

# **Cours n° 5**

# **Les exceptions**



## Sommaire

---

- 1. Notion d'exception**
- 2. Bibliothèque des classes d'exception**
- 3. Création et déclenchement**
- 4. Traitement d'une exception**
- 5. Propagation d'une exception**

## Introduction

### Gestion des événements exceptionnels

Détection et gestion des problèmes pendant l'exécution

- Données ou résultats incorrects,

- Débordement mémoire,

- Opérations interdites

Détection par la JVM ou par le programme

### Simplification de la programmation

Traitement local des problèmes

- Correction puis reprise de l'exécution normale

Traitement déporté des problèmes

- Correction puis reprise de l'exécution à une étape précédente

- Centralisation des traitements et génération de compte rendu

## Concept

**Gérer certaines anomalies de programme, provoquant un arrêt du programme, de manière à reprendre un cours normal d'exécution**

**Typologie de quelques anomalies provoquant un arrêt de programme** (immédiat/différé)

- Opération interdite (division par zéro, racine carrée d'un entier négatif, ...)
- Calcul conduisant à un overflow/underflow
- Lecture d'une donnée de format invalide (lors d'une interaction –utilisateur, –fichier, ... )
- Mauvaise allocation mémoire (utilisation du `new`)
- Mauvaise utilisation de fonctions/méthodes (violation de précondition/postcondition)

## Mécanisme des exceptions – Traitement des exceptions

### Surveillance d'un bloc d'instructions

**try** { bloc d'instructions dont l'une au moins peut déclencher une exception }

Marque et surveille un bloc d'instructions qui potentiellement peut déclencher une exception : -soit explicitement lancée par l'exécution d'une instruction `throw`,  
-soit lancée par la JVM

Lorsqu'une exception est levée : le flux d'exécution est interrompu et transféré au bloc `catch` spécialisé sur le traitement du type de l'exception

### Interception de l'exception

**catch** (<typeException> e) { bloc d'instructions gérant l'exception }

**Intercepte l'exception e**, si l'exception déclenchée dans le bloc `try` est du type désigné dans le `catch`, le bloc `try` est exécuté. Plusieurs blocs `catch` peuvent se suivre.

**finally** { bloc optionnel de traitement terminal pour tout type d'exception }

**Optionnel**, ses instructions seront automatiquement exécutées quelque soit le type de l'exception attrapée

## Mécanisme des exceptions – Propagation des exceptions

---

### Envoi d'une exception

**throw** : if (constatation d'anomalie) throw e;

Déclenchement d'une exception (utilisateur) e objet de type Exception

(objet contenant toutes les informations que l'on souhaite donner sur l'erreur)

L'instruction throw est généralement délocalisée (dans une méthode/fonction) relativement aux instructions (try catch finally)

### Délocalisation du traitement d'une exception

#### throws

Dans un prototype de méthode/fonction, signale que la méthode/fonction peut déclencher une exception qui n'est pas traitée dans la méthode (ou partiellement traitée puis relancée)

L'appelant est supposé surveiller (dans un bloc try) l'éventuel déclenchement d'exception lié à l'exécution de la méthode et attraper/gérer les exceptions de ce type.

**Règle : Toute exception envoyée et non interceptée provoque l'arrêt du programme**

---

## Cadre d'utilisation des exceptions

---

### Lecture robuste

**Objectif** : lecture d'une donnée de format valide dans le cadre d'une interaction utilisateur/fichiers...

**Principe** : Boucler sur la requête de lecture tant que la donnée lue est de format invalide

Le mécanisme des exceptions permet :

de détecter l'anomalie de format (levée d'exception dans le bloc try),

de rediriger le flux d'exécution (dans le bloc catch traitant l'exception levée)

