ENSET

Compte rendu
TP 3

Source code : https://github.com/YassirJr/jeraidi-yassir-java-oop.git

Elaboré par : **Jeraidi Yassir**

Encadrant pédagogique : **Aminou Loubna**

exercice 1: Test de Vitesse du Véhicule

Cet exercice montre comment créer une classe d'exception personnalisée en Java. Le programme simule un test de vitesse de véhicule et lance une exception personnalisée si la vitesse dépasse une certaine limite.

Classe : TropViteException

Ce fichier définit une classe d'exception personnalisée TropViteException qui étend la classe Exception. Cette exception est utilisée pour signaler qu'un véhicule va trop vite.

```
package org.example.ex1;

public class TropViteException extends Exception {
    public TropViteException(int number) {
        super(
                "C'est une exception de type TropViteException. Vitesse en cause : " + numbe
        );
    }
}
```

- Le constructeur prend un entier number comme paramètre, qui représente la vitesse qui a causé l'exception.
- Le constructeur appelle le constructeur de la superclasse (Exception) avec un message d'erreur personnalisé qui inclut la vitesse.

Classe : Vehicle

Ce fichier définit une classe Vehicule qui contient une méthode pour tester la vitesse du véhicule et lance une TropViteException si la vitesse dépasse une certaine limite.

```
package org.example.ex1;

public class Vehicule {

    public void testVitesse(int vitesse) throws TropViteException {
        if (vitesse > 90) {
            throw new TropViteException(vitesse);
        }
        System.out.println("La vitesse est de " + vitesse + " km/h");
    }

    public static void main(String[] args) {
        Vehicule vehicule = new Vehicule();
        try {
            vehicule.testVitesse(80);
```

```
            vehicule.testVitesse(100);
        } catch (TropViteException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- La méthode testVitesse prend un entier vitesse comme paramètre et vérifie s'il dépasse 90.
- Si la vitesse est supérieure à 90, elle lance une TropViteException avec la vitesse comme argument.
- Si la vitesse est dans la limite, elle affiche la vitesse.
- La méthode main crée une instance de Vehicule et teste deux vitesses : 80 et 100.
- Si une TropViteException est lancée, elle attrape l'exception et affiche le message d'erreur.

exercice 2: Calculateur de Racine Carrée

Cet exercice montre comment créer une classe d'exception personnalisée en Java. Le programme simule le calcul de la racine carrée d'un nombre et lance une exception personnalisée si le nombre est négatif.

Classe : RacineCarreeException

Ce fichier définit une classe d'exception personnalisée RacineCarreeException qui étend la classe Exception. Cette exception est utilisée pour signaler qu'un nombre est négatif et que sa racine carrée ne peut pas être calculée.

```
package org.example.ex2;

public class RacineCarreeException extends Exception {
    public RacineCarreeException(int number) {
        super(
                "C'est une exception de type RacineCarreeException. Nombre négatif : " + nu
        );
    }
}
```

- Le constructeur prend un entier number comme paramètre, qui représente le nombre négatif.
- Le constructeur appelle le constructeur de la superclasse (Exception) avec un message d'erreur personnalisé qui inclut le nombre négatif.

Classe : Calculateur

Ce fichier définit une classe Calculateur qui contient une méthode pour calculer la racine carrée d'un nombre et lance une RacineCarreeException si le nombre est négatif.

```
package org.example.ex2;

public class Calculateur {
    public void testRacineCarree(int number) throws RacineCarreeException {
        if(number < 0){
            throw new RacineCarreeException(number);
        }
        System.out.println("La racine carrée de " + number + " est " + Math.sqrt(number));
    }

    public static void main(String[] args) {
        Calculateur calculateur = new Calculateur();
        try {
            calculateur.testRacineCarree(25);
            calculateur.testRacineCarree(-1);
        } catch (RacineCarreeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- La méthode testRacineCarree prend un entier number comme paramètre et vérifie s'il est négatif.
- Si le nombre est négatif, elle lance une RacineCarreeException avec le nombre comme argument.
- Si le nombre est positif, elle affiche la racine carrée du nombre.
- La méthode main crée une instance de Calculateur et teste deux nombres : 25 et -1.
- Si une RacineCarreeException est lancée, elle attrape l'exception et affiche le message d'erreur.

exercice 3: Évaluateur de Notes

Cet exercice montre comment créer une classe d'exception personnalisée en Java. Le programme simule l'évaluation des notes et lance une exception personnalisée si la note est en dehors de la plage valide (0 à 20).

Classe : NoteInvalideException

Ce fichier définit une classe d'exception personnalisée NoteInvalideException qui étend la classe Exception. Cette exception est utilisée pour signaler qu'une

note est invalide.

```
package org.example.ex3;

public class NoteInvalideException extends Exception {
    public NoteInvalideException(int note) {
        super(
                "Exception de type NoteInvalideException. Note invalide : " + note
        );
    }
}
```

- Le constructeur prend un entier note comme paramètre, qui représente la note invalide.
- Le constructeur appelle le constructeur de la superclasse (Exception) avec un message d'erreur personnalisé qui inclut la note invalide.

Classe : Evaluateur

Ce fichier définit une classe Evaluateur qui contient une méthode pour vérifier la validité d'une note et lance une NoteInvalideException si la note est en dehors de la plage valide.

```
package org.example.ex3;

public class Evaluateur {
    public void verifierNote(int note) throws NoteInvalideException {
        if(note < 0 || note > 20){
            throw new NoteInvalideException(note);
        }
        System.out.println("La note est de " + note);
    }

    public static void main(String[] args) {
        Evaluateur evaluateur = new Evaluateur();
        try {
            evaluateur.verifierNote(15);
            evaluateur.verifierNote(25);
        } catch (NoteInvalideException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- La méthode verifierNote prend un entier note comme paramètre et vérifie s'il est compris entre 0 et 20.
- Si la note est en dehors de cette plage, elle lance une NoteInvalideException avec la note comme argument.

- Si la note est dans la plage valide, elle affiche la note.
- La méthode main crée une instance de Evaluateur et teste deux notes : 15 et 25.
- Si une NoteInvalideException est lancée, elle attrape l'exception et affiche le message d'erreur.