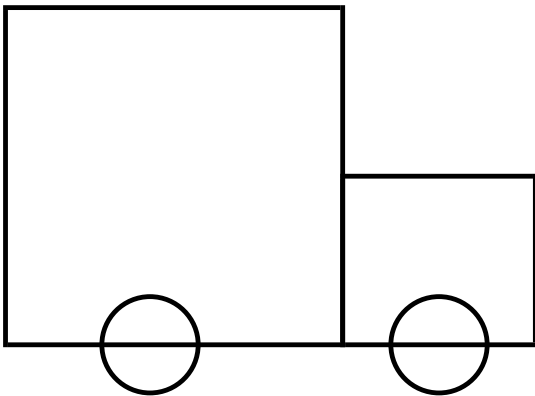


1 Dessin de camion

On considère la classe suivante :

```
public class MyTruck {  
    public static String generateSVG() {  
        SVGGraphics2D drawer = new SVGGraphics2D(130, 110);  
        drawer.drawRect(10, 10, 70, 70);  
        drawer.drawRect(80, 45, 40, 35);  
        drawer.drawOval(30, 70, 20, 20);  
        drawer.drawOval(90, 70, 20, 20);  
        return drawer.getSVGDocument();  
    }  
    public static void drawWithJavaFX(Canvas canvas) {  
        canvas.setWidth(130);  
        canvas.setHeight(110);  
        GraphicsContext context = canvas.getGraphicsContext2D();  
        context.strokeRect(10, 10, 70, 70);  
        context.strokeRect(80, 45, 40, 35);  
        context.strokeOval(30, 70, 20, 20);  
        context.strokeOval(90, 70, 20, 20);  
    }  
}
```

La méthode `generateSVG` permet de générer une chaîne de caractères représentant une image de camion dans le format SVG (format pour des images vectorielles). La méthode `drawWithJavaFX` permet d'afficher cette même image dans un *canvas JavaFX*. Notez que les classes `Canvas` et `GraphicsContext` font partie de *JavaFX* et que la classe `SVGGraphics2D` fait partie d'une librairie qui permet de générer des documents SVG. Le dessin obtenu est le suivant :



Question 1.1 : Que doit-on ajouter dans le code de la classe donnée en début d'exercice pour ajouter une nouvelle façon d'écrire ou d'afficher notre image. Est-ce satisfaisant ? Justifiez votre réponse.

Nous souhaitons refactorer le code de la classe `MyTruck` afin de corriger le problème de conception évoqué à la question précédente. Nous voulons pouvoir écrire le code suivant :

```
public class MyTruck {
    private static void draw(MyDrawer drawer) {
        drawer.setSize(130, 110);
        drawer.drawRectangle(10, 10, 70, 70);
        drawer.drawRectangle(80, 45, 40, 35);
        drawer.drawOval(30, 70, 20, 20);
        drawer.drawOval(90, 70, 20, 20);
    }
    public static String generateSVG() {
        MySVGDrawer drawer = new MySVGDrawer();
        draw(drawer);
        return drawer.getSVG();
    }
    public static void drawWithJavaFX(Canvas canvas) {
        MyJavaFXDrawer drawer = new MyJavaFXDrawer(canvas);
        draw(drawer);
    }
}
```

Question 1.2 : Donnez le code de l'interface `MyDrawer`.

Question 1.3 : Proposez un diagramme de classes qui soit compatible avec le nouveau code de la classe `MyTruck`.

Question 1.4 : Proposez une implémentation des classes `MySVGDrawer` et `MyJavaFXDrawer`.

1.1 Gestion de panier

Le code suivant permet de modéliser des listes d'achats :

— une classe pour les livres :

```
public class Book {
    private final double price, numberOfpages;
    private String name;
    public double price() { return price; }
    public String name() { return name; }
    public Double numberOfpages(){ return numberOfpages; }
}
```

— une interface pour les articles :

```
public interface Item{
    public String name();
    public double price();
    public double expiryDate();
}
```

— une classe pour les articles d'alimentation :

```
public class Food implements Item{
    private final String name;
    private final double price, expiryDate;
    public double price() { return price; }
    public String name() { return name; }
    public Date expiryDate() {return expiryDate; }
}
```

— une classe correspondant à un panier :

```
public class ShoppingCart {
    List<Item> items = new ArrayList<>();
    public List<Item> items(){return items;}
    public void addItem(Item item){items.add(item);}
    public String description(){
        String string="";
        for(Item item: items)
            string += item.name()+" "+item.price()+"€\n";
        return string;
    }
    public void print(){
        System.out.println(this.description());
    }
}
```

Question 2.1 : Nous voulons que Book puisse être mis dans le ShoppingCart. Comment résoudre le problème ? Quel principe SOLID était violé ? Justifiez votre réponse.

Question 2.2 : On voudrait afficher le nombre de pages des livres dans la méthode print() (pour obtenir un affichage La vie devant soi (8€, 240 pages), que devrait-on changer dans le code pour y parvenir sans ajouter de classe ?

Question 2.3 : Est-ce que le nouveau code sans ajouter de classe respecte les principes SOLID ? Justifiez votre réponse.

Question 2.4 : Utilisez le patron visiteur pour implémenter la méthode print() qui affichera aussi le nombre de pages quand il y a un Book dans le ShoppingCart.

```

Book book = new Book();
book.name = "La vie devant soi";
book.price = 8;
book.numberOfpages = 240;
Food pizza = new Food();
pizza.name = "frozen pizza";
pizza.price = 6;
ShoppingCart cart = new ShoppingCart();
cart.add(book);
cart.add(pizza);
// initialisation du visiteur visitor
String string = visitor.print(cart);
System.out.println(string);

```

L'exécution du code ci-dessus devra donner l'affichage :

```

La vie devant soi (8€, 240 pages)
frozen pizza (6€)

```

Vous commencerez par donner le diagramme de classe. Ensuite, vous implémenterez les nouvelles classes utilisées et vous ne mentionnerez que les changements dans les classes existantes.

On voudrait pouvoir afficher Joyeux Noël! (Version 1) ou Joyeux Noël! suivi de Nous pensons à vous chaleureusement. (Version 2) à la fin du ticket. On voudrait aussi avoir la possibilité d'afficher le prix total de la commande (Version 3). Sur le panier de l'exemple ci-dessus, les affichages devraient être les suivants :

print Version 1	print Version 2	print Version 3
La vie devant soi (8€, 240 pages)	La vie devant soi (8€, 240 pages)	La vie devant soi (8€, 240 pages)
frozen pizza (6€)	frozen pizza (6€)	frozen pizza (6€)
Joyeux Noël!	Joyeux Noël!	Total 14€
	Nous pensons à vous chaleureusement.	

On voudrait aussi potentiellement que les impressions de ticket soit en plus sauvegardées dans un fichier log pour garder une trace, c'est-à-dire que le retour de print soit de plus écrit dans un fichier donné.

Question 2.5 :

Quelles sont d'après vous les difficultés générées par l'ajout de telles fonctionnalités, et à quel(s) principe(s) SOLID les associez-vous ? Justifiez votre réponse.

Question 2.6 : Utilisez le patron *decorator* pour implémenter ces 4 fonctionnalités. Vous commencerez par donner le diagramme de classe. Ensuite, vous implémenterez les nouvelles classes utilisées et vous ne mentionnerez que les changements dans les classes existantes. Vous pouvez utiliser `new FilePrinter("log").println("toto")` pour écrire `toto` dans le fichier `log`.