### Sécurisation de composants

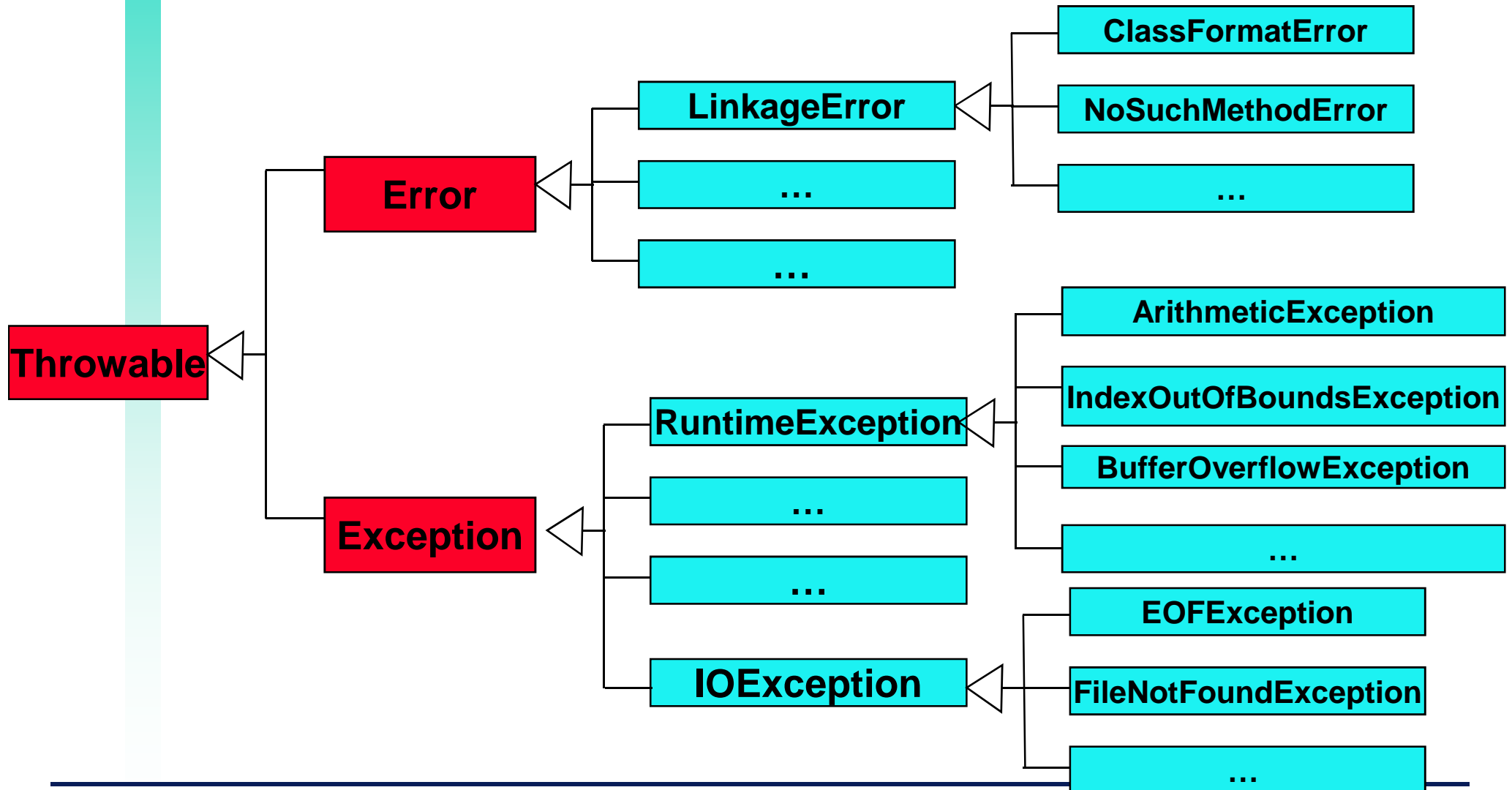
**Objectif** : gérer les anomalies l'utilisation des méthodes du composant

