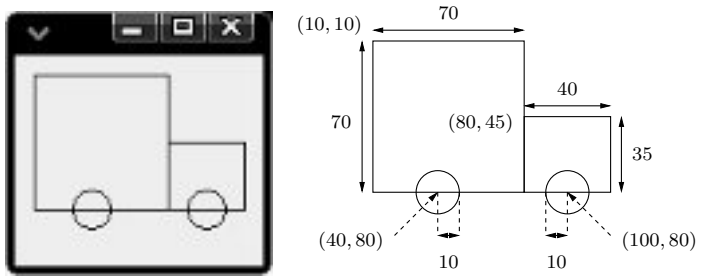
PCOO – TD/TP 4

Vous devez terminer l'implémentation des programmes en dehors des TP.

Exercice 1 : Camion (en TD uniquement)

Nous avons réalisé une classe qui dessine le camion suivant :



Le code de cette classe est donné ci-dessous :

```
public class Truck extends Canvas {
    public Truck() {
        super(130, 110);
        GraphicsContext graphicsContext = this.getGraphicsContext2D();
        FrenchPainter painter = new FrenchPainter(graphicsContext);
        draw(painter);
    }

    private void draw(FrenchPainter painter) {
        painter.drawRectangle(10.0, 10.0, 70.0, 70.0);
        painter.drawRectangle(80.0, 45.0, 40.0, 35.0);
        painter.drawCircle(40.0, 80.0, 10.0);
        painter.drawCircle(100.0, 80.0, 10.0);

    }
}
```

1. Écrivez la classe `FrenchPainter` de façon à faire fonctionner le code précédent en utilisant les méthodes de la classe `GraphicsContext`.
2. `FrenchPainter` doit maintenant implémenter l'interface `Painter`. Remplacer `FrenchPainter` par `Painter` où vous pouvez dans `Truck`.
3. Un anglais a développé une classe `EnglishPainter` qui permet de dessiner des cercles et des rectangles. Attention, nous ne disposons pas du code source de cette classe donc nous ne pouvons pas la modifier. Elle possède une méthode `void drawRectangle(Point2D p1, Point2D p2)` (où `p1` et `p2` sont deux coins opposés du rectangle) et une méthode `void drawCircle(Point2D center, Point2D point)` (où `center` et `point` sont respectivement le centre et un point du cercle). Elle possède également un constructeur qui prend en paramètre une instance de la classe `GraphicsContext`.

   Nous souhaitons utiliser la classe `EnglishPainter` à la place de notre classe `FrenchPainter` en modifiant le moins possible le code déjà écrit. Nous voulons également pouvoir passer d'une version à l'autre en changeant uniquement la ligne `Painter painter= new FrenchPainter(graphicsContext)` par `Painter painter = new EnglishPainterAdapter(graphicsContext)` dans la classe `Truck`.

   Proposez une implémentation de la classe `EnglishPainterAdapter` qui ne demande aucune modification du code existant.

   `Rappel` : l'instruction `new Point2D(x,y)` permet d'instancier le point de coordonnées (x, y).

Exercice 2 : Filtres (en TD et en TP)

Supposons que nous ayons la classe suivante :

```
class BadFilter {
  public enum Type { ODD, LEQ }
  private Type type;
  private int value;

  public BadFilter(Type type) { this.type = type; }

  public BadFilter(Type type, int value) {
    this.type = type;
    this.value = value;
  }

  public List<Integer> apply(List<Integer> list) {
    List<Integer> result = new ArrayList<Integer>();
    switch (type) {
    case ODD: for (int i : list) if (i%2==1) result.add(i);
              break;
    case LEQ: for (int i : list) if (i<=value) result.add(i);
              break;
    }
    return result;
  }
}
```

1. Qu'affiche le code suivant ?

```
List<Integer> list = new ArrayList<Integer>();
for (int i = 0; i < 10; i++) list.add(i);
BadFilter filterOdd = new BadFilter(BadFilter.Type.ODD);
BadFilter filterLeq = new BadFilter(BadFilter.Type.LEQ, 6);
List<Integer> result =
   filterLeq.apply(filterOdd.apply(list));
for (int i : result)
  System.out.println(i);
```

2. Que doit-on modifier dans le code de la classe `BadFilter` pour ajouter un filtre qui sélectionne tous les entiers supérieurs à une valeur ? Est-ce satisfaisant ?

3. Écrivez le code d'une interface `Predicate` contenant l'unique méthode `boolean test(int i)`.

4. En supposant que les classes `Odd` et `Leq` implémentent l'interface `Predicate`, donnez le diagramme de classes d'une nouvelle organisation du code qui corrige le défaut de conception évoqué aux questions 2 et 3. Vous devez faire en sorte que le code suivant ait le même comportement que celui donné en début d'exercice :

```
List<Integer> list = new ArrayList<Integer>();
for (int i = 0; i < 10; i++) list.add(i);
Filter filterOdd = new Filter(new Odd());
Filter filterLeq = new Filter(new Leq(6));
List<Integer> result = filterLeq.apply(filterOdd.apply(list));
for (int i : result) System.out.println(i);
```

5. Implémentez les classes du diagramme de la question précédente.

6. Combien de fois la liste est-elle parcourue par la méthode `apply` de la classe dans le code donné à la question 5 ? Proposez une implémentation de la classe `And` de sorte que le code suivant ait le même comportement que les codes des questions précédentes en ne parcourant qu'une seule fois la liste. Vous ne devez pas modifier les classes précédemment écrites.

```
List<Integer> list = new ArrayList<Integer>();
for (int i = 0; i < 10; i++) list.add(i);
Filter filter = new Filter(new And(new Odd(), new Leq(6)));
List<Integer> result = filter.apply(list);
for (int i : result) System.out.println(i);
```

