

M2103 (POO)



Bases de la programmation orientée objet

Responsable : Cristina Onete

cristina.onete@gmail.com

<http://onete.net/#teaching>

Les interfaces

The background features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the right side of the slide, creating a modern, layered effect. The text 'Les interfaces' is centered in a clean, sans-serif font.

Les interfaces en Java

- ▶ Une interface est un ensemble de caractéristiques liées décrites par des méthodes abstraites
 - ▶ Prenons l'exemple d'un match de sport (foot, rugby, handball, basket...)
 - ▶ Ces jeux ont des caractéristiques similaires :
 - ▶ À chaque match, il y a une équipe locale et des visiteurs
 - ▶ On a un système de points (buts, essais, penaltys, transformations)
 - ▶ On a un système de reprises, avec une durée déterminée
- ▶ Une interface Match définira ces caractéristiques en termes de méthodes
- ▶ Attention : une interface n'est pas une classe
 - ▶ Une interface est implémentée par des classes, qui concrétisent ses méthodes

L'interface Match

```
interface Match {  
    // les equipes  
    void equipeLocale(String nomEquipe);  
    void equipeVisiteuse(String nomAutreEquipe);  
  
    // les buts  
    int butMinimal();  
    int butSpeciale();  
    int autreBut();  
  
    // le score  
    int scoreEquipeLocale(int butMin, int butSpec, int autreBut);  
    int scoreVisiteurs(int butMin, int butSpec, int autreBut);  
    String vainqueur();  
}
```

← mot dédié : interface

← méthodes abstraites

← même sans mention public,
toute méthode est publique

Comment utiliser cette interface

```
public class Foot implements Match {  
    String equipe1, equipe2, victeur;  
    int score1, score2, temps;  
    // ... Un ou plusieurs constructeurs  
  
    public void equipeLocale(String nomEquipe){  
        this.equipe1 = nomEquipe;  
    }  
    public void equipeVisiteuse(String nomAutreEquipe){  
        this.equipe2 = nomAutreEquipe;  
    }  
    public void dureeDuMatch {  
        this.temps = 90;  
    }  
    public int butMinimal(){  
        return 1;}  
    public int butSpeciale(){  
        return 0;}
```