**Principe** : L'anomalie d'utilisation (précondition violée) est détectée dans la méthode, une exception est alors déclenchée/lancée (throw). Généralement la méthode n'attrape/traite pas l'exception, le bloc catch se trouve dans l'appelant qui gère la reprise d'exécution

## 2.

## BIBLIOTHEQUE DES CLASSES D'EXCEPTION

## Arborescence des types d'exception de la JVM





## Arborescence des types d'exception de la JVM

### 53 classes dérivées de la classe Exception

AclNotFoundException, ActivationException, AlreadyBoundException, ApplicationException, AWTException, BackingStoreException, BadLocationException, CertificateException, ClassNotFoundException, CloneNotSupportedException, **DataFormatException**, DestroyFailedException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSEException, IllegalAccessException, InstantiationException, InterruptedException, IntrospectionException, InvalidMidiDataException, InvalidPreferencesFormatException, InvocationTargetException, **IOException**, LastOwnerException, **LineUnavailableException**, MidiUnavailableException, MimeTypeParseException, NamingException, NoninvertibleTransformException, NoSuchFieldException, NoSuchMethodException, NotBoundException, NotOwnerException, **ParseException**, ParserConfigurationException, **PrinterException**, PrintException, PrivilegedActionException, PropertyVetoException, RefreshFailedException, RemarshalException, **RuntimeException**, **SAXException**, ServerNotActiveException, SQLException, TooManyListenersException, TransformerException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, UnsupportedLookAndFeelException, URISyntaxException, UserException, XAException

## Arborescence des types d'exception de la JVM

---

### 21 classes dérivées de la classe IOException

ChangedCharSetException, CharacterCodingException, CharConversionException, **ClosedChannelException**, **EOFException**, FileLockInterruptedException, **FileNotFoundException**, **IOException**, InterruptedIOException, **MalformedURLException**, **ObjectStreamException**, **ProtocolException**, **RemoteException**, **SocketException**, SSLException, SyncFailedException, UnknownHostException, UnknownServiceException, UnsupportedEncodingException, UTFDataFormatException, ZipException

**ClosedChannelException** : entrée/sortie sur un flot fermé

**EOFException** : fin de fichier

**FileNotFoundException** : fichier inconnu

**IOException** : problème au cours d'une écriture ou d'une lecture

**MalformedURLException** : adresse URL (internet, ...) mal construit

**ObjectStreamException** : exceptions sur les flots binaires

**ProtocolException**, **RemoteException**, **SocketException** : exceptions réseaux

## Arborescence des types d'exception de la JVM

### 29 classes dérivées de la classe RuntimeException

**ArithmeticException**, **ArrayStoreException**, **BufferOverflowException**, **BufferUnderflowException**, **CannotRedoException**, **CannotUndoException**, **ClassCastException**, **CMMException**, **ConcurrentModificationException**, **DOMException**, **EmptyStackException**, **IllegalArgumentException**, **IllegalMonitorStateException**, **IllegalPathStateException**, **IllegalStateException**, **ImagingOpException**, **IndexOutOfBoundsException**, **MissingResourceException**, **NegativeArraySizeException**, **NoSuchElementException**, **NullPointerException**, **ProfileDataException**, **ProviderException**, **RasterFormatException**, **SecurityException**, **SystemException**, **UndeclaredThrowableException**, **UnmodifiableSetException**, **UnsupportedOperationException**

**ArithmeticException** : division par zéro, racine carrée d'un nombre négatif...

**BufferOverflowException** : écriture dans un tampon plein

**BufferUnderflowException** : lecture dans un tampon vide

**ClassCastException** : conversion non valide d'un type vers un autre type

**IndexOutOfBoundsException** : Débordement d'un indice d'une structure de données séquentielle (tableau, string, ...)

## Contenu d'une exception

### Classe Exception

Classe dérivée de la classe **Throwable**

- Un constructeur avec comme paramètre (String) la description du problème
- Deux méthodes

