de **finally** dans le traitement d'une exception ?
 - De réaliser un traitement dans le cas où il n'y a pas eu d'exception de levée
 - De réaliser un traitement systématique, qu'il y ait eu ou non une exception de levée
 - De finaliser le traitement uniquement dans le cas où il ya eu une exception



Java – Quiz

- Quelle est la différence entre **throw** et **throws** ?
 - Les 2 syntaxes sont interchangeables
 - throw permet de remonter à l'appelant une Exception, throws permet de remonter au système une exception
 - throw permet de déclarer qu'une méthode est susceptible de monter une exception, throws permet de lever une exception
- throws permet de déclarer qu'une méthode est susceptible de monter une exception, throw permet de lever une exception



Java – Quiz

- Dans le code suivant :

try(Ressource r) { code } catch(Exception e) {...

- r est une ressource, mais n'est plus supportée depuis Java 8
- Cette syntaxe n'est pas valable

• r est une ressource qui est automatiquement libérée à la sortie du bloc

- r est l'objet unique qu'il est possible d'accéder dans le bloc try



MERCI POUR VOTRE
ATTENTION

Faites-moi part de vos remarques
concernant le cours afin qu'il soit
amélioré pour les prochaines
sessions : nicolas.sanou@wijin.tech