Chapitre 4

Les exceptions

Plan

- Traitement des erreurs
- Gérer les exceptions
- Types d'exceptions
- Lancer des exceptions

Références

- Tutorial de Sun
  - tutorial/essential/exceptions/index.html
- Thinking in Java
  - thinkingInJava/Chapter10.html
- Penser en Java
  - chapitre non traduit actuellement
- Au cœur de Java, Volume 1, chapitre 11

Traitement des erreurs

Détection des erreurs

- Le développement et l'utilisation d'un programme implique la gestion de nombreuses erreurs.
- Du plus souhaitable au moins souhaitable, on peut citer le statut des différentes erreurs :
  1. erreurs détectées à la compilation
  2. erreurs détectées et traitées (corrigées) à l'exécution
  3. erreurs détectées et non traitées à l'exécution
  4. erreurs non détectées à l'exécution
- Le mécanisme d'exceptions utilisé en C++ et java correspond aux points 2 et 3.

Origine des erreurs

- Erreurs d'entrée/sortie
  - faute de saisie de l'utilisateur
  - fichier corrompu ou inaccessible
- Erreurs de matériel
  - imprimante déconnectée
  - disque dur saturé
  - connexion réseau défaillante
- Erreurs de programmation
  - index de tableau invalide
  - emploi d'une variable référence non initialisée

Traitement des erreurs

- En présence d'une erreur, un programme peut se comporter comme suit.
  - idéalement : il permet de revenir à un état défini et de poursuivre ainsi l'exécution.
  - honnêtement : il avertit l'utilisateur, permet une sortie correcte après une sauvegarde de l'ensemble du travail en cours.
  - lamentablement : il s'interrompt brusquement (avec éventuellement un message d'erreur abscons)

Traitement des erreurs en C

- Conventions de programmation
  - valeurs spéciales de retour
  - positionnement de flags
- Mais aucune garantie que les conventions soient suivies par tout le monde.
- Aussi, les programmes C sont-ils peu surs.

Traitement des erreurs en Java

- Gestion d'exceptions
  - si un problème survient, une exception est levée
  - toute exception peut/doit être attrapée
- Le code de gestion des exceptions est placé à part.
- Aussi, les programmes C++ et Java sont-ils plus clairs et plus surs.

Exception

- Une exception est un objet envoyé à la JVM
- Avant de lever (lancer) une exception, celle-ci doit être créée.
  - lancer : throw
  - créer : new
- Une exception peut être attrapée dans tout contexte plus général.
  - attraper : catch

Gérer les exceptions

Gérer une exception

- Certaines méthodes peuvent lancer des exceptions.
- Il est nécessaire lorsqu'on utilise une telle méthode :
  - soit d'attraper les exceptions levées
  - soit de laisser passer les exceptions

Méthode lançant un exception

- Dans l'en-tête d'une telle méthode figure après le mot-clef « throws » la liste des types d'exceptions pouvant être lancées.
- Dans le corps d'une telle méthode figure (éventuellement) une ou plusieurs instructions spéciales « throw » permettant de lancer une exception.

Exemple

- Dans la classe InputStream figure la méthode read dont l'en-tête est comme suit :

  public abstract int read() throws IOException

- Cela signifie que tout objet instance de InputStream (ou d'une sous-classe) exécutant read peut éventuellement lancer une IOException.

Attraper une exception

- Un bloc try { } délimite le code à surveiller
- Un ou plusieurs blocs catch(…) { } suivent:
  - chaque bloc catch(…) permet de récupérer la main pour traiter un type d'erreurs.
  - ils sont considérés dans l'ordre jusqu'à ce que l'exception levée corresponde au type ou à un sous-type de celui indiqué en argument.
- Un bloc finally { } apparaît optionnellement.

Exemple

```
int lireOctet()
{
   int x=0;
   try { x= System.in.read(); }
   catch (IOException e)
   {
        System.out.println("Erreur : " + e);
   }
   return x;
}
```

Laisser passer une exception

- Il est intéressant de laisser passer une exception si le contexte (méthode) courant ne permet pas de traiter (corriger) l'erreur.
- Il faut alors ajouter dans l'en-tête le type de l'exception devant être relancée (laissée passer).
- Si la méthode principale (main) laisse passer l'exception, le programme s'arrête.