**getMessage()** permettant d'obtenir cette description,

**printStackTrace()** donnant la localisation dans le programme du déclenchement de l'exception

### Instance de la classe Exception

Traitement unique pour différents types de problème

### Création de classes dérivées d'Exception

Traitement adapté à chaque type de problème

### **throw IdentException;**

Envoi de l'exception **IdentException** au gestionnaire d'exception de la JVM

### **throw new TypeException(..)**

Création et lancement d'une exception de type **TypeException**

### **Déclenchement d'exception par la JVM**      Détection d'une anomalie

## Principes

### Quatre étapes de traitement

A la levée d'exception, arrêt de l'exécution du code

Recherche du type d'exception employé et de son bloc de traitement

Exécution du bloc de traitement

Reprise ou arrêt de l'exécution normale

### Trois Structures spécialisées et consécutives

**try** {bloc d'instructions où l'on l'on surveille –les déclenchement d'exceptions explicitement lancés par throw ou –les déclenchements d'exception lancés par la JVM }

**catch**(TypeException IdentException) {bloc de traitement associé à l'exception de type TypeException)}

**finally** {bloc optionnel de traitement pour tous les types d'exception}

## 4. TRAITEMENT D'UNE EXCEPTION – EXEMPLE

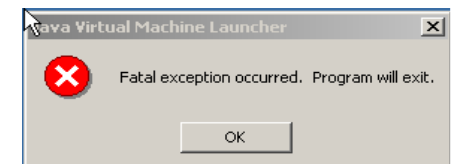
## Exception déclenchée par la JVM (1/5)

```
public class Tableau {
    protected String[] tab_;
    /** Création d'une nouvelle instance de la classe Tableau
     * @param n taille du tableau */
    public Tableau(int n) {tab_ = new String[n];}
    /** lecture d'un élément du tableau
     * @param i indice du tableau
     * @return élément */
    public String get(int i) {return(tab_[i]);}
    /** écriture d'un élément dans le tableau
     * @param i indice du tableau
     * @param mot élément à écrire */
    public void put(int i, String mot) {tab_[i] = mot;}
}
```

## 4. TRAITEMENT D'UNE EXCEPTION – EXEMPLE

## Exception déclenchée par la JVM (2/5)

```
public class testTableau {  
    public static void main(String[] args) {  
        Tableau phrase = new Tableau(2);  
        String mot;  
        phrase.put(0, "le");  
        mot = phrase.get(0); System.out.println(mot);  
        phrase.put(1, "livre");  
        mot = phrase.get(1); System.out.println(mot);  
        phrase.put(2, "de");  
        mot = phrase.get(2); System.out.println(mot);  
    }  
}
```

**Erreur et arrêt de l'exécution**

```
le  
livre  
java.lang.ArrayIndexOutOfBoundsException: 2  
at sem16.Tableau.put(Tableau.java:29)  
at sem16.Exo1.main(Exo1.java:22)
```

## 4.

## TRAITEMENT D'UNE EXCEPTION – EXEMPLE

**Analyse du programme**

```
Tableau phrase = new Tableau(2);
```

phrase est un objet de type Tableau instancié avec le paramètre 2

Son attribut `tab_` est un tableau de dimension 2 (indexé de 0 à 1)

**Dans le main :**

```
phrase.put(2, "de");
```

conduit à l'exécution de la méthode `put` de Tableau :

```
tab_[2] = mot;
```

qui provoque une exception de type **ArrayIndexOutOfBoundsException** déclenchée par la **JVM**

L'exception (`e`) n'est « attrapée puis gérée » à aucun niveau (jusqu'au main) :

le **programme affiche** automatiquement la **trace** de la pile d'exécution

(`e.printStackTrace()`) à partir du déclenchement de l'exception

puis **s'arrête**.



