Chapitre 5:

Les exceptions

Les exceptions

- La gestion des exceptions est très importante, voire primordiale, dans tout système informatique. Elle confère un aspect beaucoup plus professionnel aux applications.
- Eviter les applications qui plantent sans information précise

- Les exceptions sont des instances des classes héritant des classes
  - **`java.lang.Error`** (pour des erreurs graves quasi impossibles à gérer : plus de mémoire, classe manquante, ...)
  - **`java.lang.Exception`** (pour des exceptions attendues sinon probables pouvant être gérée: débordement d'un tableau, erreur de calcul, ...).
- Ces deux classes implémentent l'interface **`Throwable`**

Les exceptions

- La classe Exception possède deux constructeurs `Exception()` et `Exception(String msg)`
- Elle implémente l'interface `Throwable` :
  - `getMessage()` qui permet de récupérer le message de l'exception s'il existe.
  - `toString()` qui retourne la classe et le message sous forme de chaîne.
  - `printStackTrace()` qui fait appel à `toString()`, mais qui en plus indique l'endroit du programme où a été levée l'exception.

Traiter les exceptions

- Quand une exception est lancée, les instructions suivantes sont ignorées.
- La levée d'une exception provoque une remontée dans l'appel des méthodes jusqu'à ce qu'un bloc catch acceptant cette exception soit trouvé.
- Si aucun bloc catch n'est trouvé, l'exception est capturée par l'interpréteur (JVM) qui l’affiche et le programme s'arrête.

Capture d'une exception

- Le traitement d'une exception permet de séparer un bloc d'instructions de la gestion des erreurs pouvant survenir dans ce bloc.

```
try {
   // Code pouvant lever des Exception
   // (exemple : ouverture de fichier)
} catch (Exception e) {
    // Gestion des Exceptions
    // on écrit ici le code pour récupérer le programme
    // malgré cette erreur
    // on peut se contenter d'afficher un message et
    // poursuivre le programme
}
```

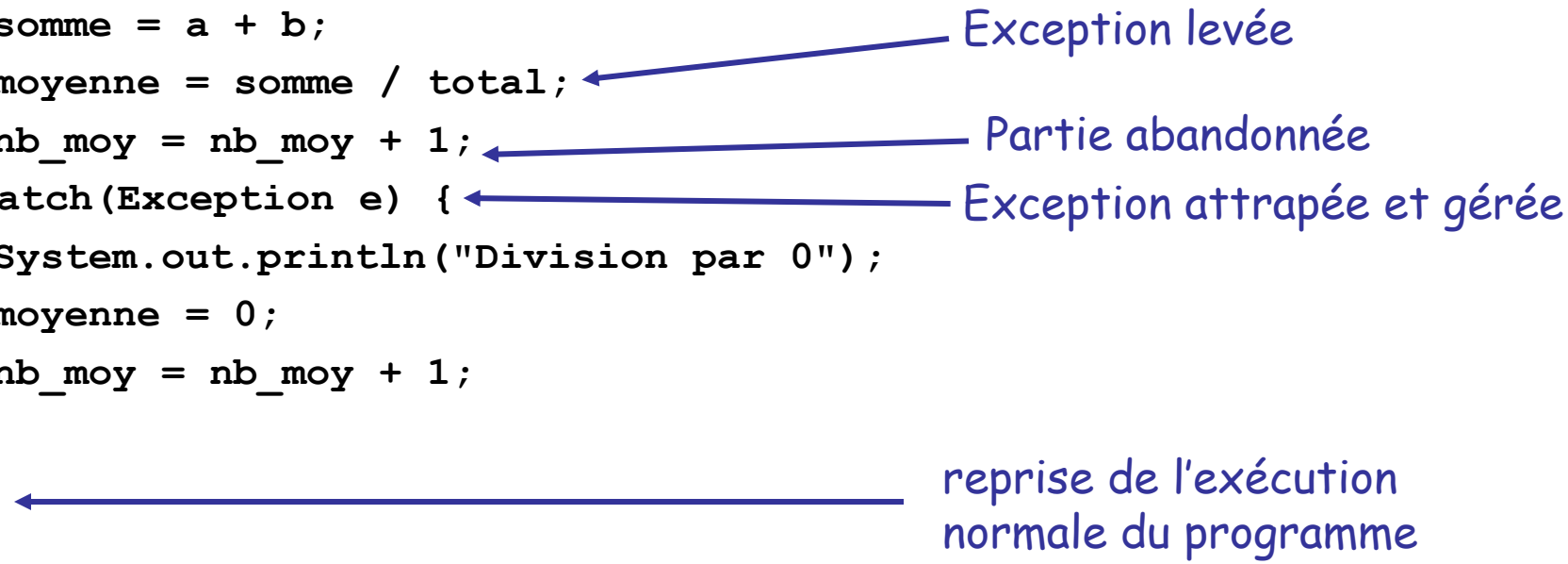
Capture de plusieurs exceptions

- Un bloc optionnel **`finally`** peut-être posé à la suite des **`catch`**. Son contenu sera exécuté quelque soit il y a eu une exception ou pas.

```
...
try {
   //endroit où pourrait apparaître une exception
} catch (type_exception_1 e1) {
   // traitement de l'exception de type 1
} catch (type_exception_2 e2) {
   // traitement de l'exception de type 2
} finally {
   // dans tous les cas, passer par ici
}
...
```