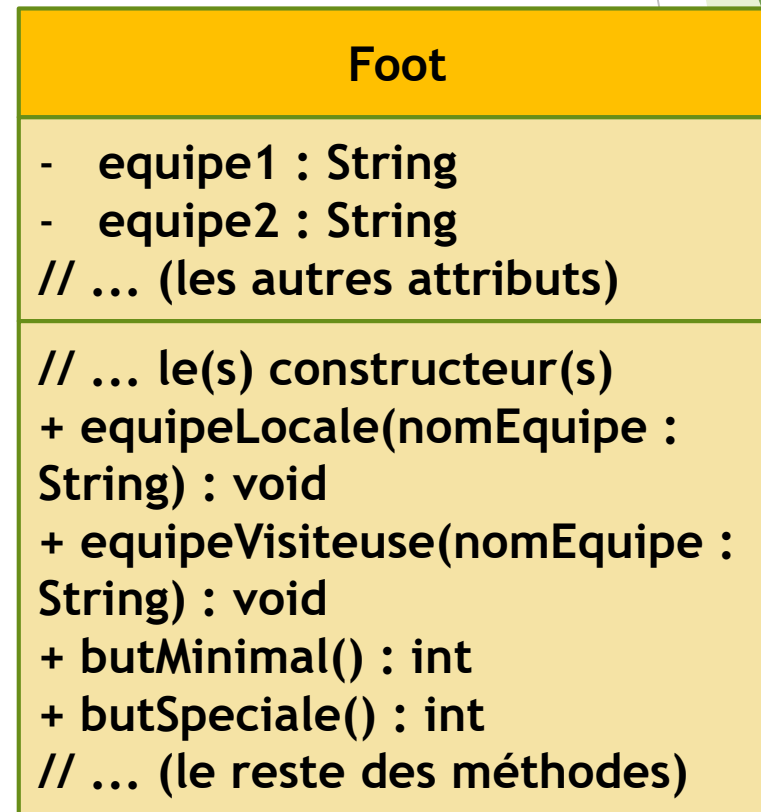
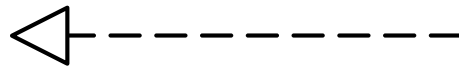
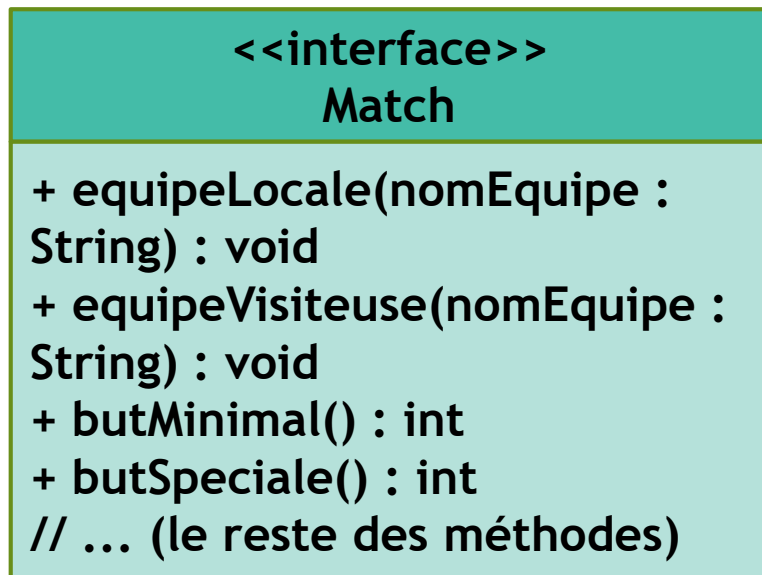
```
    public int autreBut(){  
        return 0;}  
    public int scoreEquipeLocale(int butMin, int butSpec, int  
    autreBut){  
        this.score1 = butMin * butMinimal();  
        return this.score1;}  
    public int scoreVisiteurs(int butMin, int butSpec, int  
    autreBut){  
        this.score2 = butMin * butMinimal();  
        return this.score2;}  
  
    public String victeur(){  
        // ... code en détail  
    }  
}
```

Implémenter une interface

- ▶ Une classe qui implémente une interface détaille toutes ses méthodes
- ▶ Une interface peut être implémentée par plusieurs classes
 - ▶ Et une classe peut implémenter plusieurs interfaces

