Exceptions en JAVA

Exceptions en JAVA

1. **Définitions:**

   - Soit le code suivant

```
int t[]= new int[5];


for( int i = 0; i <= t.length; i++)
      System.out.println(t[i]);
System.out.println("End of the program ");
```

   - Ce code génère une erreur à l'exécution
   - Le programme s'arrête et la ligne d'après n'est même pas exécutée

Exceptions en JAVA

1. **Définitions:**

   - On peut donc surveiller cette instruction et capturée l'exception qui est levée pour la traiter:

Exceptions en JAVA

1. **Définitions:**

```
int t[]= {3, 4, 6, 8, 10};
try{
    for( int i = 0; ; ++i)
        System.out.println(t[i]);
}
catch( IndexOutOfBoundsException iobe){
        // instruction;
}
```

Exceptions en JAVA

1. **Définitions:**

   - Une séquence d'instructions qui va lever une exception de type ArrayIndexOutOfBoundsException lorsque i va être égal à t.length.
   - Le programme continu et la ligne d'après est exécutée normalement.

Exceptions en JAVA

2. Traitement d'une exception: Exception multiple

- Les instructions surveillées par un bloc **try** peuvent lever plusieurs types d'exceptions et donc le bloc **try** peut être suivi de plusieurs catch. *(VOIR CODE)*
- Lorsque une exception est levée, les **catch** sont parcourus en **séquence** et le premier qui capture l'exception du type de celle qui a été levée, est celui qui traite l'exception.
- Pour un bloc **try** suivi de plusieurs catch, s'il existe une hiérarchie dans les classes Exception des différents catch , les classe parents doit apparaitre après les classes filles . En revanche cette dernière caché.

Exceptions en JAVA

2. Traitement d'une exception: Exception multiple

- Exemple:
  - multipleCatchBlocs

Exceptions en JAVA

```
try{
    int [] array = { 10, 0, 8, 3, 5};
    int r ;
    r= array[0]/array[1];
    System.out.println(r);
    System.out.println(array[5]);
}
catch( ArithmeticException ae){
    System.out.println(ae.getMessage());
}
catch(IndexOutOfBoundsException iobe){
    System.out.println(iobe.getMessage());
}
```

Exceptions en JAVA

2. Traitement d'une exception: try /catch imbriqué

- Exemple : nestedTryCatchbloc

Exceptions en JAVA

2. Traitement d'une exception: Bloc finaly

- Un bloc finally peut éventuellement suivre le bloc catch.
- Le bloc finally est exécuté **impérativement** soit l'exception a été levée ou non
- S'il y a un bloc finally, il est exécuté après toutes les instructions du bloc try, ou après la dernière instruction qui a traité une exception levée dans le bloc try.

Exceptions en JAVA

2. Traitement d'une exception: Bloc finaly

- Il est aussi exécutée si un **catch** ou le bloc **try** se termine par un **return**, un **break** ou un **continue**.
- Exemple:

Exceptions en JAVA

2. Traitement d'une exception: Bloc finaly

```
try{
    int [] array = { 10, 0, 8, 3, 5};
    int r = array[0]/array[1];
    System.out.println(r);
}
catch(ArithmeticException ae){
    System.out.println(ae.getMessage());
}
finally{
 System.out.println("fin du programme");
}
```

Exceptions en JAVA

3. Hiérarchie des exceptions

- Si Exception2 **dérive** de Exception1 le catch de Exception2 doit se trouver avant le catch de Exception1.

Exceptions en JAVA

3. Hiérarchie des exceptions

```
try{
        I1 ;
        I2 ;
        I3 ;
}
catch(Exception2 e2){ T2; }
catch(Exception1 e1){ T1 }
finally{ F1; }
InstructionSuivantes ;
```

Si Exceptione2 dérive de Exception1, Exception2 doit apparaitre dans le 1er catch et ainsi de suite

Exemple: hiearchicalException

Exceptions en JAVA

5. Exceptions Vérifiées (Checked) vs exceptions non vérifiées (unchecked)

- Les exceptions peuvent être divisées en 2 catégories:
  - **Vérifiées (Checked)** classes et **non vérifiées (unchecked)** classes
  - Les exceptions vérifiées doivent être gérées par un **try** / **catch**, le compilateur Java vous oblige à écrire **try** and **catch**.
  - Les exceptions non vérifiées ne doivent pas être gérées par un **try/catch**
  - Seules **RuntimeException** sont les exceptions non vérifiées.

