UNIVERSIT MOHAMMED V DE RABAT
FACULT DES SCIENCES DE RABAT
DPARTEMENT INFORMATIQUE

Cours

Programmation Oriente Objet

JAVA

Lahoucine Ballihi

Universit Mohammed V de Rabat
Facult des Sciences de Rabat
Dpartement Informatique
LRIT (Laboratoire de Recherche en Informatique et Tlcommunications)
B.P 1014 Rabat - Maroc
Phone : + 212 (0) 5 37 68 69 44
Fax : + 212 (0) 5 37 68 69 44

E-mail : l.ballihi@um5r.ac.ma

Table des matires

1 Programmation Orienté Objet **5**
1.1 Gestion des Exception 5

2 Réception et Traitement d'une exception **6**

3 Capture et traitement d'exception : try ... catch **9**

4 Créer vos propres classes d'exceptions **11**

1 Programmation Orienté Objet

1.1 Gestion des Exception

Université Mohammed V de Rabat – Faculté des Sciences

Année Universitaire : 2022/2023

LPRT : FI & FTA

Programmation Orientée Objet en Java

Gestion des Exceptions

Coordonnateur : Lahoucine BALLIHI

23/11/2022

Dans cette section je vais décrire les notions suivantes :

1. Réception et Traitement d'une exception
2. Capture et traitement d'exception : try ... catch
3. Créer vos propres classes d'exceptions

2 Réception et Traitement d'une exception

Notion d'erreur / Exception

- **Différents types d'erreurs**
  - ✓ Mauvaise gestion des classes : accès hors tableau, …
  - ✓ Entrées utilisateurs non valides : effacer un fichier inexistant, …
  - ✓ Liées aux périphériques : manque de papier dans l'imprimante, …
  - ✓ Limitation physique : disque plein, …
- **Le traitement des erreurs / les différentes possibilités**
  - ✓ Retourner un code d'erreur
  - ✓ Ne rien faire (absorber l'erreur)
  - ✓ Imprimer des messages d'erreur
  - ✓ Mettre à jour des variables globales d'erreur
  - ✓ Utiliser le mécanisme d'exception unifié et standardisé