class NomClasse implements Interface1,Interface2

- ▶ Diagrammes de classe

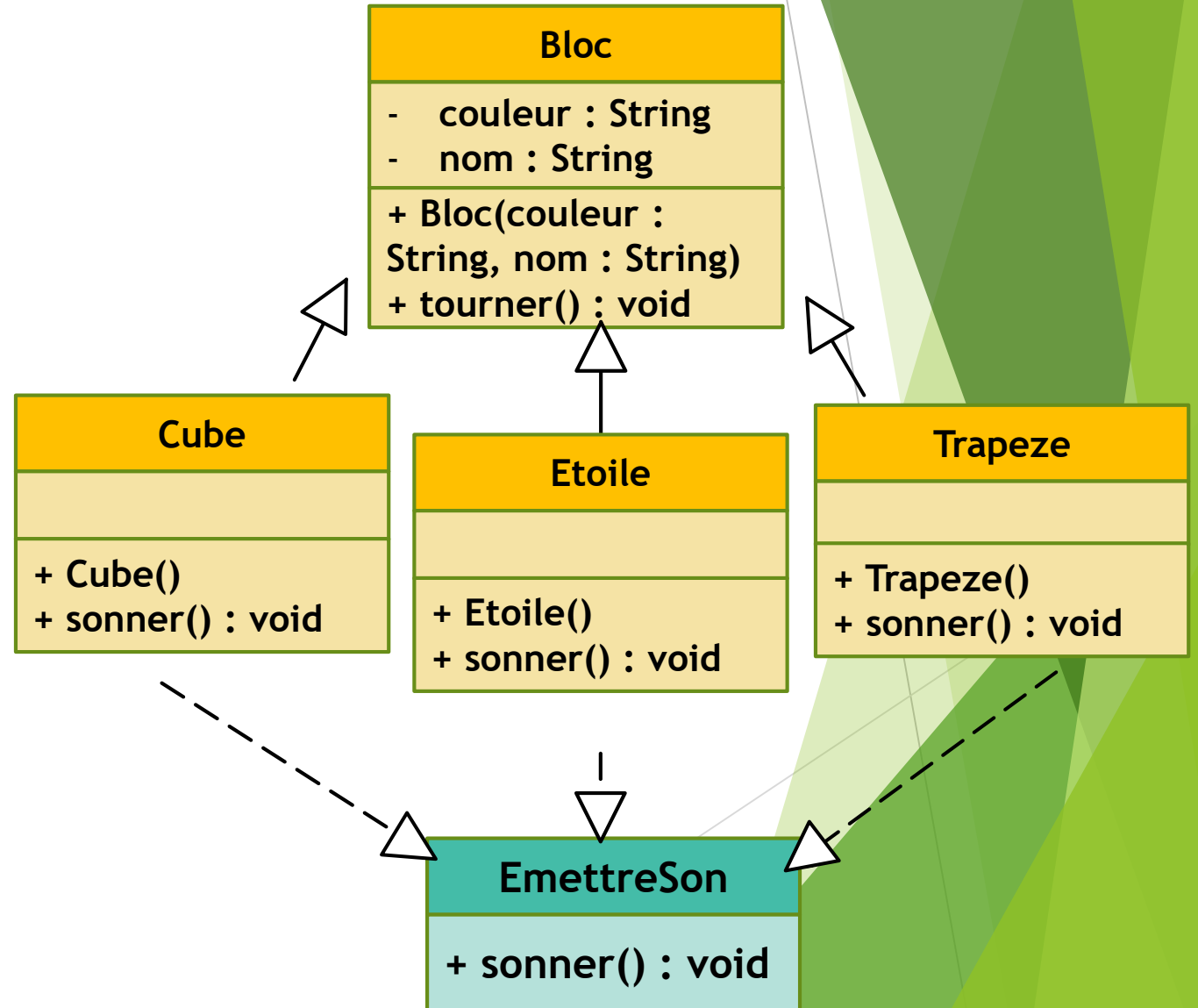
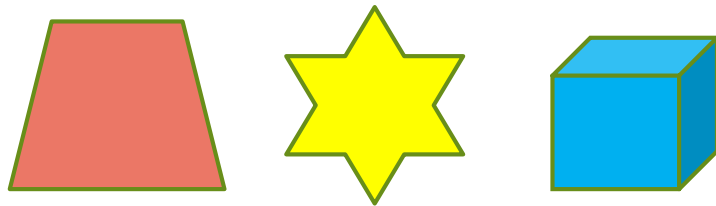


Interface vs. héritage

Héritage	Interface
<ul style="list-style-type: none">• = "est un type de"	<ul style="list-style-type: none">• = "fonctionne comme"
<ul style="list-style-type: none">• Relation superclasse - sous-classe(s)	<ul style="list-style-type: none">• Relation classe - interface
<ul style="list-style-type: none">• Mot clé extends	<ul style="list-style-type: none">• Mot clé implements
<ul style="list-style-type: none">• Peut contenir des variables et méthodes concrètes	<ul style="list-style-type: none">• Ne contient jamais de variables• N'a que des déclarations de méthodes
<ul style="list-style-type: none">• Une sous-classe peut redéfinir les méthodes de la superclasse• Mot clé @Override	<ul style="list-style-type: none">• Une classe concrète doit détailler toutes les méthodes de l'interface

Exemple : interface et héritage

- ▶ On se rappelle de nos blocs :
 - ▶ Chaque bloc émet un son différent
 - ▶ Ils peuvent également être tournés

