Programmation Orientée Objet

5 – Exceptions et Java

P. Berthomé

INSA Centre Val de Loire
Département STI — 3 ème année

5 décembre 2017

Plan

Exceptions

- Définition
- Mise en place dans Java

Principes

Gestion d'erreur

- Erreur de conception

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes
  - ...

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes
  - ...

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes
  - ...

Exception

- L'erreur n'est pas traitable dans le code direct

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes
  - ...

Exception

- L'erreur n'est pas traitable dans le code direct
- On laisse la responsabilité du traitement à l'appelant

Principes

Gestion d'erreur

- Erreur de conception
- Erreur de codage
- Utilisation d'une méthode hors de ses spécifications
  - division par zéro
  - chercher un élément hors bornes
  - ...

Exception

- L'erreur n'est pas traitable dans le code direct
- On laisse la responsabilité du traitement à l'appelant
- Mécanisme particulier arrêtant le flot normal des instructions

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Récupération

- Si la fonction est prête pour recevoir l'exception

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Récupération

- Si la fonction est prête pour recevoir l'exception
- Elle gère en fonction de l'exception reçue

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Récupération

- Si la fonction est prête pour recevoir l'exception
- Elle gère en fonction de l'exception reçue
- Sinon, elle l'envoie à la fonction appelante

Mécanisme

Envoi d'une exception

- Lorsqu'une méthode/fonction détecte une utilisation *frauduleuse*
- Elle s'arrête
- Elle envoie un message à la fonction appelante

Récupération

- Si la fonction est prête pour recevoir l'exception
- Elle gère en fonction de l'exception reçue
- Sinon, elle l'envoie à la fonction appelante
- Jusqu'à éventuellement la machine virtuelle

En Java

Attraper

- Savoir gérer les problèmes en local

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***
- On peut attraper différentes exceptions

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***
- On peut attraper différentes exceptions
- Avec des traitements spécifiques

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***
- On peut attraper différentes exceptions
- Avec des traitements spécifiques

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***
- On peut attraper différentes exceptions
- Avec des traitements spécifiques

Spécifier

- Dire que la méthode peut retourner des exceptions

En Java

Attraper

- Savoir gérer les problèmes en local
- ***catch***
- On peut attraper différentes exceptions
- Avec des traitements spécifiques

Spécifier

- Dire que la méthode peut retourner des exceptions
- ***throws***

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Error

- Dues à problèmes externes au programme

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Error

- Dues à problèmes externes au programme
- Par exemple, un problème hardware

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Error

- Dues à problèmes externes au programme
- Par exemple, un problème hardware

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

Error

- Dues à problèmes externes au programme
- Par exemple, un problème hardware

Runtime Exception

- Utilisation abusive des API

Types d'exceptions

Checked Exception

- Les situations exceptionnelles doivent être prévues
- Par exemple, le nom d'un fichier
- **Doivent** être gérées