Exemple (1)

```
...
total = 0;
try {
   somme = a + b;
   moyenne = somme / total;
   nb_moy = nb_moy + 1;
} catch(Exception e) {
   System.out.println("Division par 0");
   moyenne = 0;
   nb_moy = nb_moy + 1;
}
...
```

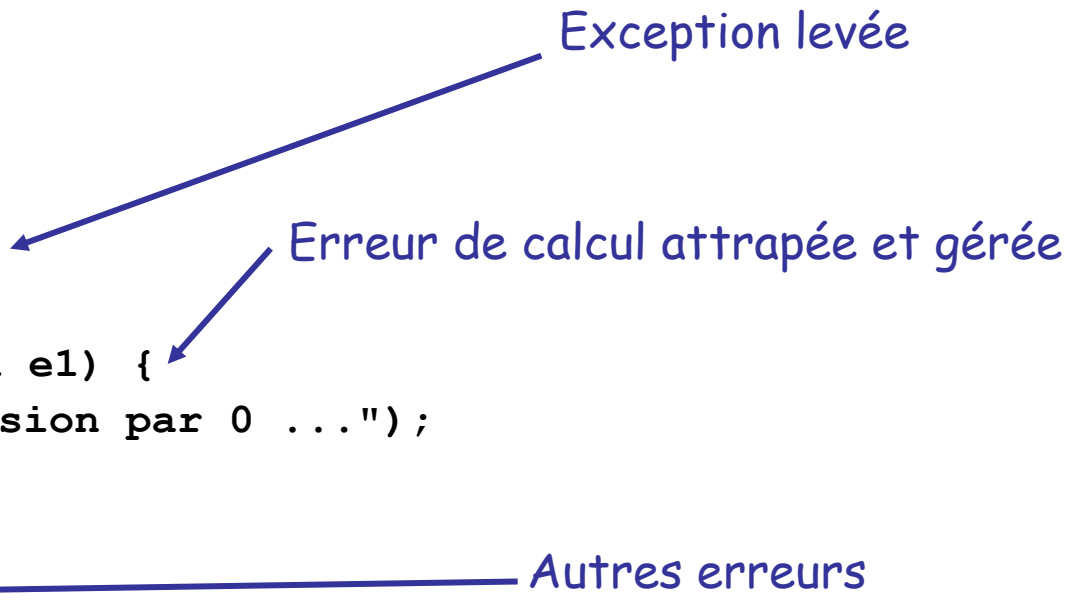


Exemples de classes d'exceptions

```
ClassCastException
IllegalArgumentException
ArithmeticException
ClassNotFoundException
NoSuchFieldException
NoSuchMethodException
RuntimeException
NumberFormatException
ArrayIndexOutOfBoundsException
StringIndexOutOfBoundsException
NullPointerException
```

Exemple (2)