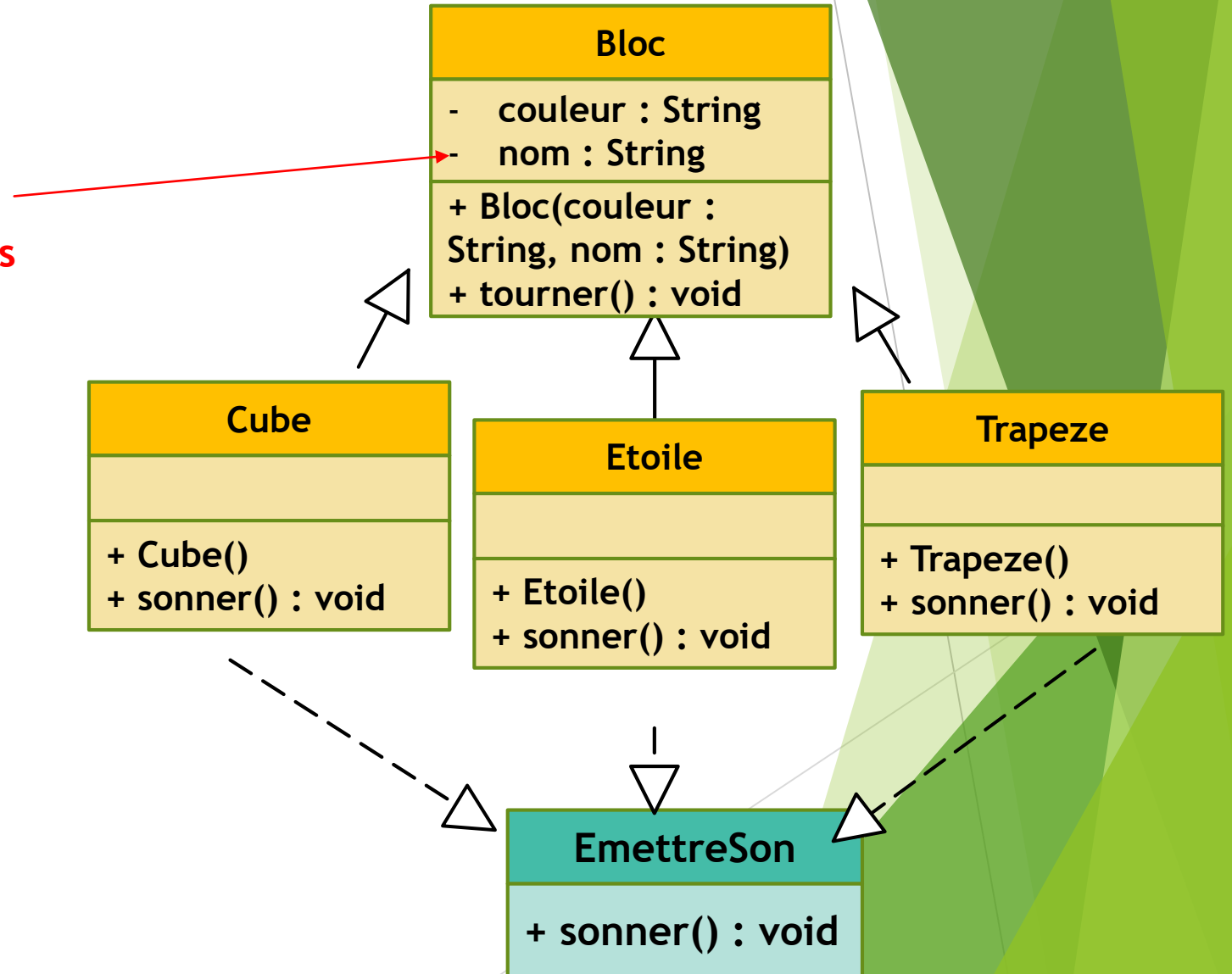


Exemple : interface et héritage

Les attributs sont privés

Alors on ne peut pas y accéder des classes Cube, Etoile, Trapeze

Il nous faut des méthodes get



Les exceptions

The background features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the right side of the slide, with some extending towards the center. The overall aesthetic is clean and modern.

C'est quoi, une exception ?

