

Programmation Orientée Objet avec JAVA

Plan

- Introduction
- Le langage JAVA
- La programmation objet
- Les bases du langage
- Les classes et les objets
- L'héritage et le polymorphisme
- La gestion des exceptions
- Le graphisme

Chapitre 6 – Gestion des exceptions

Chapitre 6 – Gestion des exceptions

- **Définition**

- ✦ Mécanisme de gestion des incidents survenus lors de l'exécution d'un programme
- ✦ Lever une exception consiste à indiquer qu'un incident « exceptionnel » vient de se produire
- ✦ Capturer une exception consiste à indiquer qu'on va la gérer
- ✦ Il est possible de propager les exceptions aux méthodes « appelantes »
- ✦ Il est obligatoire de traiter l'exception à un des niveaux (sauf si la classe d'exceptions hérite de « RuntimeException »)

Chapitre 6 – Gestion des exceptions

- Intérêts

- ✦ Gérer systematiquement les incidents
- ✦ Traiter les incidents au « bon » moment grâce au mécanisme de propagation
- ✦ Clarifier le code source

- Contraintes

- ✦ Lourdeur de certaines parties du code
- ✦ Traitements ralentis

Chapitre 6 – Gestion des exceptions

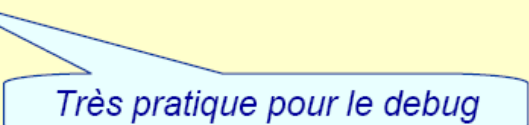
- **Les zones « try - catch »**
 - ✦ Zone « **try** » contient le code susceptible de déclencher un incident
 - ✦ Zone « **catch** » contient le code permettant de gérer l'incident survenu dans la zone « try »

```
public void LitFichier (String nomfic) {  
    try {  
        FileInputStream fic = new FileInputStream ( nomfic );  
    }  
    catch ( FileNotFoundException fnfe ) {  
        System.out.println ("Fichier "+nomfic+" non trouvé !" );  
    }  
}
```

Chapitre 6 – Gestion des exceptions

- **Exemple**

```
public void LitFichier ( String nomfic ) {  
    try {  
        FileInputStream fic = new FileInputStream ( nomfic );  
        fic.read ( ... );  
    }  
    catch ( FileNotFoundException fnfe ) {  
        System.out.println ( "Fichier "+nomfic+" non trouvé !" );  
    }  
    catch ( IOException ioe ) {  
        System.out.println ( "Problème à la lecture du fichier "+nomfic );  
        ioe.printStackTrace ( );  
    }  
}
```



Très pratique pour le debug

Chapitre 6 – Gestion des exceptions

- **Ma zone** « `finally` »

✦ Zone dans laquelle le code est exécuté quoiqu'il arrive

```
public void LitFichier ( String nomfic ) {  
    try {  
        FileInputStream fic = new FileInputStream ( nomfic );  
    }  
    catch ( FileNotFoundException fnfe ) {  
        System.out.println ( "Fichier "+nomfic+" non trouvé !" );  
    }  
    finally {  
        ...  
    }  
}
```


Chapitre 6 – Gestion des exceptions

- **Propagation**

Remarques

- ✦ Une exception levée au niveau N peut être traitée au niveau N+1
- ✦ La méthode dans laquelle se produit l'incident doit être « habilitée » à propager les exceptions (prédéfinies ou non)

Syntaxe

- ✦ Le mot-clé « **throws** »

openFile propage les exceptions mais ne les traite pas

```
public void openFile ( ... ) throws FileNotFoundException, IOException {  
    ...  
}
```

Chapitre 6 – Gestion des exceptions

```
public void LitFichier ( String nomfic ) {  
    try {  
        obj.openFile ( nomfic ) ;  
    }  
    catch ( FileNotFoundException fnfe ) {  
        System.out.println ( "Fichier "+nomfic+" non trouvé !" ) ;  
    }  
    catch ( MonException me ) {  
        System.out.println ( " ... " )  
    }  
}
```

« *openFile* » lève
FileNotFoundException
et *MonException*

Chapitre 6 – Gestion des exceptions

- **Intérêt**

- ✦ Le niveau N+1 connaît généralement mieux le contexte global (traitements plus appropriés)
- ✦ Renvoyer une erreur vers le client si l'incident s'est produit côté serveur
- ✦ Enregistrer les erreurs dans un fichier
- ✦ Augmenter la souplesse du traitement des incidents

Chapitre 6 – Gestion des exceptions

- ✦ Explicitement levée par le programmeur
- ✦ Mot-clé « **throw** »
- ✦ La méthode dans laquelle est levée l'exception doit être « habilitée » à lever ce type d'exception

```
public void openFile ( ... ) throws FileNotFoundException, IOException {  
    FileInputStream fic = new FileInputStream ( nomfic );  
    ...  
    if ( ... ) { // problème  
        throw new IOException ( );  
    }  
    ...  
}
```

Chapitre 6 – Gestion des exceptions

- ✦ La nouvelle classe d'exception doit hériter de classe « Exception » prédéfinie dans Java

```
public class AEgalBException extends Exception {  
    private String Msg ;  
    public AEgalBException ( String ch ) {  
        Msg = ch ;  
    }  
    public String message ( ) {  
        return Msg ;  
    }  
}
```

Chapitre 6 – Gestion des exceptions

```
public class EssaiException {
    static void MaMethode ( int a, int b ) throws AEgalBException {
        if ( a == b )
            throw new AEgalBException ( "A égal B" );
    }

    static public void main ( String argv [] )
        try {
            MaMethode ( 2, 2 );
            System.out.println ( " Pas d'erreur !" );
        }
        catch ( AEgalBException e ) {
            System.out.println ( " Erreur : "+e.message ( ) );
        }
    }
}
```

Chapitre 6 – Gestion des exceptions

- La hiérarchie des classes

