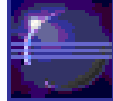
	<b>TP</b> <b>Gestion des exceptions en JAVA</b>	
---	--	---

Objectifs	Temps alloué	Outils
<ul style="list-style-type: none"> <li>- S'initier à la capture et au traitement des exceptions par l'utilisation des blocs try/catch.</li> <li>- Tester la création d'exceptions personnalisées.</li> </ul>	3h00	Eclipse

<b>Exercice n°1 :</b>	<b>[Initiation au traitement des exceptions]</b>
-----------------------	--

Soit le programme « ExceptionTest.java » suivant :

```
import java.util.Scanner;
public class exceptionTest {
    public static void main(String[] args) {
        int a, b, res;
        Scanner clavier = new Scanner(System.in);
        System.out.println("a =");
        a = clavier.nextInt();
        System.out.println("b =");
        b = clavier.nextInt();
        res = a / b;
        System.out.println(a + " / " + b + " = " + res);
        clavier.close();
        System.out.println ("Fin du programme") ;
    }
}
```

A première vue, le programme ne présente aucun problème (pas d'erreurs de compilation). Seulement à l'exécution, vous vous apercevriez très vite que votre programme recèle une faille gênante puisque nous n'avons pas vérifié que le diviseur n'est pas égal à zéro.

1) Néanmoins, lancez le programme en saisissant une valeur nulle pour b.

- a. Le programme s'est-il exécuté correctement ?
- b. Le message "Fin du programme" est-il apparu ?
- c. Quelle exception a été levée par la machine Java ?

Nous allons maintenant faire en sorte que le programme ne se termine pas aussi brutalement, et nous renseigne un peu plus. Pour cela, nous allons mettre en place un bloc try/catch afin d'attraper l'exception levée précédemment

```
try {  
    res = a / b;  
    System.out.println(a + " / " + b + " = " + res);  
}  
catch( ArithmeticException e){  
    System.out.println("oops ! un problème dans la division");  
    System.out.println ("le message officiel est " + e.getMessage() );  
}
```

Relancez le programme en saisissant une valeur nulle pour b.

- d. Le programme a-t-il affiché qu'il y avait un problème dans la division ?
- e. Le message "Fin du programme" est-il apparu ?
- f. Quel est le message d'erreur officiel correspondant à une telle exception ?

A la suite de l'instruction catch vous allez maintenant rajouter le bloc finally suivant

```
finally {  
    System.out.println("le bloc finally sera toujours exécuté") ;  
    System.out.println("c'est là qu'on fermera par exemple les fichiers") ;  
}
```

Relancez le programme en saisissant encore une valeur nulle pour b.

- g. Le bloc finally a-t-il été exécuté ?

Mettez en commentaire le bloc catch et relancez le programme en saisissant encore une valeur nulle pour b.

- h. Le bloc finally a-t-il été exécuté ?
- i. L'exception a-t-elle été traitée ?

Pour terminer, vous allez maintenant relancer le programme en saisissant une lettre à la place d'un nombre

j. Que se passe-t-il ?

k. Quelle exception a été lancée ?

Corriger le problème en traitant l'exception lancée et en intégrant les lectures au clavier dans le bloc try.

#### Exercice n°2 :

#### [Utilisation des Exceptions]

La méthode `parseInt` est spécifiée ainsi :

```
public static int parseInt  
(String s) throws  
NumberFormatException
```

Parameters	s - a String containing the <b>int</b> representation to be parsed
Returns	the integer represented by the argument in decimal.
Description	Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value.
Exception	<code>NumberFormatException</code> if the string does not contain a parsable integer.

- 1) Utilisez cette méthode pour faire la somme de tous les entiers donnés en argument de la ligne de commande, les autres arguments étant ignorés.

<b>Exercice n°3 :</b>	<b>[Exceptions personnalisées et constructeurs]</b>
-----------------------	---

Une adresse IP est une suite de 4 octets représentée de la manière suivante :

192.168.10.37

- 1) Écrire la classe adresseIP.java ayant comme attribut un tableau d'entiers nommé **octet**.
- 2) Le constructeur de la classe adresseIP.java permet la création et l'initialisation du tableau octet par 4 entiers (o1, o2, o3, o4) qui sont passés en paramètre au constructeur.
- 3) Redéfinir la méthode toString() de manière à ce qu'elle affiche l'adresse IP sous la forme suivante : 192.168.10.37
- 4) Dans la méthode « main » créer une adresse IP et l'afficher.

Jusque-là normalement il n'y a pas de problème. Seulement, vous n'êtes pas sans savoir que les octets représentant une adresse IP sont obligatoirement compris entre 0 et 255.

- 5) Vous allez donc mettre en place un traitement par exception pour éviter que l'on puisse construire une adresse IP avec des nombres farfelus.
  - a. Pour cela, vous allez définir votre propre classe d'exception « ExceptionAdriP » dérivant de la classe de base Exception
  - b. Puis vous allez modifier le constructeur de la classe adresseIP de manière à ce qu'il propage des exceptions de type « ExceptionAdriP » lorsque l'un des octets de l'adresse IP n'est pas compris entre 0 et 255.
  - c. Et enfin, vous mettrez en place dans la fonction main le traitement des exceptions de type ExceptionAdriP afin que lors de la levée d'une telle exception, l'erreur soit récupérée et le message indiquant l'octet faux soit affiché.

**Exercice n°4 :****[Exceptions personnalisées & traitement séparé]**

Dans les failles de sécurité réseau, on trouve très souvent les problèmes de dépassement. Par exemple, sur certaines anciennes versions de *telnet*, un login ou un mot de passe de plus d'un mega-octet faisait "planter" le programme.

Cet exercice va gérer ce type de problème en séparant les exceptions pour une meilleure gestion.

Parmi les exceptions que notre programme peut lever nous citons :

- \* *WrongLoginException* qui se produit lorsque l'utilisateur saisit un login incorrecte
- \* *WrongPwdException* lorsque le mot de passe est erroné
- \* *WrongInputLength* lorsque le login où le pwd saisi dépasse 10 caractères.

Remarque: La lecture de l'entrée standard peut lever une *java.io.IOException*.

- 1) Implémentez les différentes classes d'exceptions de manière à ce que chacune affiche un message d'erreur correspondant.
- 2) Notre classe principale nommée « Authentification » est caractérisée par :
  - a. Les **constantes** : `LoginCorrect` et `PwdCorrect`, deux chaînes de caractères initialisées à « scott » et « tiger » (celui-ci étant le seul utilisateur référencé dans notre programme)
  - b. Les méthodes : `getUserLogin()` et `getUserPwd()` qui permettent de lire à partir du clavier le login et le password de l'utilisateur. Chacune de ces méthodes est susceptible de lever un ensemble d'exceptions que vous devez identifier.
  - c. Le constructeur de la classe « authentification » fait appel aux méthodes précédemment énoncées pour lire le login et le mot de passe.
- 3) Implémenter la classe « `TestAuthentification` » intégrant la méthode « `main()` ». Cette classe aura comme objectif la réalisation d'une authentification. Ce programme tournera en boucle, tant que la création d'une nouvelle authentification est incorrecte (capture d'exception).

**NB :** La gestion des exceptions doit être traitée de façon judicieuse et avec la granulosité la plus fine.