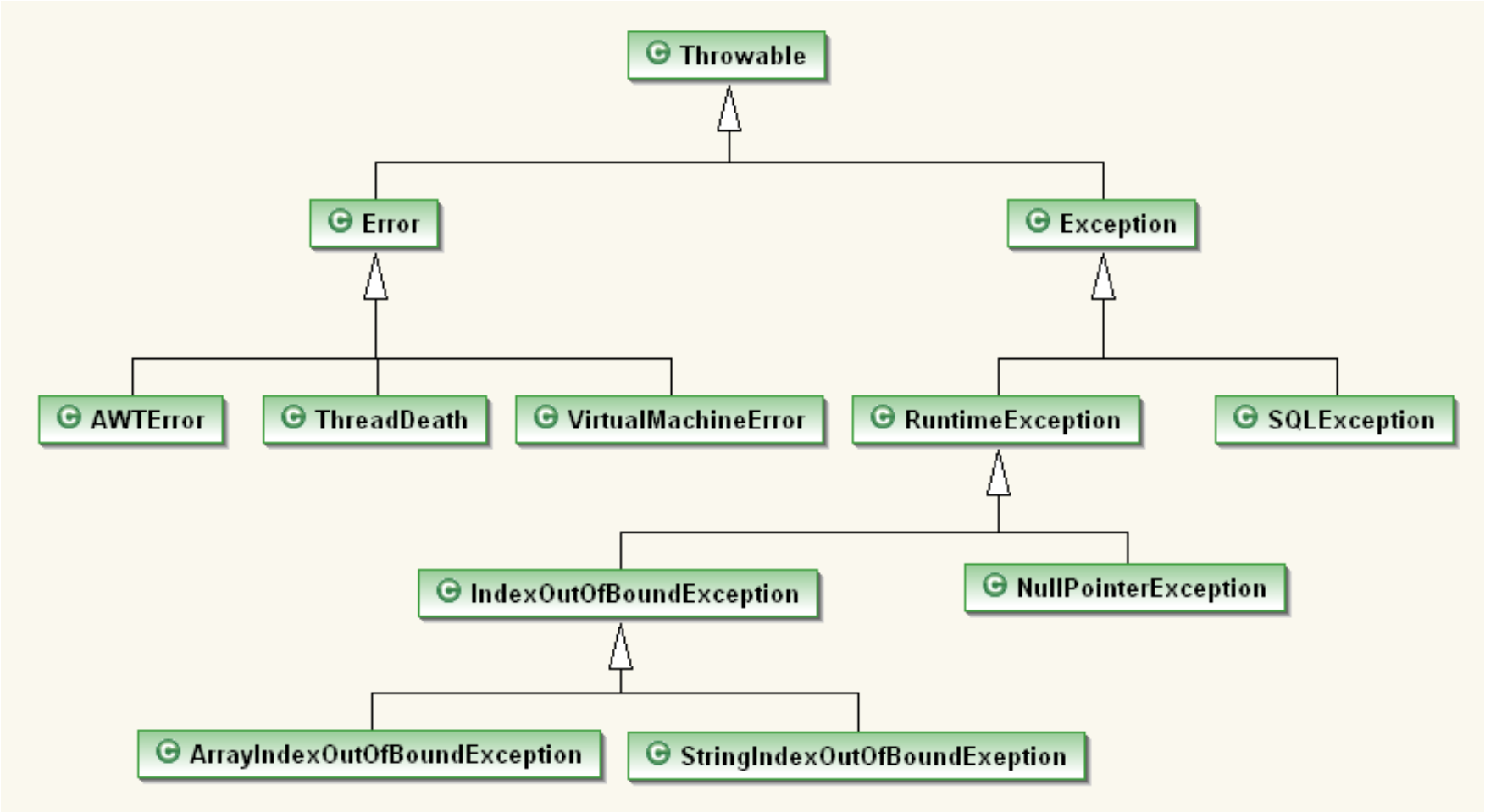
Error

- Dues à problèmes externes au programme
- Par exemple, un problème hardware

Runtime Exception

- Utilisation abusive des API
- Par exemple, écriture en dehors des bornes, pointeur ***null***

Hiérarchie (très partielle) des exceptions



try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

Bloc *catch*

- Récupération des exceptions

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

Bloc *catch*

- Récupération des exceptions
- Un traitement par exception

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

Bloc *catch*

- Récupération des exceptions
- Un traitement par exception
- Traitement des exceptions des plus spécialisées aux plus générales

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

Bloc *catch*

- Récupération des exceptions
- Un traitement par exception
- Traitement des exceptions des plus spécialisées aux plus générales

try et *catch*

Bloc *try*

- Code critique pouvant générer des exceptions
- Les variables déclarées dans ce bloc sont locales

Bloc *catch*

- Récupération des exceptions
- Un traitement par exception
- Traitement des exceptions des plus spécialisées aux plus générales

Bloc *finally*

- Ce qui est fait de toute manière

Exemple

```
PrintWriter out = null;
```

Exemple

```
PrintWriter out = null;
try {
```

Exemple

```
PrintWriter out = null;
try {
      System.out.println("Entering try statement");
      out = new PrintWriter(new FileWriter("OutFile.txt"));
            for (int i = 0; i < SIZE; i++)
                  out.println("Value at: " + i + " = " + vector.elementAt(i));
}
```

Exemple

```
PrintWriter out = null;
try {
      System.out.println("Entering try statement");
      out = new PrintWriter(new FileWriter("OutFile.txt"));
            for (int i = 0; i < SIZE; i++)
                  out.println("Value at: " + i + " = " + vector.elementAt(i));
}  catch (ArrayIndexOutOfBoundsException e) {
       System.err.println("Caught "
             + "ArrayIndexOutOfBoundsException: " + e.getMessage());
}
```

Exemple

```
PrintWriter out = null;
try {
      System.out.println("Entering try statement");
      out = new PrintWriter(new FileWriter("OutFile.txt"));
            for (int i = 0; i < SIZE; i++)
                  out.println("Value at: " + i + " = " + vector.elementAt(i));
}  catch (ArrayIndexOutOfBoundsException e) {
       System.err.println("Caught "
             + "ArrayIndexOutOfBoundsException: " + e.getMessage());
}  catch (IOException e) {
       System.err.println("Caught IOException: " + e.getMessage());
}
```

