Système d'exceptions

Exceptions

- Que se passe t'il si on a une erreur?
- Methode numéro une : le statut / code d'erreur
- Méthode numéro deux : les exceptions

Principe

- On identifie un bout du code susceptible de produire des erreurs
- On définit une réaction en cas de problème (et **en fonction** du problème)
- Eventuellement on prévoit de continuer ou non l'éxécution

Vocabulaire

- Une exception est un objet
- On lance ou on lève (throw) une exception
- On la rattrape

Identification

- void readFromFile(String path) throws FileNotFoundException
- Vous **devez** la traiter ou la passer à l'échelon supérieur

Pile d'appel

- L'exception est levée
- Elle remonte au niveau immédiatement supérieur en interrompant l'éxécution
- Si on la traite, ça s'arrête là
- Si on l'ignore elle remonte d'un cran
- Si elle n'est pas traitée au niveau du main, le programme se termine

Syntaxe

- try
- catch
- finally

Blocs Try/Catch

```
try {
  // code dangereux
} catch (<Type d'exception> <nom>) {
  // réagir en conséquence
} catch (<Autre type d'exception> <nom>) {
  // encore
} (...)
   finally {
  // que fait on quoi qu'il arrive?
}
```

Exemple

```
try {
  readFromFile(path);
} catch(FileNotFoundException fnf) {
  System.out.println("No such file");
} catch(IOException io) {
  System.out.println("Can't read that file");
} catch(Exception e) {
  e.printStackTrace();
} finally {
  System.out.println("Finito");
}
```

Lever une exception

```
Exception e = new FileNotfoundException();
throw e;
```

Créer une exception

- Sous classer la classe Exception
- Implémenter l'interface "Throwable"

Exemple

- public class MyException extends Exception {
  }
- public class MyException implements Throwable {
  }

Avantages pratiques

- Gestion des exceptions
- Passage de messages dans l'arbre d'éxécution