--- Java supporte le mécanisme de gestion d'exception ---

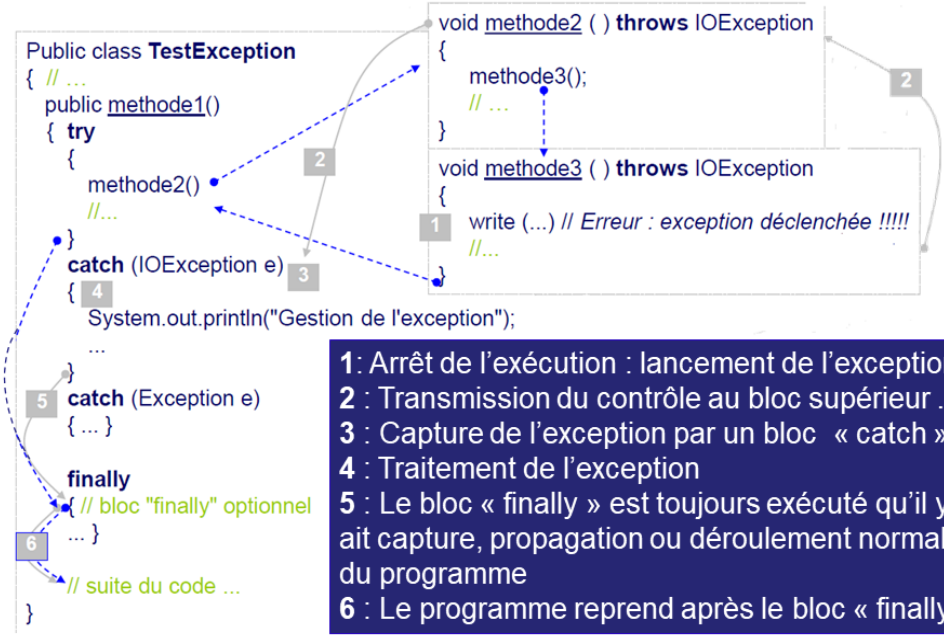
Pr. Lahoucine BALLIHI 2 P.O.O en Java

Traitement des cas d'erreurs exceptionnelles

- **Mécanisme d'exception / gestionnaire d'exception**
  - ✓ Permet de transmettre le problème du traitement de l'erreur dans un **contexte qualifié** (de niveau supérieur) pour gérer l'erreur
  - ✓ Simplifie en allégeant le code de la gestion locale des erreurs par une **recentralisation** des procédures de traitement d'erreur
- **Mise en œuvre des exceptions**
  - ✓ On **déclenche** (lance) une exception par l'instruction **throw**
  - ✓ On **capte** (attrape) une exception dans bloc de type **try**
  - ✓ On **traite** (gère) une exception avec l'instruction **catch**

Pr. Lahoucine BALLIHI 3 P.O.O en Java

Exemple : mécanisme d'exception / finally



1: Arrêt de l'exécution : lancement de l'exception
2 : Transmission du contrôle au bloc supérieur …
3 : Capture de l'exception par un bloc « catch »
4 : Traitement de l'exception
5 : Le bloc « finally » est toujours exécuté qu'il y ait capture, propagation ou déroulement normal du programme
6 : Le programme reprend après le bloc « finally »

Pr. Lahoucine BALLIHI 4 P.O.O en Java

Réception / Traitement d'une exception

➢ **Lorsque qu'intervient une exception (réception par la JVM)**

- ✓ L'exécution normale du programme est arrêté
- ✓ Recherche du bloc de traitement de l'exception (catch)
  - en local,
  - puis on remonte la liste des appelants
- ✓ Traitement de l'exception
- ✓ Tous les blocs "finally" rencontrés sont exécutés
- ✓ Reprise du code

```
i == 1
i == 2
erreur i est égal a 0
reprise du code
```

```
public class TestException{

    public void test(int i) throws Exception{
        if (i == 0)
        throw new Exception("erreur i est égal a 0");
        System.out.println("i == " + i);
    }

    public static void main(String[] args){
    TestException t = new TestException();
    try{
        t.test(1);
        t.test(2);
        t.test(0);
        t.test(3);
    } catch (Exception e){
        System.out.println(e.getMessage());
    }
    System.out.println("reprise du code");
    }
}
```

Pr. Lahoucine BALLIHI 5 P.O.O en Java

Lancement d'exceptions / throw

- **Une exception est lancée par la commande throw e;**
  - **e : est un objet qui DOIT dériver de la classe Throwable**
    throw new IllegalArgumentException("Pb en lecture");
- **Spécification obligatoire des exceptions susceptibles d'être lancées**
  - ✓ Toute fonction susceptible d'émettre des exceptions "explicites" doit le mentionner (vérification à la compilation) :
    - **Exceptions levées dans la méthode et non attrapées par celle-ci**
    - **Exceptions levées dans des méthodes appelées par la méthode**
    - **Exceptions levées et traitées par la méthode puis propagées**

```
public void read(DataInputStream in) throws IOException
{     ... double s = in.readDouble ( );          // peut générer une exception
      ... }
```

- **Une fonction sans clause throws**
  - garantit qu'elle ne va pas générer d'exception explicite

Pr. Lahoucine BALLIHI 6 P.O.O en Java

3 Capture et traitement d'exception : try ... catch

Capture et traitement d'exception : try ... catch

- **Capture (bloc try) et traitement d'exceptions (clause catch)**

```
try
{
        // code susceptible de déclencher une exception
}
catch (Type1 id1) {     // capture des exception de type Type1
                        // ou type dérivé de Type1
        // traitement de l'exception de Type1
}
catch (Type2 id2) {     //capture des exceptions de type Type2

        // traitement de l'exception de Type2
}

// etc.
```

