P.O.O (JAVA)

TD N°5

Objectif : Gestion des erreurs

Séance : 6éme semaine

Exercice 1 : (Intégrité des données)

- Considérons le code suivant qui ne fait aucune vérification sur l'intégrité des données servant à la création d'un objet de type Project.
- Modifiez-le de sorte à :
  a) Ce que ne soient créés que des projets dont le nom et le sujet n'excède pas les 50 caractères ;
  b) Garantir que la durée du projet soit bien un entier positif.
- Les noms, sujet et durée seront redemandés à l'utilisateur tant qu'ils sont introduits de façon incorrecte.Vous introduirez pour cela deux classes d'exceptions personnalisées *WrongDurationException* et *NameTooLongException*
- Une *String*, *strNumber*, correspondant à un entier peut être transformée en *int* par l'appel à la méthode statique *parseInt* de la classe *Integer*(***Integer.parseInt(strNumber)***).
- Si *strNumber* ne corresponds pas à un *int* une *RunTimeException* de type *NumberFormatException* sera lancée.

```
import java.util.Scanner;
import java.util.ArrayList;
class SafeProject {
   private+9 final static int NB_PROJECTS = 3;
   public static void main(String[] args) {
         ArrayList<Project> projects = new ArrayList<Project>();
      do {  Project project = new Project();
               project.readProject();
               projects.add(project);
         } while (projects.size() < NB_PROJECTS);
   }              }
```

```
class Project {
   private String name = null;
   private String subject = null;
   private int duration = -1;
   public Project() {}
   public void readProject() {
         Scanner scanner = new Scanner(System.in);
         System.out.println("Donnez le nom du projet : ");
         name = scanner.nextLine();
         System.out.println("Donnez le sujet du projet : ");
         subject = scanner.nextLine();
         System.out.println("Donnez la durée du projet : ");
         duration = scanner.nextInt(); }
 }
```

Exercice 2 : (Se compile … ou pas)

- Pas de programmation pour cet exercice ! Répondez simplement aux questions posées dans le code fourni ci-dessous. Quelques incursions dans la documentation de l'API de Java vont être nécessaires pour répondre aux questions.

```
class Exemple {
```
/*Expliquer pourquoi ce code ne se compile pas */
```
public void m1() {foo();}
public void foo() throws Exception { throw new Exception();      }
```
/*Expliquer pourquoi ce code n'est pas considéré comme bon*/
```
public void m2() {
      try { //do stuff...
      } catch (Exception e) {}
}
```
/*Expliquer pourquoi ce code ne se compile pas */
```
public void m3() {
      try { //do stuff...
      } catch (Exception e) {}
        catch (NullPointerException e) {} }
```
/*Expliquer pourquoi ce code ne se compile pas */
```
public void m4() {throw new CustomCheckedException();}
private class CustomCheckedException extends Exception {
private static final long serialVersionUID = -7944813576443065516L;
public CustomCheckedException() {//nothing}
}
```
/*Expliquer pourquoi ce code ne se compile pas */
```
public int m5() {
      int age;
      String s = "24";
      try {
            age = getAccessCode();
      } catch (IllegalAccessException e) {e.printStackTrace();}
      return age;
}
public int getAccessCode() throws IllegalAccessException {
      throw new IllegalAccessException();
}
```
/*Expliquer pourquoi ce code se compile */
```
public void m6() { bar(); }
public int bar() {throw new RuntimeException();}
}
```

Exercice 3 :

Ecrire une classe nommée «**CalcLigneCommande**» ne contenant que la méthode « **main()** », qui:

- Calcule et affiche :
  - la somme de tous les entiers donnés comme arguments de la ligne de commande.
  - la somme de tous les réels purs (un entier n'est pas considéré come un réel) donnés comme arguments de la ligne de commande.
- Affiche les autres arguments qui ne sont pas des entiers et des réels.