- ▶ Une exception est un objet
- ▶ Elle est instanciée lors d'un événement qui interrompt le flot normal d'un programme
- ▶ Une exception peut apparaître lors de l'arrêt d'un programme, ou dans un comportement aberrant

Anticiper et traiter ce comportement peut éviter que des erreurs importantes se produisent

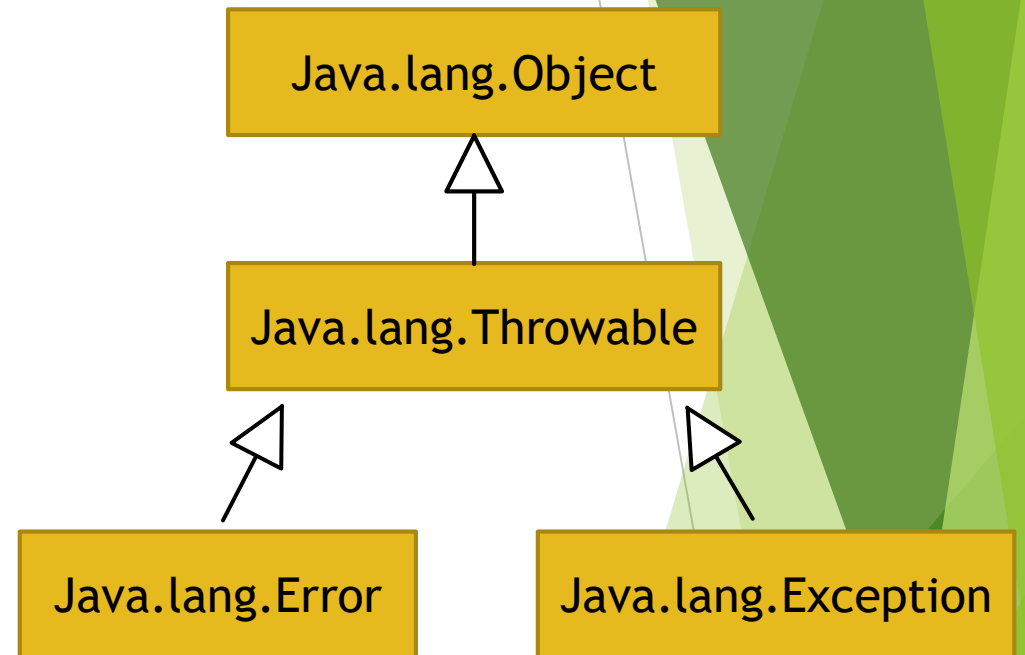
Des divers types d'exceptions

► Error :

- un type d'exception terminale : il arrête l'exécution d'un programme
- Les erreurs signalent un défaut grave et ne doivent pas être traitées (nous devons savoir si elles se produisent)
- VirtualMachineError, OutOfMemory, ...

► Exception :

- Moins grave qu'une erreur, peuvent toutefois créer des problèmes
- Deux types : exceptions à **compilation**, exceptions à **l'exécution (runtime)**
- **IOException**, **SQLException**, **NullPointerException**



Les exceptions se lèvent ...

- ▶ Si une faille **technique** interrompt le fonctionnement du programme
 - ▶ Par exemple si un index déborde (par rapport à la taille d'un tableau)
IndexOutOfBoundsException
 - ▶ Ou si la Java Virtual Machine rencontre un problème :
OutOfMemoryError
- ▶ Si une faille **applicative** interrompt le fonctionnement du programme
 - ▶ Par exemple si un fichier donné en entrée n'est pas trouvé :
FileNotFoundException

Comment lever des exceptions

- ▶ Mod dédié : throw
- ▶ Lever l'exception dans une clause if :

```
if (monFichier == null ){  
    throw new FileNotFoundException("Ce fichier n'existe pas.");  
}
```

- ▶ Capturer l'exception dans un bloc try-catch (mots dédiés)

```
try{  
    // code a executer  
}  
catch (FileNotFoundException e) {  
    // code a executer si l'exception est levee  
}
```

Contrôler les exceptions

- ▶ Une exception peut être contrôlée ou non (**checked/unchecked**)
- ▶ Une méthode peut contrôler elle même les exceptions :

```
public void lire() {  
    try{  
        return (new FileReader(monFichier));  
    } catch (FileNotFoundException e){ //traitement erreur  
    }  
}
```

- ▶ ... ou non

```
public void lire() throws FileNotFoundException{  
    ... // autre code  
    return (new FileReader(monFichier));  
}
```