- **Remarques**
  - Les blocs **try** encapsulent de nombreux appels de fonctions ...
  - Le transfert de contrôle est donné à la 1ère clause **catch** de bon type
  - Il possible d'exploiter la notion d'héritage pour hiérarchiser les exceptions
    ➔ l'ordre des blocs catch doit respecter l'ordre d'héritage

Pr. Lahoucine BALLIHI 7 P.O.O en Java

Retransmission d'une exception

- **Il est possible de retransmettre une exception**

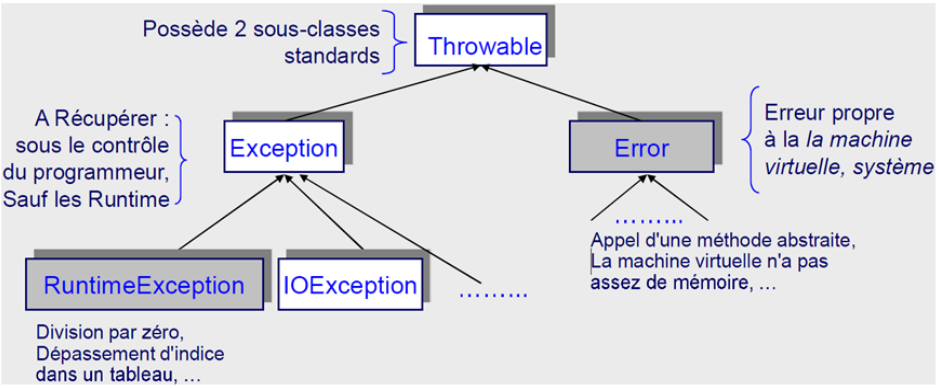
```
try
{
                // code
}

catch (Throwable t)             // capture tout type d'exception
{
                ...
                throw t;        // retransmet l'erreur courante

}
```

- **Si une exception est propagée sans être rattrapée :**
  - ✓ Si propagation jusqu'à la méthode "main"
    public static void main (String[] args) throws Exception
    - Affichage d'un message d'erreur et de la pile des appels,
    - Arrêt de l'exécution du programme.

Pr. Lahoucine BALLIHI 8 P.O.O en Java

Exception standard en Java

➢ **Java possède une hiérarchie standard des exceptions**



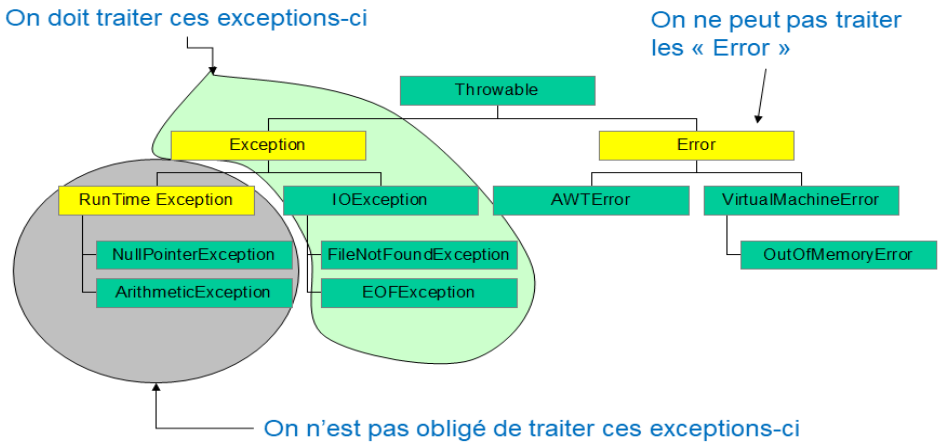
- ✓ Exception techniques et d'erreurs (Runtime, Error) : un programme n'est pas obligé de lever ces 2 types d'exception (raisons essentiellement pratiques)
- ✓ Exception explicite ou applicatives : pour toutes les autres exceptions le programme doit les lever (imposé par le compilateur)
- ✓ Pour définir de nouveaux types d'exception on hérite en général de java.lang.Exception

