Bloc finally: exemple (2)

Exemples d'exécution :

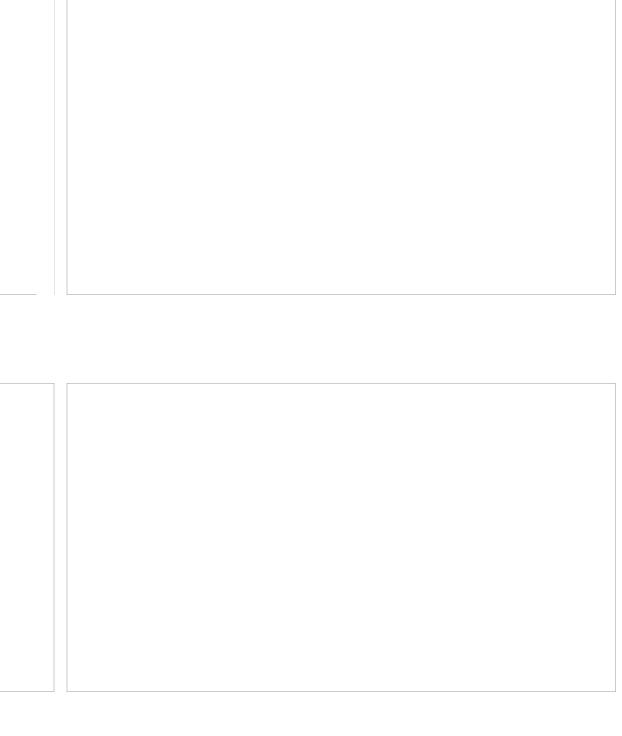
```
>java Inverse 4.1
Il faut un nombre entier!
Passage obligé !

>java Inverse 0
Parti vers l'infini!
Passage obligé !

>java Inverse 4
Inverse * 100 = 25
Passage obligé !

>java Inverse
Passage obligé !

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
```



« Relancement »

Une exception peut être **partiellement traitée** par un bloc catch et *attendre* un *traitement* plus complet *ultérieur* (c'est-à-dire à un niveau supérieur).

Il suffit pour cela de « *relancer* » l'exception au niveau du bloc n'effectuant que le traitement partiel.

(Il faudra bien sûr pour cela que l'appel à ce bloc catch soit lui-même dans un autre bloc try à un niveau supérieur).

« Relancement » : exemple

Déclaration d'une exception

Une méthode lançant une exception sans la traiter localement doit généralement informer qu'elle le fait

Ceci se fait en ajoutant une clause throws à l'entête de la méthode Syntaxe:

Type method(...) throws Exception1, Exception2, ...

Exemple:

```
int inverse(int x) throws Exception
{
  if (x == 0) {
    throw new Exception("Une division par zéro");
  }
  return 1/x;
}
```

Règle « déclarer ou traiter »

Toutes les exceptions en dehors des RunTimeException et des Error doivent :

- ▶ soit être interceptées dans la méthode où elles sont lancées;
- soit être déclarées par la méthode.

Si une exception de ce type est lancée sans être interceptée

- le compilateur émettra un message d'erreur
- « Checked exceptions »

Exceptions personnalisées

Il est possible de programmer ses propres classes d'exception

sous-classe de Exception (ou d'une autre sous-classe d'exception existante)

Contenu minimal:

```
class MonException extends Exception
{
  public MonException() {
    super("mon message par défaut";);
  }
  public MonException(String message) {
    super(message);
  }
}
```

permet de préserver le comportement attendu de getMessage()

Exceptions personnalisées

Il et possible de définir dans une sous-classe d'exception personnalisée tout membre jugé utile :

- code d'erreurs
- informations sur le contexte de détection de l'exception
- etc.

Exemple ...

Exceptions personnalisées : exemple

```
class TropChaudException extends Exception
{
   private double temperatureAnormale;
   private String consigne;

public TropChaudException() { super("Température trop élevée"); }
   public TropChaudException(String message) { super(message); }

public TropChaudException(double uneTemperature, String uneConsigne) {
    super("Température trop élevée");
    temperatureAnormale = uneTemperature;
    consigne = uneConsigne;
   }
   public double getTemperature() {
    return temperatureAnormale;
   }
   public String getConsigne() {
    return consigne;
   }
}
```

Exceptions personnalisées : exemple (2)

Exemple d'exécution du bloc catch :

```
Température trop élevée : 150.0
Consigne -> Vérification de l'appareil de mesure
```

Exemple complet (1)

```
public static void main(String[] args) {
  int nbEssais = 0;
  final int MAX_ESSAIS = 2;
  ArrayList<Double> mesures = new ArrayList<Double>();

do {
   nbEssais++;
   acquerirTemp(mesures);  // remplit le tableau

  try {
    plotTempInverse(mesures);
   }
  // ...
```

Exemple complet (3)

```
private static void plotTempInverse(ArrayList<Double> t)
throws ArithmeticException
{
  for(int i = 0; i < t.size(); i++) {
    try {
      plot(inverse(t.get(i)));
    } catch (ArithmeticException e) {
      System.out.println("Problème à l'indice :" + i);
      // RELANCEMENT
      throw e;
    }
}</pre>
```

Exemple complet (2)

Exemple complet (4)

Conseils/mise en garde

La gestion d'une exception coûte beaucoup plus en temps de calcul qu'un simple if.. then.. else

Si l'erreur peut-être traitée là où elle est découverte, il faut le faire sans passer par les exceptions

Lancer des exceptions spécifiques est plus informatif et utile!