A savoir

- ▶ Les blocs try-catch peuvent avoir multiples blocs catch
 - ▶ Mais il n'y a qu'un bloc try, qui contient un code commun
 - ▶ Si blocs catch multiples : exceptions en hiérarchie inversée
 - ▶ Sous-classes avant superclasses : `NullPointerException` avant `RuntimeException`
- ▶ Les blocs try peuvent avoir un bloc finally après 0, 1, 2... blocs catch
 - ▶ Il s'exécute obligatoirement
 - ▶ Il permet de bien clôturer des processus ouverts (e.g. un fichier ouvert)
- ▶ Deux instructions utiles :
 - ▶ `System.err.println(String)` : une méthode qui permet d'afficher des textes lors du lever d'une exception
 - ▶ `printStackTrace()` : méthode applicable à tout objet qui implémente l'interface `Throwable`; elle trace la source de l'erreur ou exception

Un exemple : index out of Bounds

Regardons ce code

MonException.java

```
1 |
2 public class MonException {
3     public static void main(String[] args) {
4         MonException objet = new MonException(); // nous avons besoin d'un objet pour pouvoir utiliser les methodes en bas
5         int taille = 3;
6         int[] monTableau = new int[taille];
7         int monEntree = 10;
8         monTableau[0] = monEntree;
9         System.out.println("La premiere composante du tableau est " + monTableau[0]);
10        objet.doubler(monEntree);
11        monTableau[objet.augmenter(0)] = objet.doubler(monEntree);
12        System.out.println("La deuxieme composante du tableau est " + monTableau[1]);
13    }
14 }
15
16 public int augmenter (int index) {
17     System.out.println("La valeur initiale de l'index est " + index);
18     index = index + 1;
19     System.out.println("L'index a ete augmente a " + index);
20     return index;
21 }
22
23 public int doubler (int valeur) {
24     System.out.println("Nous allons doubler la valeur " +valeur);
25     valeur = valeur * valeur;
26     System.out.println("Valeur doublee " + valeur);
27     return valeur;
28 }
29 }
30
```

Lors d'une execution normale

Problems Javadoc Declaration Console

```
<terminated> MonException [Java Application] C:\Program Files\Java\
La premiere composante du tableau est 10
Nous allons doubler la valeur 10
Valeur doublee 100
La valeur initiale de l'index est 0
L'index a ete augmente a 1
Nous allons doubler la valeur 10
Valeur doublee 100
La deuxieme composante du tableau est 100
```

Avec une erreur

```
1 ErrorProgram/src/MonException.java
2 public class MonException {
3     public static void main(String[] args) {
4         MonException objet = new MonException(); // nous avons besoin d'un objet pour pouvoir utiliser les methodes en bas
5         int taille = 1;
6         int[] monTableau = new int[taille];
7         int monEntree = 10;
8         monTableau[0] = monEntree;
9         System.out.println("La premiere composante du tableau est " + monTableau[0]);
10        objet.doubler(monEntree);
11        monTableau[objet.augmenter(0)] = objet.doubler(monEntree);
12        System.out.println("La deuxieme composante du tableau est " + monTableau[1]);
13    }
14 }
15
```

taille changée

```
Problems Javadoc Declaration Console
<terminated> MonException [Java Application] C:\Program Files\Java\jre1.8.0_
La premiere composante du tableau est 10
Nous allons doubler la valeur 10
Valeur doublee 100
La valeur initiale de l'index est 0
Exception in thread "main" L'index a ete augmente a 1
Nous allons doubler la valeur 10
Valeur doublee 100
java.lang.ArrayIndexOutOfBoundsException: 1
    at MonException.main(MonException.java:11)
```

Lever l'exception

```
1
2 public class MonException {
3     public static void main(String[] args) {
4         MonException objet = new MonException(); // nous avons besoin d'un objet pour pouvoir utiliser les methodes en bas
5         int taille = 1;
6         int[] monTableau = new int[taille];
7         int monEntree = 10;
8         monTableau[0] = monEntree;
9         System.out.println("La premiere composante du tableau est " + monTableau[0]);
10        objet.doubler(monEntree);
11        try {
12            monTableau[objet.augmenter(0)] = objet.doubler(monEntree);
13            System.out.println("La deuxieme composante du tableau est " + monTableau[1]);
14        }
15        catch(IndexOutOfBoundsException e) {
16            System.out.println("Taille du tableau depassee !");
17        }
18    }
19
20    public int augmenter (int index) {
21        System.out.println("La valeur initiale de l'index est " + index);
22        index = index + 1;
```

```
Problems Javadoc Declaration Console
<terminated> MonException [Java Application] C:\Program Files\Jav
La premiere composante du tableau est 10
Nous allons doubler la valeur 10
Valeur doublee 100
La valeur initiale de l'index est 0
L'index a ete augmente a 1
Nous allons doubler la valeur 10
Valeur doublee 100
Taille du tableau depassee !
```