Exemple d'exécution :

```
> java CalcLigneCommande  Semestre  S  5  notes   14   18.5    SMI   12.3   8F
La somme des arguments entiers:  19       //  5+14
La somme des arguments réels:  38.8      //  18.5 + 12.3 + 8F
Les arguments : Semestre, S, notes, SMI,  ne sont ni des entiers ni des réels.
```

Indication : Une chaine de caractères qui ne peut pas être convertie en un entier ou en un réel, lance l'exception «**NumberFormatException**».

Exercice 4 : (Déménagement : révision)

Vous déménagez bientôt et commencez à ranger vos affaires dans des cartons. Pour retrouver rapidement vos objets et savoir dans quel carton vous avez rangé tel ou tel objet, vous décidez d'écrire un programme orienté objet gardant trace de vos rangements.
Un carton porte un numéro. Il est rempli d'objets et/ou d'autres cartons remplis de la même façon. Le contenu d'un carton est l'ensemble des objets qu'il contient (y compris ceux contenus dans des cartons plus petits).
Vous considérerez :

- que chaque objet à un nom;
- qu'un déménagement est un *ensemble* de cartons.

Les fonctionnalités dont vous souhaitez disposer sont les suivantes :

1. mettre quelque chose (un objet ou un autre carton) dans un carton;
2. afficher le contenu d'un carton donné (le nom de tous ses objets);
3. ajouter un carton au déménagement;
4. afficher le contenu de tous les cartons du déménagement.
5. trouver le numéro du carton où se trouve un objet de nom donné : en ne retournant que le numéro du carton le plus externe (et un nombre négatif par exemple si l'objet ne se trouve pas dans les cartons);

Libre à vous maintenant d'implémenter une solution correcte (qui devra toutefois être compatible avec le programme principal fourni ci-dessous). Veillez toutefois à ne pas avoir de duplication dans votre code et à respecter une bonne encapsulation. Pour commencer, vous pouvez dessiner le schéma de la hiérarchie de classes que vous imaginez, puis regroupez les méthodes/attributs communs dans les classes mères. Puis enfin lorsque la hiérarchie sera claire pour vous, vous pouvez coder le programme.

Voici le programme principal fourni :

```
class Demenagement
{     public static void main(String[] args) {
//On cree un demenagement qui pourra contenir 2 cartons principaux
        Demenagement demenagement = new Demenagement(2);
// On cree et remplis ensuite 3 cartons. Arguments du constructeur de Box
(Carton) : argument 1 : nombre d'items (objets simple ou carton) que le
carton peut contenir
//          argument 2 : numero du carton le carton 1 contient des ciseaux
        Box box1 = new Box(1,1);
        box1.addItem(new SimpleItem("ciseaux"));
        // le carton 2 contient un livre
        Box box2 = new Box(1,2);
        box2.addItem(new SimpleItem("livre"));
        // le carton 3 contient une boussole
        // et un carton contenant une echarpe
        Box box3 = new Box(2,3);
        box3.addItem(new SimpleItem("boussole"));
        Box box4 = new Box(1,4);
```

```
        box4.addItem(new SimpleItem("echarpe"));
        box3.addItem(box4);
        //On ajoute les trois cartons au premier carton du demenagement
        Box box5 = new Box(3,5);
        box5.addItem(box1);
        box5.addItem(box2);
        box5.addItem(box3);
        //On ajoute un carton contenant 3 objets au demenagement
        Box box6 = new Box(3,6);
        box6.addItem(new SimpleItem("crayons"));
        box6.addItem(new SimpleItem("stylos"));
        box6.addItem(new SimpleItem("gomme"));
        //On ajoute les deux cartons les plus externes au demenagement
        demenagement.addBox(box5);
        demenagement.addBox(box6);
        //On imprime tout le contenu du demenagement
        demenagement.print();
//On imprime le numéro du carton le plus externe contenant l'objet "echarpe"
System.out.println("L'écharpe est dans le carton numero " +
demenagement.find("echarpe"));    }
```

- Il devrait produire un affichage ressemblant à ce qui suit :

```
Les objets de mon déménagement sont :
ciseaux
livre
boussole
écharpe
crayons
stylos
gomme
L'écharpe est dans le carton numéro 5
```

Bonne Chance