UNIVERSITÉ TOULOUSE CAPITOLE

FACULTÉ D'INFORMATIQUE

Travaux pratiques n°8

Programmation Objet
Interface

Consignes

- Lire et apprendre le support sur Héritage, classe abstraite et interface.

Objectifs

- Comprendre le concept d'interface.
- Savoir implémenter la relation de contrat d'une interface.

Travail à faire

Toutes les classes doivent être réalisées, codées proprement, commentées et correctement testées dans le programme principal.

Exercice : Les formes géométriques (Schéma de classes fait en cours)

A partir d'un ensemble de formes géométriques, vous devez écrire un programme pour :

- retrouver la forme géométrique ayant le plus grand périmètre et
- faire la somme des surfaces de l'ensemble des formes géométriques.

1) Effectuer le schéma des classes représentant les formes géométriques suivantes, les propriétés sont entre parenthèses : le cercle (rayon), le rectangle (longueur et largeur), le carré, le triangle (base et hauteur) et le triangle équilatéral.

2) Implémenter les classes et tester-les.

Conseil : Considérer que le type forme géométrique est un contrat.

Exercice : Tri sur les véhicules (Exercice similaire à l'exemple traité en cours sur les personnes)

3) A partir de l'exercice sur le ferry du TP 7, ajouter la capacité aux véhicules d'être comparés en fonction de leur charge totale. Pour cela, la classe `Vehicule` doit implémenter l'interface `Comparable<T>` et définir sa seule méthode `int compareTo(T)`. Attention de bien respecter les spécifications sur les valeurs de retour ! Ensuite dans votre programme principal, il vous suffit d'utiliser la méthode `Arrays.sort(Object[])` pour trier votre tableau de véhicules.

4) L'implémentation, comme ci-dessus, de l'interface `Comparable<T>` fournit une seule façon de comparer des objets de la classe `Vehicule`. Nous souhaiterions également les trier en fonction de leur hauteur.
Pour cela, créer une classe qui implémente l'interface `Comparator<T>` et définir sa seule méthode `int compare(T,T)`. Ensuite dans votre programme principal, créer une instance de votre comparateur et utiliser la méthode `Arrays.sort(T[], Comparator<T>)` pour trier votre tableau de véhicules.

Exercice : ToolsArrays (Correction donnée en cours)

5) Répondre à la question 6) du TP 4 et tester-la sur votre tableau de véhicules.
   Penser à contraindre / borner votre paramètre de type !

Exercice : Map - Filter - Reduce

Nous allons pas à pas implémenter les différentes étapes d'un célèbre modèle de programmation très utilisé dans le monde du Big Data : le MapReduce.
Cet exercice nous amène progressivement vers un nouveau paradigme de programmation : la **programmation fonctionnelle**.

Le code sur la page suivante vous montre comment :
- créer un filtre générique,
- créer une implémentation de ce filtre générique et
- utiliser ce filtre pour faire un traitement sur des données.

6) Copier / copier ce code et tester-le.

7) Implémenter un filtre sur des `String` et tester-le.

8) Implémenter un filtre sur des `Vehicule` et tester-le.

9) En vous inspirant de l'interface `Filter`, créer une interface `Map<U,T>` et les "outils" associés qui vont transformer les données de type `U` en données de type `T`.
   Par exemple, on pourra transformer :
   - une liste de chaines de caractères en une liste de leur longueur : ("abc", "d", "efgh") devient (3, 1, 4).
   - une liste de véhicules en une liste de leur charge totale respective.

10) Il ne manque plus qu'à faire le `Reduce` !

```java
/**
 * Interface pour définir un filtre générique.
 */
public interface Filter<T>
{
   public boolean matches (T obj);
}
```

```java
/**
 * Filtre les entiers > 5.
 */
public class FilterIntGt5 implements Filter<Integer>
{
   @Override
   public boolean matches (Integer obj)
      {
      return obj > 5;
      }
}
```

```java
/**
 * Programme pour tester les filtres.
 */
public class AppFilter
{
   /*----- Méthode générique pour filtrer une liste de données -----*/
   public static <T> List<T> filterList (List<T> elts, Filter<T> filtre)
      {
      List<T> result = new ArrayList<>();

      for (T e : elts)
         if (filtre.matches(e))
            result.add(e);

      return result;
      }

   /*----- Programme principal -----*/
   public static void main (String[] args)
      {
      List<Integer> nbs = List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

      System.out.println(AppFilter.filterList(nbs, new FilterIntGt5()));
      }
}
```