UMP. EST Oujda **AU 2015-16**
2ème année DAI

TPs N° 6 : Les exceptions

Exercice 1: Expliquer les clauses " throws", "throw", "try", "catch()" et "finally"

Exercice 2:
Il existe en java des exceptions prédéfinies telles que (Voir l'index du livre):

Exception
- RunTimeException
  - ArithmeticException
  - ArrayStoreException
  - ClassCastException
  - IllegalArgumentException
    - IllegalThreadStateException
    - NumberFormatException
  - IllegalMonitorStateException
  - IllegalStateException
  - IndexOutOfBoundsException
    - ArrayIndexOutOfBoundsException
    - StringIndexOutOfBoundsException
  - NegativeArraySizeException
  - NullPointerException
  - SecurityException
  - UnsupportedOperationException

-Expliquer quand-est-ce-que chacune de ces exceptions est déclenchée.
-A-t-on besoin de déclarer la clause throws dans des méthodes susceptible de déclencher une de ces méthodes.
- Quelles informations fournissent les méthodes :
getMessage, getCause, printStackTrace, getStackTrace.
-Quelle sont les éléments du tableau : StackTraceElement
-Donner un exemple d'utilisation pour ces méthodes.

Exercices 3 :
Complétez le programme suivant pour que les erreurs susceptibles de se produire soient gérées jusqu'à ce qu'un calcul soit effectivement mené à bout.

```
class Except_1{
static int[] tableau = {17, 12, 15, 38, 29, 157, 89, -22, 0, 5};
static int division(int indice, int diviseur){
return tableau[indice]/diviseur;}
public static void main(String[] args){
int x, y;
scaner r=new scaner(system.in) ;
println("Entrez l'indice de l'entier à diviser: ");
x =r.nextInt();
Printf ("Entrez le diviseur: ");
```

```
y = r.nextInt();
println ("Le résultat de la division est: "+ division(x,y) );}
}
```

Exercice 3.
Il s'agit de faire une saisie de choix pour un menu sécurisé.
Question 1 levée d'exception
Nous allons faire une méthode saisirChoix qui prendra en paramètre un entier n et renverra
une valeur comprise entre 1 et n, tapée au clavier par l'utilisateur. Les différentes erreurs qui pourront se produire seront :
– n est inférieur ou égal à 1
– l'utilisateur a entré un nombre qui n'est pas compris entre 1 et n
– l'utilisateur n'a pas entré un nombre
Chaque erreur devra être détectée par le programme et être signalée par une exception spécifique. Il faut donc créer trois classes différentes d'exception.
Question 2 affichage du menu
Ecrire une méthode qui prend en paramètre un tableau de chaînes de caractères. Chaque chaîne de caractère décrit un choix du menu. La méthode doit afficher ces différents choix.
Question 3 question et réponse
Ecrire une méthode qui utilise les deux méthodes déjà écrites pour afficher un menu et saisir le choix de l'utilisateur. Cette méthode devra gérer les trois exceptions définies à la question 1. Elle prendra en entrée un tableau d'options (string) et en sortie le choix effectué (sous la forme d'un entier).
Question 4 classe
Ecrivez une classe Menu qui possède un constructeur pour initialiser un tableau de chaînes représentant différents choix. Cette classe permettra de réaliser une saisie d'un des choix au moyen d'un entier, en utilisant les exceptions nécessaires.
Cette classe reprendra les fonctionnalités développées aux questions précédentes, mais adaptées à la structure de classe.
Ecrivez un programme qui utilise cette classe Menu avec trois menus différents qui seront trois objets différents. Ce programme capturera les exceptions susceptibles d'être levées.

Exercice 4
Une pile est une queue premier-entré-dernier-servi. Ecrire un programme nommé MyIntStack qui utilise un tableau pour stocker le contenu de la pile d'éléments entiers. Puis ecrire le programme de test :

A stack is a first-in-last-out queue. Write a program calld MyIntStack, which uses an array to store the contents, restricted to int. Puis, Ecrire un programme de test

```
public class MyIntStack {
   private int[] contents;
   private int tos;  // Top of the stack
   public MyIntStack(int capacity) {
   contents = new int[capacity];
 tos = -1;   }
public void push(int element) {
 contents[++tos] = element;   }
public int pop() {
```

```
return contents[tos--];   }
public int peek() {
return contents[tos];   }
public boolean isEmpty() {
return tos < 0;   }
public boolean isFull() {
return tos == contents.length - 1;   }
}
```

Modifier la méthode push() pour lever une IllegalStateException si la pille est vide.
Modifier la méthode push() pour retourner true si l'opération est réussie, ou false sinon.
Modifier la méthode push() pour augmenter la capacité par réallocation d'un autre tableau si la pille est remplie.