Exercices Exception

QCM

Question 1 :

Que va afficher le programme suivant à la console ?

```
public static void main (String[] args)
{
   Try
   {
      System.out.print ("A");
      int value = Integer.parseInt ("8A");
      System.out.print ("B");
   }
   catch (NumberFormatException exception)
   {
      System.out.print ("C");
      return;
   }
   Finally
   {
      System.out.print ("D");
   }
   System.out.print ("E");
}
```

1) ABCDE

2) ACD

3) A0CDE

4) A0BCD

5) ACE

Question2 :

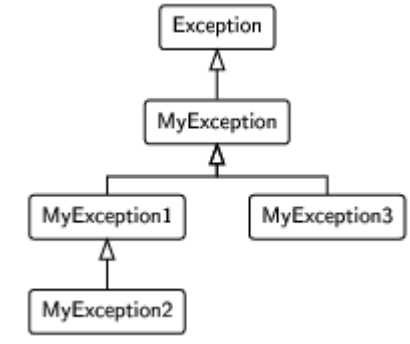
Complétez la classe suivante qui définit une nouvelle exception avec le moins d'instructions possible

```
public class MyException extends Exception
{
   // À compléter
}
```

1) Public MyException () {super () ;}
2) Public MyException () {super (‘’MyException’’) ;}
3) Public MyException (String msg) {super (msg) ;}
4) **On ne peut rien mettre du tout (dans ce cas la classe va posséder d’un constructeur par défaut)**

Question 3 :

Soit la hiérarchie de classes suivante:



Je veux appeler la méthode dont la signature est donnée ci-dessous et gérer les trois erreurs séparément, dans quel ordre dois-je mettre mes clauses `catch` *?*

```
public static void compute () throws MyException1, MyException3, MyException2;
```

1) Dans le même ordre que leur déclaration MyException1, MyException3, MyException2
2) Dans n’importe quel ordre
3) MyException3, MyException2, MyException1
4) MyException2, MyException3, MyException1
5) MyException1, MyException2, MyException3

L'ordre des blocs `catch` est important : il faut placer les classes filles avant leurs classes mères. Dans le cas contraire le compilateur génère une erreur.

Question 4:

Qu’affiche le programme a la console?

```
public static void main (String[] args)
{
   int value = 0;
```

```
   try
   {
      value = 1;
      value = compute (value);
      value = 2;
   }
   catch (ArithmeticException e)
   {
      value = value + 10;
   }
   catch (RuntimeException e)
   {
      value = value + 20;
   }
   System.out.println (value);
}
private static int compute (int value)
{
   return value / (value - 1);
}
```

1) **11**
2) 21
3) 31
4) 2
5) Une erreur d'exécution apparait

Exercice 2 :

Soit la classe Temps suivante :

Class Temps
{
//attributs
Private int heures, minutes, secondes ;
//constructeur
Temps (int h, int m, int s) **throws TimeException**
{
If (h<0 |h>23 | m<0 |m>59 |s<0 |s>59)
throw new TimeException(''temps invalide'') ;
Else

```
Heures=h ;
Minutes=m ;
Secondes=s ;
}
Public static void main (String [] args)
{
Try{
Temps t=new Temps (24, 12, 67) ;
}
Catch (TimeException ex) {
System.out.println(ex.getMessage()) ;
}
}
```

1) Modifier le constructeur de cette classe de manière a ce qu'il lance une exception de type **TempsException** (qu'il ne traitera pas) si les heures, les minutes ou les secondes ne correspondent pas a un temps valide

2) Modifier le code de la méthode main de manière a ce que l'exception TempsException soit traitée en affichant le message suivant : « Temps invalide »