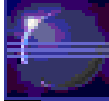
	<p style="text-align: center;">TP6</p> <p style="text-align: center;">Gestion des exceptions en JAVA</p>	
---	--	---

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

Exercice1 :

On veut écrire un programme Java qui définit la fonction mathématique suivante :

$f(n,x) = \sqrt{x/n}$ avec n un double non nul et x est un double positif.

- 1) Implémentez la classe « **Mathematique** » qui définit la fonction statique f.
- 2) Ecrivez un programme qui prend en paramètre deux doubles n et x et renvoie la valeur de f(n,x).

Ensuite essayez successivement ces cas, en exécutant le programme :

- Ne faire figurer aucun paramètre en ligne de commande
- Mettre un paramètre non numérique (contenant des lettres par exemple)
- Mettre une valeur nulle pour n
- Mettre un paramètre négatif pour la valeur de x

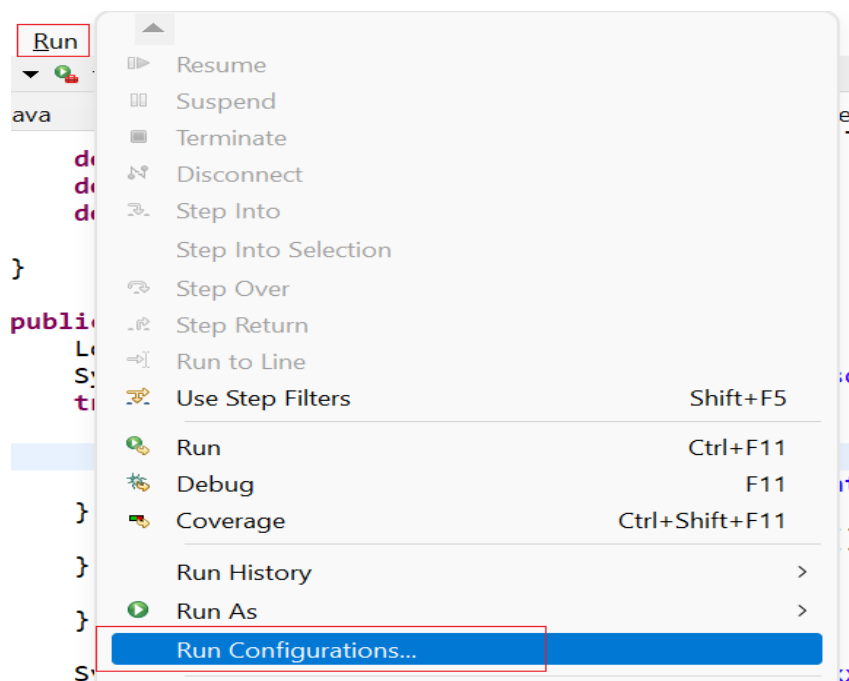
Dans les deux premiers cas, une exception est signalée. Dans les deux derniers cas, le résultat est faux. Vous devez modifier le programme pour que, dans chacun de ces cas, l'erreur soit attrapée par le programme et signalée à l'utilisateur.

- 1- S'il n'y a pas de paramètre sur la ligne de commande, il s'agit d'une exception de type **ArrayIndexOutOfBoundsException** ; on souhaite que le programme affiche par exemple : « Vous devriez entrer deux nombres sur la ligne de commande pour que ça fonctionne » puis qu'il se termine.

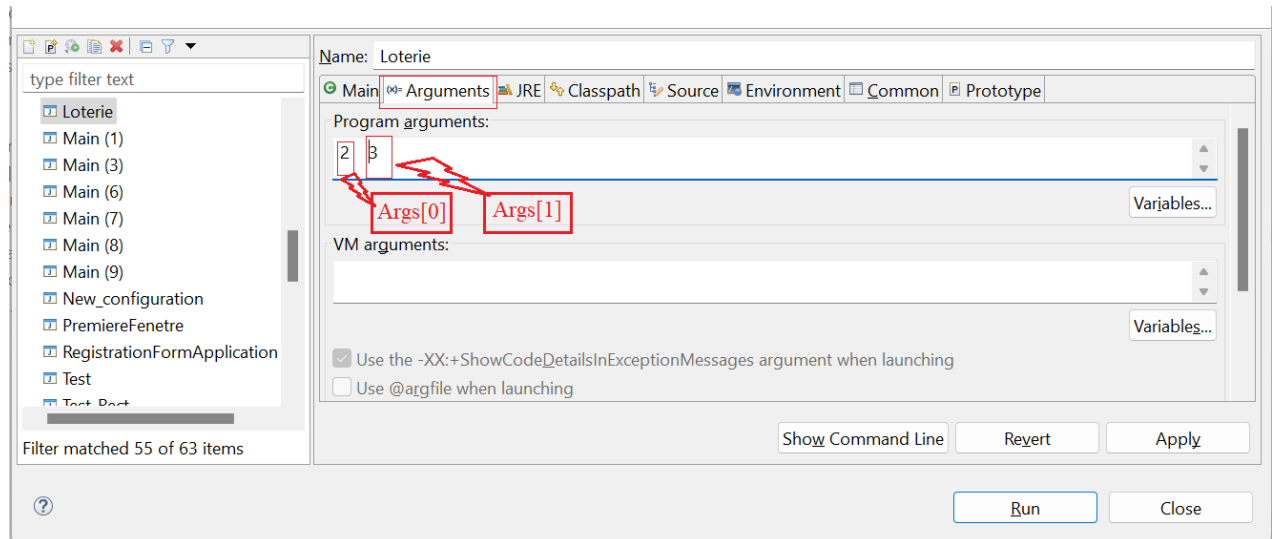
- 2- Si le paramètre indiqué ne représente pas un numérique, il s'agit d'une exception de type **NumberFormatException** ; on souhaite que le programme affiche par exemple : « Les arguments doivent être des nombres » puis qu'il se termine.
- 3- Si le paramètre n est null, on souhaite que le programme affiche par exemple : « impossible de faire une division par zéro ! » puis qu'il se termine.
Il faut pour cela définir sa propre classe d'exception : **ParametreNullException**.
- 4- Si le paramètre x est égale à -3, on souhaite que le programme affiche par exemple : « -3 est négatif : on ne peut pas calculer une racine pour un réel négatif !!!!! » puis qu'il se termine. il faut pour cela définir sa propre classe d'exception :
NegatifException
- 5- Ajoutez dans la fonction main() un bloc finally contenant le message « bloc finally exécuté quel que soit le résultat d'exécution ».

NB :

- Vous pouvez utiliser la méthode **Double.parseDouble(String s)** permettant de convertir une chaîne de type string en double.
- Pour passer des paramètres en ligne de commande sous eclipse, vous allez au sous menu Run et vous choisissez Run Configurations



Dans la fenêtre qui s'affiche allez dans l'onglet argument et écrivez vos arguments séparés par des espaces.



Exercice2 :

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 incorrect
- * ***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 susceptibles de lever un ensemble d'exceptions que vous devez identifier.
 - c. Le **constructeur** de la classe « authentication » fait appel aux méthodes précédemment énoncées pour lire le login et le mot de passe.
- 3) Implémenter la classe « **TestAuthentication** » intégrant la méthode « **main()** ».
- Cette classe aura comme objectif la réalisation d'une authentication. Ce programme tournera en boucle, tant que la création d'une nouvelle authentication est incorrecte (capture d'exception).
- S'il y a un problème de type **java.io.IOException** ou **WrongInputLength**, on doit arrêter l'exécution du programme(en utilisant `System.exit(0)`) afin d'éviter les risques d'attaques.

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