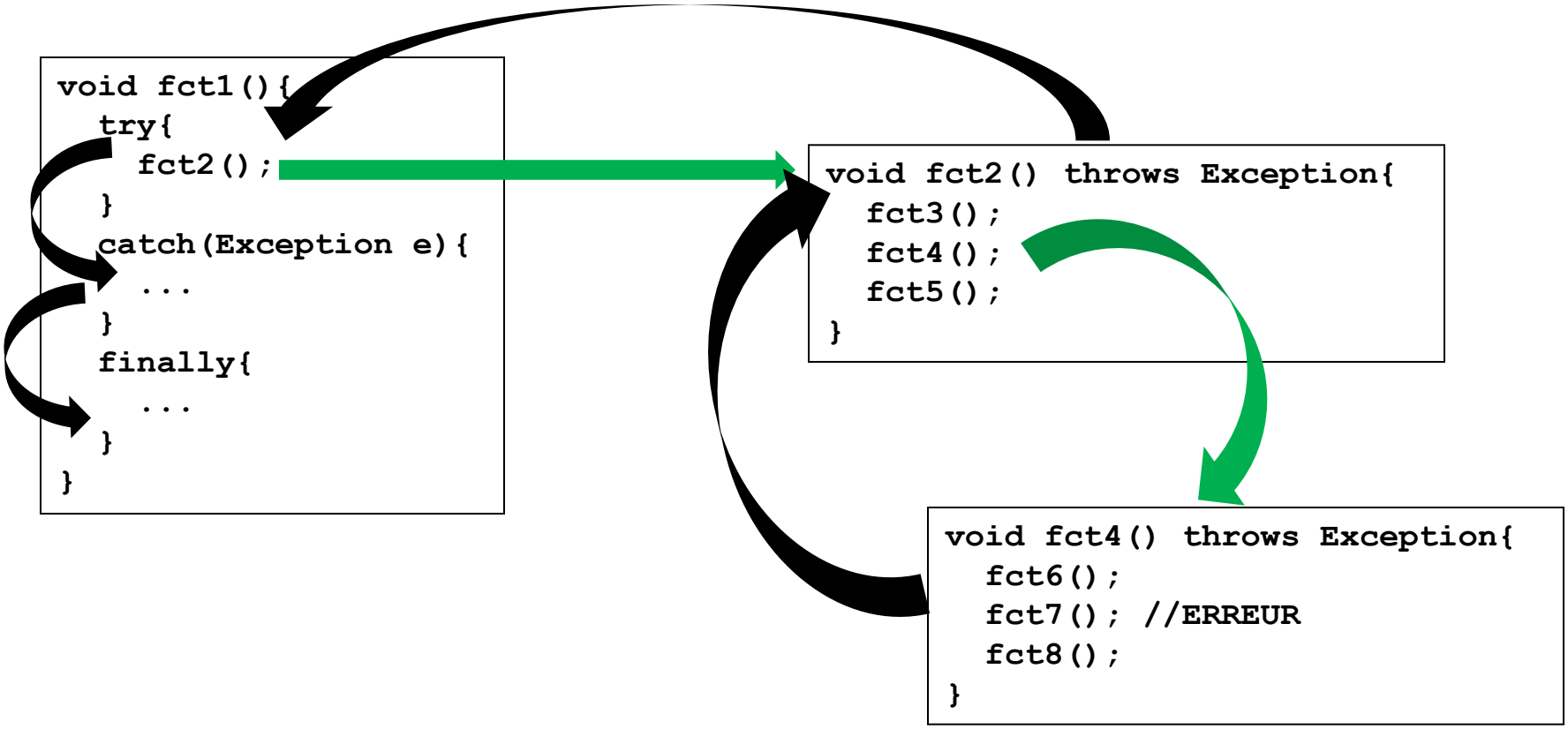
```
...
total = 0
try {
   somme = a + b;
   moyenne = somme / total;
   nb_moy = nb_moy + 1;
} catch(ArithmeticException e1) {
   System.out.println("Division par 0 ...");
   moyenne = 0;
   nb_moy = nb_moy + 1;
} catch(Exception e2) {
   System.out.println("Une autre erreur s'est produite");
}

...
```



Propagation d'une exception

- Si une exception n'est pas gérée dans une procédure, elle se propage hors de la procédure,
- Il faut donc que cette procédure soit capable de propager l'exception (utilisation du mot clé `throws`),
- Si à la fin du programme, une exception n'a pas été attrapée, la pile des méthodes traversées par l'exception est affichée.

Propagation d'une exception: exemple



Exemple (3)

```
void moyenne() throws Exception {
   total = 0;
   somme = a + b;
   moyenne = somme / total;
   nb_moy = nb_moy + 1;
}

public static void main(String args[]) {
   try {
        moyenne();
   } catch(Exception e) {
        System.out.println("Division par 0");
   }
}
```

Exception propagée

Exception levée

Exception attrapée et gérée

Définir son exception

- Il est possible de définir une exception en la faisant dériver de la classe Exception :

```
public class ExceptionMoyenne extends Exception{
  public ExceptionMoyenne(String msg){
     super(msg);
  }
  public ExceptionMoyenne(){
     super("Problème lors de calcul de la moyenne");
  }
  public String toString(){
    return "Problème lors de calcul de la moyenne";
  }
}
```

Throw: lancer une exception

- Il est possible de lancer et propager une exception
- **`throw`**: permet lancer une instance d'exception sur une instruction donnée
- On peut aussi lancer les exceptions natives de l'API Java
- Pour pouvoir lancer une exception:
  - On doit la déclarer dans l'entête: avec **`throws`** dans l'entête de la fonction
  - Ou bien la mettre dans un bloc **`try`** … **`catch`** …

```
int stock() throws PlusDeStockException {
  if (Quantite == 0)
    throw new PlusDeStockException ("Je n'ai plus de stock pour l'article " + designation);
  return (Quantite) ;
}
```

Exemple (4)

```
void moyenne() throws ExceptionMoyenne {
    total = 0;
    somme = a + b;
    if (total == 0) throw (new ExceptionMoyenne());
    moyenne = somme / total;
    nb_moy = nb_moy + 1;
}

public static void main(String args[]) {
    try {
        moyenne();
    } catch(ExceptionMoyenne e){
          System.out.println(e.getMessage());
          System.out.println(e);    // e.toString()
    }
}
```

Exception propagée

Exception lancée

Exception attrapée et gérée

Le mot clé **`throw`** permet de lancer son exception.

Exercice

```
public class Somme {
    public static void main(String[] args) {
        String tab[] = {"12", "55", "ab", "63", "29"};
        System.out.println("Longueur de tab = " + tab.length);
        int somme = 0;
        for (int i = 0; i < tab.length; i++) {
            somme += Integer.parseInt(tab[i]);
        }
        System.out.println("somme = " + somme);
    }
}
```

Quelle exception peut se produire? Pensez à résoudre ce problème :

1. En utilisant une classe d'exception prédéfinie.
2. En utilisant une classe d'exception prédéfinie avec un message personnalisé.
3. En utilisant votre propre classe d'exception (donnez un nom et un message significatifs).