7. Que doit-on modifier pour ne plus avoir à instancier la classe Filter et donc pouvoir écrire directement le code suivant ?

```
List<Integer> list = new ArrayList<Integer>();
for (int i = 0; i < 10; i++) list.add(i);
Predicate predicate = new And(new Odd(), new Leq(6));
List<Integer> result = Filter.filter(predicate, list);
for (int i : result) System.out.println(i);
```

Exercice 3 : Listes (en TP uniquement)

Nous avons écrit une classe permettant d'afficher une liste de chaînes de caractères en utilisant plusieurs formats différents :

```
public class ItemList {
  private List<String> items;

  public ItemList() { items = new ArrayList<String>(); }

  public void add(String item) { items.add(item); }

  public void printHTML() {
    System.out.println("<ul>");
    for (String i : items) System.out.println("<li>"+i+"</li>");
    System.out.println("</ul>");
  }

  public void printLaTeX() {
    System.out.println("\\begin{itemize}");
    for (String i : items) System.out.println("\\item "+i);
    System.out.println("\\end{itemize}");
  }
}
```

1. Que génèrent les lignes suivantes sur la sortie standard ?

```
ItemList list = new ItemList();
list.add("Maison"); list.add("Immeuble"); list.add("Hutte");
list.printHTML();
list.printLaTeX();
```

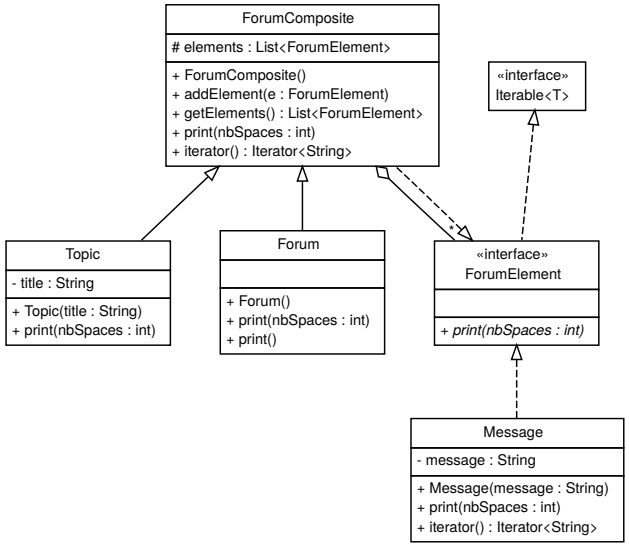
2. Que doit-on modifier dans la classe `ItemList` pour ajouter un nouveau format d'affichage ? Pourquoi n'est-ce pas satisfaisant ?
3. Afin de résoudre ce problème de conception, nous souhaitons factoriser les deux méthodes `printHTML` et `printLaTeX` de façon à n'avoir plus qu'une seule méthode `void print(ListFormat format)` dans `ItemList`.

   Écrivez le code `Java` de l'interface `ListFormat` en choisissant correctement le nom des méthodes qui la compose. L'interface `ListFormat` doit contenir quatre méthodes censées retourner quatre chaînes de caractères qui indiquent respectivement :
   — le début de la liste (`<ul>` ou `\begin{itemize}`) ;
   — la fin de la liste (`</ul>` ou `\end{itemize}`) ;
   — le début d'un élément de la liste (`<li>` ou `\item`) ;
   — la fin d'un élément de la liste (`</li>` ou la chaîne vide) ;
4. Proposez une implémentation de la méthode `void print(ListFormat format)` de la classe `ItemList`.
5. Écrivez le code des deux classes `HTMLListFormat` et `LaTeXListFormat` de sorte que le code suivant compile et produise le même résultat que le code de la question 1 :

```
ItemList list = new ItemList();
list.add("Maison"); list.add("Immeuble"); list.add("Hutte");
list.print(new HTMLListFormat());
list.print(new LaTeXListFormat());
```

Exercice 4 : Forum (en TP uniquement)

Nous souhaitons réaliser un forum. Notre forum est composé de message (classe `Message`) qui peuvent être regroupés en topics (classe `Topic`). Les topics peuvent contenir d'autres topics (en plus des messages). Tous les éléments du forum implémentent l'interface `ForumElement`. Nous obtenons l'organisation suivante (qui respecte le patron de conception `Composite`) :



1. Implémentez le diagramme de classe précédent. La méthode `print(int nbSpaces)` affiche sur la console l'élément avec un décalage sur la gauche de `nbSpaces` espaces. Le code suivant...

```
Forum forum = new Forum();
Topic topic1 = new Topic("Premier topic");
topic1.addElement(new Message("Message 1"));
topic1.addElement(new Message("Message 2"));
forum.addElement(topic1);
Topic topic2 = new Topic("Deuxième topic");
topic2.addElement(new Message("Message 3"));
forum.addElement(topic2);
forum.print();
```

...doit générer la sortie suivante...

```
Forum :
  Topic : Premier topic
    Message 1
    Message 2
  Topic : Deuxième topic
    Message 3
```

2. Nous souhaitons pouvoir parcourir l'ensemble des messages du forum à l'aide d'un itérateur. Pour ce faire, modifiez l'interface `ForumElement` de sorte qu'elle étende l'interface `Iterable<String>`. Ensuite, modifiez en conséquence les classes `Message` et `ForumComposite`.