Exemple

```
PrintWriter out = null;
try {
      System.out.println("Entering try statement");
      out = new PrintWriter(new FileWriter("OutFile.txt"));
            for (int i = 0; i < SIZE; i++)
                  out.println("Value at: " + i + " = " + vector.elementAt(i));
}  catch (ArrayIndexOutOfBoundsException e) {
      System.err.println("Caught "
            + "ArrayIndexOutOfBoundsException: " + e.getMessage());
}  catch (IOException e) {
      System.err.println("Caught IOException: " + e.getMessage());
}  finally {
      if (out != null)
            {System.out.println("Closing PrintWriter"); out.close();}
      else System.out.println("PrintWriter not open");
}
```

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

throw

- Permet d'envoyer un objet exception

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

throw

- Permet d'envoyer un objet exception

```
Integer getElementFromVector(Vector<Integer> v, int i)
        throws ArrayIndexOutOfBoundsException, SQLException{
```

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

throw

- Permet d'envoyer un objet exception

```
Integer getElementFromVector(Vector<Integer> v, int i)
        throws ArrayIndexOutOfBoundsException, SQLException{
if (i<0 || i>=v.size())
    throw new ArrayIndexOutOfBoundsException("Hors bornes");
```

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

throw

- Permet d'envoyer un objet exception

```
Integer getElementFromVector(Vector<Integer> v, int i)
        throws ArrayIndexOutOfBoundsException, SQLException{
if (i<0 || i>=v.size())
    throw new ArrayIndexOutOfBoundsException("Hors bornes");
if (i==0) throw new SQLException("Pas SQL");
```

Comment (r)envoyer une exception

throws

- Si l'exception n'est pas gérée par le bloc ***try/catch***
- et l'exception doit être gérée
- La méthode doit spécifier qu'elle renvoie une exception

throw

- Permet d'envoyer un objet exception

```
Integer getElementFromVector(Vector<Integer> v, int i)
        throws ArrayIndexOutOfBoundsException, SQLException{
if (i<0 || i>=v.size())
    throw new ArrayIndexOutOfBoundsException("Hors bornes");
if (i==0) throw new SQLException("Pas SQL");
return v.elementAt(i);
```

Structure d'une exception

Méthodes principales

- Constructeur.

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception
  - Un message peut être mis comme paramètre
    throw new *Exception(*"mon message d'erreur"*)*

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception
  - Un message peut être mis comme paramètre
    throw new *Exception(*"mon message d'erreur"*)*
- *getMessage*

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception
  - Un message peut être mis comme paramètre
    throw new *Exception(*"mon message d'erreur"*)*
- *getMessage*
- *printStackTrace*

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception
  - Un message peut être mis comme paramètre
    throw new *Exception(*"mon message d'erreur"*)*
- *getMessage*
- *printStackTrace*

Structure d'une exception

Méthodes principales

- Constructeur.
  - Utilisé lors de l'envoi de l'exception
  - Un message peut être mis comme paramètre
    throw new *Exception(*"mon message d'erreur"*)*
- *getMessage*
- *printStackTrace*

```
catch (Exception cause) {
StackTraceElement elements[] = cause.getStackTrace();
for (int i = 0, n = elements.length; i < n; i++) {
    System.err.println(elements[i].getFileName() + ":"
                       + elements[i].getLineNumber() + ">> "
                       + elements[i].getMethodName() + "()"); }
```

Créer ses propres exceptions

Pourquoi

- Les exceptions standard ne correspondent pas

Créer ses propres exceptions

Pourquoi

- Les exceptions standard ne correspondent pas
- Des renseignements importants doivent être consignés

Créer ses propres exceptions

Pourquoi

- Les exceptions standard ne correspondent pas
- Des renseignements importants doivent être consignés

Créer ses propres exceptions

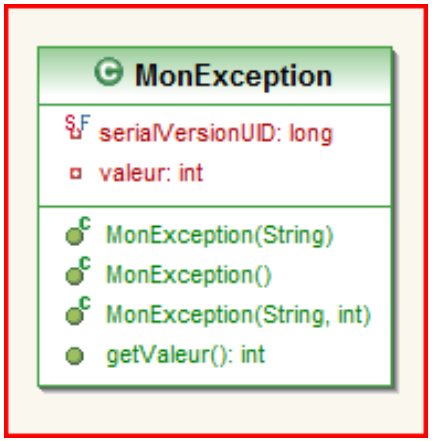
Pourquoi

- Les exceptions standard ne correspondent pas
- Des renseignements importants doivent être consignés

