Exceptions

Exercice 1 : Comprendre les exceptions
Dans cet exercice, on considère le programme donné au listing 1 qui compile sans erreur. Les identifiants sont volontairement non significatifs.

Listing 1 – Les exceptions en Java

```
1  class ExempleException {
2      private void m2(String p) {
3          System.out.print("<");
4          if (p == null) {
5              throw new NullPointerException();
6          }
7          if (p.length() == 0) {
8              throw new IllegalArgumentException("Chaîne vide");
9          }
10         System.out.print(p.charAt(0));
11         System.out.print(p.charAt(1));
12         System.out.print(">");
13     }
14
15     public void m1(String p) {
16         System.out.print("[");
17         try {
18             System.out.print("(");
19             m2(p);
20             System.out.print(")");
21         } catch (NullPointerException e) {
22             System.out.print("N");
23         } catch (IllegalArgumentException e) {
24             System.out.print("I");
25         } finally {
26             System.out.print("F");
27         }
28         System.out.print("]");
29     }
30 }
31
32 class ClassePrincipale {
33     public static void main(String[] args) {
34         String argument = (args.length == 0) ? null : args[0];
35         new ExempleException().m1(argument);
36     }
37 }
```

1.1 Indiquer ce qu'affiche l'exécution des commandes suivantes :

```
1 java ClassePrincipale un
2 java ClassePrincipale ""
3 java ClassePrincipale x
```

1.2 L'ordre des **`catch`** est-il important ?

1.3 Les exceptions `IllegalArgumentException` et `NullPointerException` sont-elles contrôlées ? En particulier, on indiquera ce qui fait qu'une exception est contrôlée en Java, qui contrôle et l'intérêt de cette notion.

Exercice 2 : Somme des arguments de la ligne de commande
La classe Somme (listing 2) affiche la somme des nombres réels donnés en argument de la ligne de commande. Par exemple, « `java Somme 10 15.5 4` » affiche `29.5`.

Listing 2 – La classe Somme

```
1  /** Calculer la somme des paramètres de la ligne de commande.  */
2  public class Somme {
3
4      /* Afficher la somme des arguments de la ligne de commande */
5      public static void main(String[] args) {
6          double somme = 0;
7          for (int i = 0; i < args.length; i++) {
8              somme += Double.parseDouble(args[i]);
9          }
10         System.out.println(somme);
11     }
12
13 }
```

2.1 L'exécution de `java Somme 10 x 3` affiche ce qui suit dans le terminal :

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "x"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2043)
    at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
    at java.lang.Double.parseDouble(Double.java:538)
    at Somme.main(Somme.java:8)
```

Expliquer comment interpréter cet affichage pour comprendre ce qu'il s'est passé.

2.2 Partant de la version initiale de la classe Somme, on propose d'en écrire de nouvelles versions qui diffèrent sur la manière de traiter un argument erroné sur la ligne de commande.

2.2.1 Modifier la classe Somme pour que le message « Certaines données sont incorrectes ! » s'affiche quand un argument de la ligne de commande n'est pas un réel. La somme des arguments ne sera donc écrite que si tous les arguments sont des réels. Voici deux exemples d'utilisation :

```
> java Somme 10 x 3
Certaines données sont incorrectes !
> java Somme 10 3
13.0
```

2.2.2 Modifier la classe Somme pour afficher la somme partielle, jusqu'à l'argument erroné. Un message devra signaler si la somme est partielle.

```
> java Somme 10 x 3
Attention, somme partielle
10.0
```

```
> java Somme 10 3
13.0
```

2.2.3 Modifier la classe Somme pour afficher la somme de tous les arguments de la ligne de commande en ignorant ceux qui ne sont pas réels. Les données ignorées doivent être signalées.

```
> java Somme 10 x 3 y
x ignorée : mauvais format !
y ignorée : mauvais format !
13.0
> java Somme 10 3
13.0
```

2.2.4 Modifier la classe Somme pour que, en cas d'argument erroné, elle propage une exception sous contrôle ArgumentIncorrectException. Définir également l'exception ArgumentIncorrect-Exception.