## Exception déclenchée par la JVM (3/5)

```
public class test2Tableau {  
    public static void main(String[] args) {  
        Tableau phrase = new Tableau(2);  
        String mot;  
        try {  
            phrase.put(0, "le"); mot = phrase.get(0); System.out.println(mot);  
            phrase.put(1, "livre"); mot = phrase.get(1); System.out.println(mot);  
            phrase.put(2, "de"); mot = phrase.get(2); System.out.println(mot);  
        }  
        catch (IndexOutOfBoundsException e) {  
            System.out.println("Reprise du contrôle d'exécution");  
            e.printStackTrace();  
            System.out.println("Sortie de programme"); System.exit(0);  
        }  
    }  
}
```

```
le  
livre  
Reprise du contrôle d'exécution  
java.lang.ArrayIndexOutOfBoundsException: 2  
at sem16.Tableau.put(Tableau.java:29)  
at sem16.Exola.main(Exola.java:33)  
Sortie de programme
```



#### 4. TRAITEMENT D'UNE EXCEPTION – EXEMPLE

### Objectifs du programme `TableauDyn.java`

---

#### Sécuriser le composant `Tableau`

**Contrainte** : le composant `Tableau` existe, il n'est pas remis en question, on souhaite juste sécuriser son utilisation

**Première sécurisation** : étendre le tableau à un tableau dynamique en cas d'écriture d'un élément à une position externe au tableau originel

#### Solution retenue :

`TableauDyn` spécialisera `Tableau` en un tableau dynamique (Héritage)

Seule la méthode `puts` sera définie, elle devra **attraper et traiter l'exception native** de la JVM (**`indexOutOfBoundsException`**). Le traitement consistera à étendre la dimension du tableau originel `t` à un tableau **`t+pas`**

**C'est ce tableau dynamique (`TableauDyn`) qui devra être utilisé pour des accès en écriture sécurisés**

## 4.

## TRAITEMENT D'UNE EXCEPTION – EXEMPLE

## Exception déclenchée par la JVM (4/5)

```
public class TableauDyn extends Tableau {  
    protected int pas_=0;  
    /** création d'une nouvelle instance de TableauDyn  
     * @param t taille du tableau originel  
     * @param pas, pas d'augmentation de taille du tableau  
     */  
    public TableauDyn(int t, int pas) { super(t); pas_ += pas; }  
    /** écriture d'un élément dans le tableau  
     * @param i indice du tableau  
     * @param mot élément à écrire  
     */  
    public void puts(int i, String mot) {  
        try { super.put(i, mot); }  
        catch (IndexOutOfBoundsException e) {  
            System.out.println("Elargissement du tableau à une taille  
                de " +(tab_.length+pas_));  
            String[] newtab = new String[tab_.length + pas_];  
            for (int j = 0;j < tab_.length;j++) newtab[j] = tab_[j];  
            tab_ = newtab; puts(i, mot);  
        }  
    }  
}
```

## 4.

## TRAITEMENT D'UNE EXCEPTION – EXEMPLE

## Exception déclenchée par la JVM (5/5)

```
public class testTableauDyn {  
    public static void main(String[] args) {  
        TableauDyn phrase = new TableauDyn(2,5);  
        String mot;  
  
        phrase.puts(0, "le");  
        mot = phrase.get(0); System.out.println(mot);  
        phrase.puts(1, "livre");  
        mot = phrase.get(1); System.out.println(mot);  
        phrase.puts(2, "de");  
        mot = phrase.get(2); System.out.println(mot);  
    }  
}
```

```
le  
livre  
Elargissement du tableau à une taille de 7  
de
```

## Principe et Utilisation

### Propagation de l'exception aux méthodes appelantes

Extension de la surveillance des déclenchements d'exception aux instructions des méthodes appelées

Séparation du déclenchement et du traitement

### **IdentMethode () throws TypeException {blocM}**

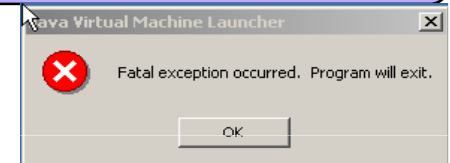
Si déclenchement d'une exception par une instruction de blocM  
traitement par le bloc try contenant l'appel à la méthode IdentMethode()