try-catch

cherche une exception du type
IndexOutOfBoundsException

exception levée

Tracer la source de l'erreur

```
2 public class MonException {
3     public static void main(String[] args) {
4         MonException objet = new MonException(); // nous avons besoin d'un objet pour pouvoir utiliser les methodes en bas
5         int taille = 1;
6         int[] monTableau = new int[taille];
7         int monEntree = 10;
8         monTableau[0] = monEntree;
9         System.out.println("La premiere composante du tableau est " + monTableau[0]);
10        objet.doubler(monEntree);
11        try {
12            monTableau[objet.augmenter(0)] = objet.doubler(monEntree);
13            System.out.println("La deuxieme composante du tableau est " + monTableau[1]);
14        }
15        catch(IndexOutOfBoundsException e) {
16            System.err.println("Taille du tableau depassee !");
17            e.printStackTrace();
18        }
19        finally {
20            System.out.println("Dans le bloc finally");
21        }
22    }
23 }
```

Problems Javadoc Declaration Console

```
<terminated> MonException [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.exe (Jan 17, 2018, 1:43:26 PM)
Nous allons doubler la valeur 10
Valeur doublee 100
La valeur initiale de l'index est 0
L'index a ete augmente a 1
Nous allons doubler la valeur 10
Valeur doublee 100
Taille du tableau depassee !
java.lang.ArrayIndexOutOfBoundsException: 1
    at MonException.main(MonException.java:12)
Dans le bloc finally
```

System.out.println remplacé par
System.err.println

tracer l'erreur

Tracer la source de l'erreur

```
2 public class MonException {
3     public static void main(String[] args) {
4         MonException objet = new MonException(); // nous avons besoin d'un objet pour pouvoir utiliser les methodes en bas
5         int taille = 1;
6         int[] monTableau = new int[taille];
7         int monEntree = 10;
8         monTableau[0] = monEntree;
9         System.out.println("La premiere composante du tableau est " + monTableau[0]);
10        objet.doubler(monEntree);
11        try {
12            monTableau[objet.augmenter(0)] = objet.doubler(monEntree);
13            System.out.println("La deuxieme composante du tableau est " + monTableau[1]);
14        }
15        catch(IndexOutOfBoundsException e) {
16            System.err.println("Taille du tableau depassee !");
17            e.printStackTrace();
18        }
19        finally {
20            System.out.println("Dans le bloc finally");
21        }
22    }
23 }
```

Problems Javadoc Declaration Console

<terminated> MonException [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.exe (Jan 17, 2018, 1:43:26 PM)

Nous allons doubler la valeur 10
Valeur doublee 100
La valeur initiale de l'index est 0
L'index a ete augmente a 1
Nous allons doubler la valeur 10
Valeur doublee 100
Taille du tableau depassee !
java.lang.ArrayIndexOutOfBoundsException: 1
at MonException.main(MonException.java:12)
Dans le bloc finally

Un bloc finally

Nouvelles exceptions & bonnes pratiques



Créer une nouvelle exception

- ▶ Nous pouvons créer de nouvelles exceptions en Java



BP1 : Toujours d'abord utiliser les exceptions qui existent déjà en Java !

- ▶ Si contrôlée, l'exception hérite impérativement de la classe Exception
- ▶ Si non-contrôlées, l'exception hérite de la classe RuntimeException
- ▶ La nouvelle exception formera une nouvelle classe, avec des attributs et des méthodes.
 - ▶ N'oubliez pas d'utiliser l'héritage d'Exception/RuntimeIOException !