Pr. Lahoucine BALLIHI 9 P.O.O en Java

4 Créer vos propres classes d'exceptions

Exception standard en Java

➢ **Hiérarchie des exceptions**



Pr. Lahoucine BALLIHI 10 P.O.O en Java

Créer vos propres classes d'exceptions

➢ **Java.lang.Throwable**

Throwable
String message
Throwable() Throwable(string s) String getMessage() Void printStackTrace() Void printStackTrace(PrintStream) …

- Message d'erreur décrivant l'exception
- Constructeur avec et sans message d'erreur
- Retourne le message d'erreur
- Imprime sur la sortie standard ou sur un stream, l'exception et la trace de l'exception dans la pile

➢ **Exemple de récupération du message de l'exception**

```
try
{
        …
}
catch (Exception e)
{
        System.out.println(e.getMessage());          // ou
        System.out.println("Exception " + e);        // cf. notion de toString
}
```

Pr. Lahoucine BALLIHI 11 P.O.O en Java

Créer vos propres classes d'exceptions

➢ **Un petit exemple Java**

```
public class MonException extends Exception
{
        public MonException() {
        }

        public MonException(String msg) {
                super(msg);
        }
}
```

```
public class UneClasse {

        public void g() throws MonException {
                // ...
                throw new MonException("Origine:
                fonction g()");
        }

        public void h() throws MonException {
                // ...
                g();
        }

        public static void main(String[] args) {
                UneClasse c1 = new UneClasse();
                try {
                    c1.h();
                } catch (MonException e) {
                    e.printStackTrace();
                }
        }
}
```

```
MonException: Origine: fonction g()
        at UneClasse.g(UneClasse.java:6)
        at UneClasse.h(UneClasse.java:11)
        at UneClasse.main(UneClasse.java:17)
```

Pr. Lahoucine BALLIHI 12 P.O.O en Java

Exemple complet

```
class MonException1 extends Exception {
        public MonException1(String msg) {
        super(msg);
        System.out.println("cons MonException1" );
        }
}
class MonException2 extends MonException1 {
        public MonException2(String msg) {
        super(msg);
        System.out.println("cons MonException2" );
        }
}
public class TestException {
        public void methodeA(int p) throws MonException1 {
        if (p==1) throw new MonException1("p==1: methodeA");
        if (p==2) throw new MonException2("p==2: methodeA");
        System.out.println("fin de methodeA");
        }
        public void methodeB() throws MonException1 {
        // ...
        try {
        methodeA(2);
        } catch (MonException1 e) {
        System.out.println("Traitement partiel
        dans methodeB : " + e);
        throw e;
        } finally {
        System.out.println("finally de methodeB");
        }
        System.out.println("fin de methodeB");
        }
}
```

```
public class MainException {
   public static void main(String[] args) throws MonException1{
        TestException c1 = new TestException();
        try {
            c1.methodeB();
        } catch (MonException2 e) {
            System.out.println("Traitement dans main() : " + e);
        } finally {
            System.out.println("finally de main()");
        }
        c1.methodeA(1);
        System.out.println("fin de main()");
   }
}
```

```
cons MonException1
cons MonException2
Traitement partiel dans methodeB : MonException2: p==2:
                                                methodeA
finally de methodeB
Traitement dans main() : MonException2: p==2: methodeA
finally de main()
cons MonException1
Exception in thread "main" MonException1: p==1: methodeA
        at TestException.methodeA(TestException.java:6)
        at MainException.main(MainException.java:13)
```

Pr. Lahoucine BALLIHI 13 P.O.O en Java