## 5. PROPAGATION DE L'EXCEPTION - EXEMPLE

## Exception déclenchée par la JVM

```
public class test2TableauDyn {  
    public static void main(String[] args) {  
        TableauDyn phrase = new TableauDyn(2,5);  
        String mot;  
        phrase.puts(0, "le");  
        mot = phrase.get(0); System.out.println(mot);  
        phrase.puts(1, "livre");  
        mot = phrase.get(1); System.out.println(mot);  
        phrase.puts(2, "de");  
        mot = phrase.get(2); System.out.println(mot);  
        mot = phrase.get(10); System.out.println(mot);  
    }  
}
```

## Erreur et arrêt de l'exécution



```
le  
livre  
Elargissement du tableau à une taille de 7  
de  
java.lang.ArrayIndexOutOfBoundsException: 10  
at sem16.Tableau.get(Tableau.java:21)  
at sem16.Exo3.main(Exo3.java:21)
```

## Objectifs du programme TableauDyn2.java

### Sécuriser le composant TableauDyn

**Contrainte** : le composant `TableauDyn` existe, il n'est pas remis en question, on souhaite juste sécuriser son utilisation

**Deuxième étape de la sécurisation** : lancer une exception en cas de lecture d'un élément à une position exclue du tableau originel

### Solution retenue :

Dans cette étape de sécurisation seule la méthode `gets` sera définie. Un **nouveau type d'exception** (utilisateur) sera **créé**. La méthode `gets` devra **déclencher une exception** de ce nouveau type si la position de l'élément à lire est exclue du tableau originel

`TableauDyn2` devra étendre `TableauDyn` en définissant la méthode `gets`

**C'est ce tableau dynamique (`TableauDyn2`) qui devra être utilisé pour des accès en lecture (et écriture) sécurisés**

## Exception déclenchée par throw (1/3)

### Création d'une exception HorsBornes

```
public class HorsBornes extends Exception {  
    /**  
     * Création d'une nouvelle instance de la classe HorsBorne  
     * @param i indice hors bornes  
     */  
    public HorsBornes(int i) {  
        super("Indice " + i + " hors bornes, recommencez...");  
    }  
}
```



## 5.

## PROPAGATION DE L'EXCEPTION – EXEMPLE

## Exception déclenchée par throw (2/3)

```
public class TableauDyn2 extends TableauDyn {
    /** création d'une nouvelle instance de TableauDyn2
     * @param t taille du tableau originel
     * @param pas, pas d'augmentation de taille du tableau
     */
    public TableauDyn2(int t, int pas) { super(t, pas); }

    /** lecture d'un élément du tableau
     * @param i indice du tableau
     * @return élément
     */
    public String gets(int i) throws HorsBornes {
        if ((i >= 0) && (i < tab_.length)) return super.get(i);
        else
            throw new HorsBornes(i);
    }
}
```

## 5. PROPAGATION DE L'EXCEPTION - EXEMPLE

## Exception déclenchée par throw (3/3)

```
public class testTableauDyn2 {  
    public static void main(String[] args) {  
        TableauDyn2 phrase = new TableauDyn2(2,5);  
        phrase.puts(0, "le"); phrase.puts(1, "livre");  
        String mot = null;  
        do {  
            try {  
                mot = phrase.gets(Keyboard.getInt("Indice du mot ? "));  
            }  
            catch (HorsBornes e){ System.out.println(e.getMessage()); }  
        } while (mot == null);  
        System.out.println(mot);  
    }  
}
```

```
Indice du mot ? 32  
Indice 32 hors bornes, recommencez...  
Indice du mot ? -1  
Indice -1 hors bornes, recommencez...  
Indice du mot ? 1  
livre
```