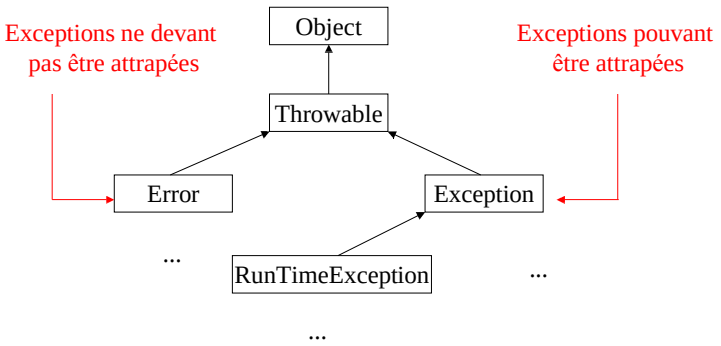
Exemple

```
int lireOctet() throws IOException
{
   int x=System.in.read();
   return x;
}
```


Types d'exceptions

- Une exception est un objet d'une classe dérivée de Throwable.
- La hiérarchie des exceptions se sépare en deux branches importantes :
  - Error
  - Exception

Types d'exceptions



Bloc finally

- Le bloc optionnel finally {} contient un traitement exécuté systématiquement :
  - que le bloc try {} ait levé ou non une exception,
  - que cette exception ait été attrapée ou non.
- Ce bloc peut être utilisé par exemple pour garantir la libération de certaines ressources (comme fermer un fichier).

Classe Error

- Un erreur (objet instance d'une classe dérivée de Error) est levée :
  - si une erreur interne se produit
  - si un manque de ressources apparaît.
- On ne doit pas tenter de les attraper.
- Ce type d'erreurs est assez rare.

Types d'exceptions

Classe Exception

- La hiérarchie Exception se sépare en deux branches :
  - les classes issues de RunTimeException
    - définies dans le SDK
    - les exceptions peuvent être lancées :
      - par la JVM
      - par le code de l'utilisateur
  - les classes issues directement de Exception
    - définies par l'utilisateur
    - Les exceptions sont lancées par le code de l'utilisateur

Classe RunTimeException

- Les classes suivantes en héritent :
  - ArithmeticException
  - NullPointerException
  - ArrayIndexOutOfBoundsException
  - NegativeArraySizeException
  - ClassCastException
  - IllegalArgumentException

Nouvelle classe d'exception

- Intéressant lorsqu'on rencontre un problème qui ne correspond pas de façon adéquate à l'une des classes d'exception standards.
- Il suffit de définir une classe héritant de Exception.
- Il est d'usage de donner un constructeur sans argument et un constructeur qui contient un message détaillé.

Exemple

- Première possibilité

```
class MonException extends Exception { }
```

- Seconde possibilité

```
class MonException extends Exception
{
      public MonException() { }
      public MonException(String msg)
      {
            super(msg);
      }
}
```


Lancer des exceptions

Lancer des exceptions

- Pour sécuriser le code, il faut envisager des situations pour lesquelles des exceptions soient lancées.
- Lancer une exception à partir d'une méthode appelée permet de prévenir la méthode appelante qu'il y a un problème.
- La méthode appelante peut alors réagir à l'erreur (ou laisser passer l'exception à la méthode qui l'a appelée…).

Nature des problèmes

- Problèmes simples
  - problèmes qui peuvent être résolus localement (dans le contexte courant).

  ⇒ non utilisation du mécanisme d'exception
- Problèmes graves (exceptionnels)
  - problèmes qui ne peuvent être traités que dans un contexte plus général.

  ⇒ utilisation du mécanisme d'exception

Exemple de problèmes simples

- Lecture d'une note (comprise entre 0 et 20) :

```
int note= in.readInt();
```

- Si la note est incorrecte ?

```
int note= in.readInt();
while (note < 0 || note > 20)
   note=in.readInt();
```


Exemple de problèmes graves

- Ajouter une note :

```
void ajouterNote(int note)
{
     notes[nbNotes++]=note;
}
```

- Si la note incorrecte ou si plus de place ?
  - on peut créer deux types d'exceptions
    - NoteIncorrecteException
    - TropDeNotesException

Exemple

```
class NoteIncorrecteException extends Exception
{
   public NoteIncorrecteException() { }
   public NoteIncorrecteException(String msg)
   {
        super(msg);
   }
}

// idem avec TropDeNotesException
```


Lancer une exception

- Pour qu'une méthode puisse lancer une exception, il faut que :
  - le type de celle-ci figure dans l'en-tête
  - un objet Exception soit créé
  - une instruction throw apparaisse

Exemple

```
void ajouterNote(int note) throws
   NoteIncorrecteException, TropDeNotesException
{
   if (note < 0 || note > 20)
        throw new NoteIncorrecteException();
   if (nbNotes == nbMaxNotes)
        throw new TropDeNotesException();
   notes[nbNotes++]=note;
}
```


Exercice

- Ecrire les instructions permettant d'ajouter une note (en appelant la méthode ajouterNote) et récupérant les exceptions éventuelles.