Comment

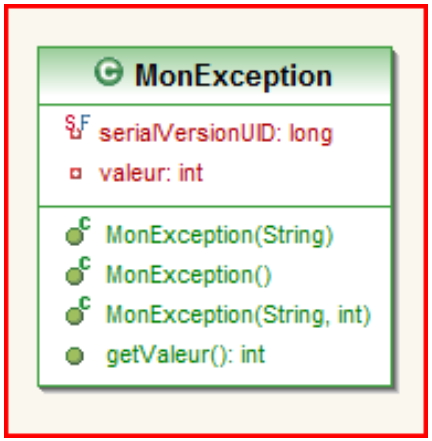
- Hériter de la classe d'Exception la plus proche

Exemple


MonException
serialVersionUID: long
valeur: int
MonException(String)
MonException()
MonException(String, int)
getValeur(): int

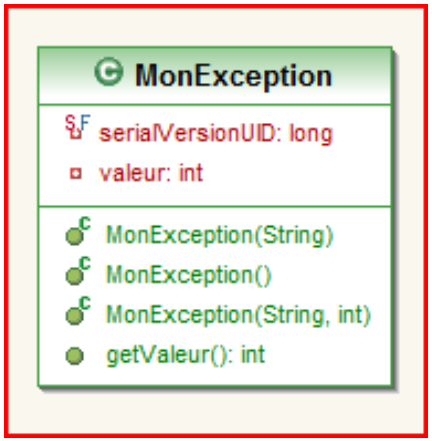
Exemple

```
public class MonException extends
        IndexOutOfBoundsException {
private int valeur;
```

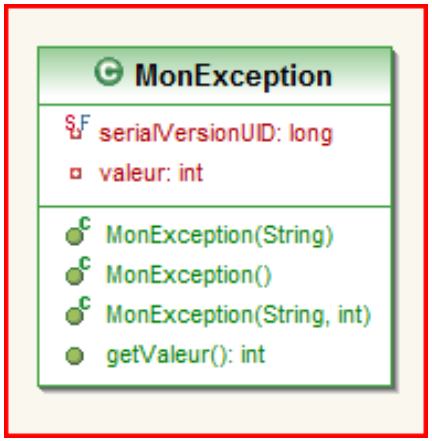


Exemple

```
public class MonException extends
        IndexOutOfBoundsException {
private int valeur;
public MonException() {
   setValeur(0);}
```

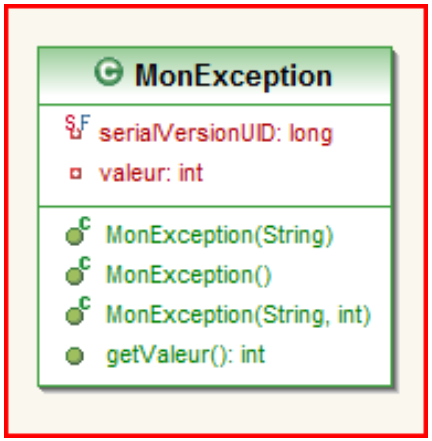


Exemple



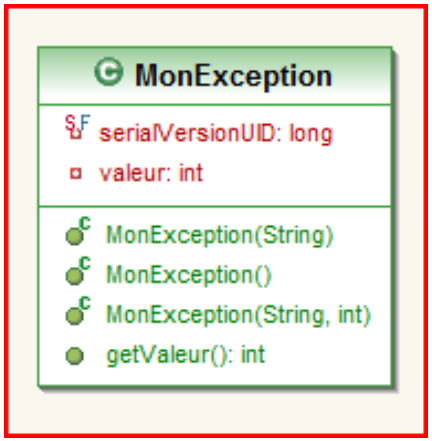

```
public class MonException extends
      IndexOutOfBoundsException {
private int valeur;
public MonException() {
  setValeur(0);}
public MonException(String arg0) {
  super(arg0);
  setValeur(0);}
```

Exemple




```
public class MonException extends
      IndexOutOfBoundsException {
private int valeur;
public MonException() {
  setValeur(0);}
public MonException(String arg0) {
  super(arg0);
  setValeur(0);}
public MonException(String message, int val){
  super(message);
  setValeur(val);}
```

Exemple




```
public class MonException extends
      IndexOutOfBoundsException {
private int valeur;
public MonException() {
   setValeur(0);}
public MonException(String arg0) {
   super(arg0);
   setValeur(0);}
public MonException(String message, int val){
   super(message);
   setValeur(val);}
private void setValeur(int valeur) {
   this.valeur = valeur;}
public int getValeur() {
   return valeur;}
}
```