Exemple : Etudiant non-trouvé

```
public class EtudiantNonTrouve extends RuntimeException{  
    // constructeur personnalisé  
    public EtudiantNonTrouve(String m){  
        super(m); // nous utilisons le constructeur d'Exception  
    }  
}
```

```
public class ClassePrincipale{  
    public static void main(String[] args) {  
        Etudiant[] mesEtudiants = new Etudiant[5];  
        for (int i = 0; i < mesEtudiants.length; i++){  
            if (mesEtudiants[i].getNom().equals("Dupont")){  
                ...  
            }  
            else  
                throw new EtudiantNonTrouve("Aucun Dupont dans la classe !");  
        }  
    }  
}
```

D'autres bonnes pratiques

- ▶ Important de capturer toute exception impactant notre code



BP2 : Utiliser les sous-classes spécifiques des exceptions, plutôt que les superclasses (Exception/RuntimeException)

- ▶ Les blocs try-catch-finally ont des structures spéciales



BP3 : Ne jamais utiliser les blocs catch{} comme branche alternative d'exécution (c'est pourquoi on a if-then-else !)



BP4 : Ne jamais utiliser return lorsqu'une exception est levée

Vos questions

Question 1

- ▶ Je souhaiterais savoir comment accéder aux informations d'un tableau. Pour le C/C++ il s'agissait de `> nom_du_tab[n° de la case].ce_que_je_cherche <`. Je voudrais savoir comment faire en java.

REPONSE :

- ▶ La bonne nouvelle est : c'est à peu près la même chose qu'en C/C++, mais la syntaxe dépend de ce qu'on cherche
- ▶ Par exemple : `Pokemon[] pokemons = new Pokemon[5];`
- ▶ Le nom du premier pokemon : `pokemons[0].getNom();`

Un pas plus loin

Animal

- nom : String
- maitre : Personne
- // d'autres attributs

+ getNom()
// d'autres méthodes

Personne

- nom : String
- numeroSS : int
- animaux: Animal[]

+ getAnimaux() : Animal[]
// d'autres méthodes

- ▶ Dans la classe Personne :

```
this.animaux[0].getNom()
```

- ▶ Dans une troisième classe :

```
<un objet de type Personne>.getAnimaux()[0].getNom()
```

Question 2:

- ▶ Comment fait on pour afficher la valeur actuelle d'un objet?

REPONSE :

- ▶ Un objet = une instance de classe
- ▶ En mémoire :

```
<valeur_attribut1> <valeur.attribut2> .....  
|                |                |  
|                |                |  
|                |                |
```

- ▶ "Afficher un objet" est donc une notion ambiguë
- ▶ Cependant, on pourrait vouloir afficher les valeurs de ses attributs
- ▶ Trois choix pour le faire

Le choix standard : toString()

- On écrit une méthode toString() dans laquelle on affiche les attributs

```
public String toString(){
    String animaux = "{}";
    for(int i = 0; i<this.animaux.length; i++){
        animaux += this.animaux[i].getNom()
                + ",";
    }
    return("[ " + this.nom + ", " +this.numeroSS
        + animaux + "}]");
```

- Pour afficher :

```
System.out.println(<nom d'objet>);
```

Personne

- nom : String
- numeroSS : int
- animaux: Animal[]

+ getAnimaux() : Animal[]
// d'autres méthodes

Un deuxième choix

- On écrit une autre méthode avec sortie String

```
public String afficherPersonne(){
    String animaux = "{}";
    for(int i = 0; i<this.animaux.length;
i++){
        animaux += this.animaux[i].getNom()
                + ",";
    }
    return("[ " + this.nom + ", "
+this.numeroSS
        + animaux + "}]");
}
```

- Pour afficher :

```
System.out.println(<nom d'objet>.afficherPersonne());
```

Personne

- nom : String
- numeroSS : int
- animaux: Animal[]

+ getAnimaux() : Animal[]
// d'autres méthodes

Un troisième choix

- On écrit une méthode pour l'affichage

```
public void afficherPersonne(){
    String animaux = "{}";
    for(int i = 0; i<this.animaux.length;
i++){
        animaux += this.animaux[i].getNom()
                + ",";
    }
    System.out.println("[ " + this.nom + " ,
    " +this.numeroSS + animaux + "}]");
}
```

- Pour afficher :
<nom d'objet>.afficherPersonne();

Personne

- nom : String
- numeroSS : int
- animaux: Animal[]

+ getAnimaux() : Animal[]
// d'autres méthodes