

DEV1 – Laboratoires Java I**TD 8 – Exceptions, Javadoc**

Ce TD introduit la notion *d'exception* qui permet de lancer une alerte lorsqu'une erreur est détectée. Ensuite nous verrons comment *documenter* le code Java à l'aide de l'outil Javadoc.

Les codes sources et les solutions de ce TD se trouvent à l'adresse :

<https://git.esi-bru.be/dev1/labo-java/tree/master/td08-exceptions>

Table des matières

1	Les exceptions	2
2	Gérer les entrées de l'utilisateur	2
3	Documentation des méthodes	3

1 Les exceptions

```
1 package esi.dev1.td7;
2
3 public class Cercle {
4
5     /**
6      * Calcule et retourne le périmètre d'un cercle de rayon donné.
7      *
8      * @param rayon le rayon du cercle
9      * @return le périmètre du cercle de rayon \code{rayon}
10     */
11     static double périmètre(double rayon) {
12         if(rayon <= 0) {
13             throw new IllegalArgumentException("Le rayon doit être positif: "+rayon);
14         }
15         return 2 * Math.PI * rayon;
16     }
17
18     public static void main(String[] args) {
19         System.out.println(périmètre(10));
20         System.out.println(périmètre(-3));
21     }
22 }
```

Cercle.java

La méthode `périmètre` de la classe `Cercle` ci-dessus reçoit en paramètre le rayon du cercle.

Le périmètre d'un cercle n'a de sens que si le rayon est strictement positif. La méthode va *lancer* une `IllegalArgumentException` dans la cas contraire. Lorsque cette erreur est lancée le programme s'arrête et affiche un message d'erreur. Lorsqu'on exécute la classe `Cercle` on obtient le message suivant :

```
> java esi.dev1.td7.Cercle
62.83185307179586
Exception in thread "main" java.lang.IllegalArgumentException: Le rayon
    ↪ doit être positif: -3.0
    at esi.dev1.td7.Cercle.périmètre(Cercle.java:13)
    at esi.dev1.td7.Cercle.main(Cercle.java:20)
```

Exercice 1 Calendrier : méthodes robustes

Dans votre projet calendrier, vérifiez que les paramètres des méthodes sont corrects et lancer une `IllegalArgumentException` avec un message adéquat sinon :

- ▷ `String nomMois(int mois)` : le mois est compris entre 1 et 12;
- ▷ `void afficherTitre(int mois, int année)` : le mois est compris entre 1 et 12;
- ▷ `void afficherMois(int décalage, int nombreJours)` : le décalage est compris entre 0 et 6 et le nombre de jours entre 1 et 31.
- ▷ `int nombreJours(int mois, int année)` : le mois est compris entre 1 et 12;
- ▷ `int numéroJour(int jour, int mois, int année)` : la date est correcte.

Vérifiez qu'une exception est lancée lorsque vous entrez un mois incorrect dans votre application.

2 Gérer les entrées de l'utilisateur

Dans la classe `Saisie` ci-dessous, la méthode `lireEntier` demande à l'utilisateur d'entrer un entier. Tant que l'utilisateur entre autre chose qu'un entier, la méthode lui

demande à nouveau d'en entrer un.

```
1 package esi.dev1.td7;
2 import java.util.Scanner;
3
4 public class Saisie {
5
6     /**
7      * Lecture robuste d'un entier.
8      * Tant que l'entrée de l'utilisateur n'est pas
9      * une entier la méthode demande une nouvelle entrée.
10     *
11     * @param message message à afficher.
12     * @return l'entier saisi par l'utilisateur.
13     */
14     static int saisieEntier(String message) {
15         Scanner clavier = new Scanner(System.in);
16         System.out.println(message);
17         while(!clavier.hasNextInt()) { //si l'entrée saisie n'est pas un entier
18             clavier.next(); // on n'en fait rien, on attend une nouvelle entrée
19             System.out.println("Le nombre saisi n'est pas un entier.");
20             System.out.println(message);
21         }
22         return clavier.nextInt();
23     }
24
25     public static void main(String[] args) {
26         int année = saisieEntier("Entrez votre année de naissance: ");
27         System.out.println("Vous avez " + (2018-année) + " ans");
28     }
29 }
```

Saisie.java

Exercice 2 Lectures robustes

Créez une classe `Lecture` et écrivez-y les méthodes suivantes :

- ▷ `int lireEntier(String message)` : comme dans l'exemple ci-dessus, lit et retourne un entier. Tant que l'utilisateur n'entre pas un entier, la méthode lui demande à nouveau.
- ▷ `double lireDouble(String message)` : lit et retourne un double. Tant que l'utilisateur n'entre pas un double, la méthode lui demande à nouveau.
- ▷ `int lireEntier(String message, int min, int max)` : lit et retourne un entier compris entre `min` et `max`. Tant que l'utilisateur n'entre pas un entier compris entre `min` et `max`, la méthode lui demande à nouveau.

Astuce : faites appel à la méthode `lireEntier`.

Exercice 3 Calendrier avec lectures robustes

Utilisez des méthodes robustes pour gérer les éventuelles erreurs de l'utilisateur dans votre application calendrier.

3 Documentation des méthodes

Dans l'exemple du cercle ci-dessus la documentation de la méthode `périmètre` se trouve juste avant l'entête de la méthode entre les balises `/**` et `*/`. Cette documentation suit le format `Javadoc` vu au cours.

Pour générer la documentation faites un clic-droit sur l'icône du projet et choisissez **Generate Javadoc** dans le menu déroulant qui apparaît. Un navigateur s'ouvre avec

votre Javadoc. Dans la console (output) les éventuels erreurs et avertissements sont affichés.

Exercice 4 Documentation du calendrier

Documentez chaque méthode du projet calendrier et générez sa documentation.