Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2020 – 2021
DEV1 – JAVL

DEV1 – JAVL – Laboratoires Java

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

Version du document
12 septembre 2020

1 Les exceptions

Cercle.java

```java
package esi.dev1.td8;

public class Cercle {

   /**
   * Calcule et retourne le périmètre d'un cercle de rayon donné.
   *
   * @param rayon le rayon du cercle
   * @return le périmètre du cercle de rayon \code{rayon}
   */
   static double périmètre(double rayon) {
      if(rayon <= 0) {
         throw new IllegalArgumentException("Le rayon doit être positif: "+rayon);
      }
      return 2 * Math.PI * rayon;
   }

   public static void main(String[] args) {
      System.out.println(périmètre(10));
      System.out.println(périmètre(-3));
   }
}
```

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.td8.Cercle
62.83185307179586
Exception in thread "main" java.lang.IllegalArgumentException: Le rayon
    ↪ doit être positif: -3.0
    at esi.dev1.td8.Cercle.périmètre(Cercle.java:13)
    at esi.dev1.td8.Cercle.main(Cercle.java:20)
```

Exercice 1 Calendrier : méthodes robustes

Dans votre projet calendrier, vérifiez que les paramètres des méthodes sont corrects et lancez 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, c'est-à-dire que `mois` est compris entre 1 et 12 et `jour` est compris entre 1 et le nombre de jours dans le mois.

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.

Saisie.java

```java
package esi.dev1.td8;
import java.util.Scanner;

public class Saisie {

   /**
    * Lecture robuste d'un entier.
    * Tant que l'entrée de l'utilisateur n'est pas
    * un entier la méthode demande une nouvelle entrée.
    *
    * @param message message à afficher.
    * @return l'entier saisi par l'utilisateur.
   */
   static int saisieEntier(String message) {
      Scanner clavier = new Scanner(System.in);
      System.out.println(message);
      while(!clavier.hasNextInt()) { //si l'entrée saisie n'est pas un entier
         clavier.next(); // on n'en fait rien, on attend une nouvelle entrée
         System.out.println("Le nombre saisi n'est pas un entier.");
         System.out.println(message);
      }
      return clavier.nextInt();
   }

   public static void main(String[] args) {
      int année = saisieEntier("Entrez votre année de naissance: ");
      System.out.println("Vous avez " + (2018-année) + " ans");
   }
}
```

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 le documentation faites un clic-droit sur l'icône du projet et choisissez **Generate Javadoc** dans le menu déroulant qui apparait. Un navigateur s'ouvre avec votre Javadoc. Dans la console (output) les éventuels erreurs et avertissements sont affichés. Les fichiers générés se trouvent dans le répertoire `dist/javadoc`.

Exercice 4 Documentation du calendrier

Documentez chaque méthode du projet calendrier et générez sa documentation.