Exceptions en JAVA

5. Exceptions Vérifiées (Checked) vs exceptions non vérifiées (unchecked)

- Exemple:

```
int a=10;
int b=0;
//unchecked exception
System.out.println(a/b);
//checked exception
try{
   FileInputStream file = new FileInputStream("my.txt");
}
catch (FileNotFoundException fnfe){
   System.out.println(fnfe.toString());
}
```

- Erreur de compilation!!!

Exceptions en JAVA

5. Les méthodes de la classe Exception:

getMessage()

toString()

getCause()

printStackTRace()

- Exemple

Exceptions en JAVA

5. Les méthodes de la classe Exception:

Exemple

```
try{
   int a=10;
   int b=0;
   System.out.println(a/b);
}
catch (Exception e){
   //System.out.println(e.getMessage());
   //System.out.println(e.toString());
   System.out.println(e);// automatically call toString method
   //e.printStackTrace();
}
```

Exceptions en JAVA

6. **throw** et throws

- Le mot clé **throw** est utilisé pour lever une exception **logique**.
- Une seul exception peut être levée a la fois en utilisant le mot clé throw.
- Il est utilisé **uniquement** à l'intérieur d'une méthode.
- Il est suivi d'une variable d'instance par ex: **new Exception()**.

Exceptions en JAVA

6. throw et throws

- Le mot clé **throws** est utilisé pour déclarer qu'une méthode peut lever une exception
- Il est écrit dans la signature d'une methode.
- Exemple:

Exceptions en JAVA

6. throw et throws

- Exemple:

```
public class Main {
   static  int area(int l , int b){
      return l * b;
   }
   static void meth1(){
      System.out.println("Area is "+ area(10,5));
   }
   public static void main(String[] args) {
      meth1();
   }
}
```

Exceptions en JAVA

6. throw et throws

- Exemple:

```
public class Main {
   static  int area(int l , int b){
      return l * b;
   }
   static void meth1(){
      System.out.println("Area is "+ area(-10,5));
   }
   public static void main(String[] args) {
      meth1();
   }
}
```

Qu'est ce qui se passe si on fait appel à la méthode area avec des paramètres négatifs <0?

Nous devons empêcher le calcul de la surface si l'un des paramètres l ou b est <0.

Comment?

Exceptions en JAVA

6. throw et throws

- Exemple:

```
static  int area(int l , int b) throws
Exception{
   if (l<0 || b <0)
      throw new Exception();
   return l * b;
}
```

Comment?
Nous devons lever une exception pas envoyer un résultat erroné !!

Exceptions en JAVA

6. throw et throws

- Exemple:

Exceptions en JAVA

6. throw et throws

- Exemple:

Mais maintenant l'erreur s'est déplacée vers la méthode meth1?

```
static void meth1(){
    System.out.println("Area is "+ area( l: 10, b: 5));
}
```

Solution?

Exceptions en JAVA

6. throw et throws

- Exemple:

Mais maintenant l'erreur s'est déplacée vers la méthode main?

```
static void meth1() throws Exception{
    System.out.println("Area is "+ area( l: 10, b: -5));
}
public static void main(String[] args) {
        meth1();
}
```

Solution?

Exceptions en JAVA

6. throw et throws

- Exemple:

Solution?

```
public static void main(String[] args) throws Exception {
        meth1();
}
```

Run:

```
Exception in thread "main" java.lang.Exception Create breakpoint
    at Main.area(Main.java:7)
    at Main.meth1(Main.java:11)
    at Main.main(Main.java:14)
```

On obtient le message d'erreur de départ
=> solution: try /catch

Exceptions en JAVA

6. throw et throws

- Exemple:

Solution?

```
public static void main(String[] args {
        try{
            meth1();
        }
        catch(Exception e){
            System.out.println(e);
        }
}
```

Exceptions en JAVA

7. Ecrire sa propre classe d'exception

Pour écrire sa propre classe d'exception , il suffit d'étendre la classe Exception de comme ceci:

```
class NegativeDimensionException extends Exception
{
   @Override
   public   String   toString()
   {
      return "Dimension could not be Negative !!!";
   }
}
```

Exceptions en JAVA

7. Ecrire sa propre classe d'exception

On peut ensuite utiliser la classe NegativeDimensionException comme ceci:

```
static  int area(int l , int b) throws NegativeDimensionException{
   if (l<0 || b <0)
      throw new NegativeDimensionException();
   return l * b;
}
static void meth1() throws NegativeDimensionException{
   System.out.println("Area is "+ area(10,-5));
}
```

```
public static void main(String[] args)  {
   try{
      meth1();
   }
   catch(NegativeDimensionException e){
      System.out.println(e);
   }
}
```