Domaine Sciences et Technologies

Licence informatique : parcours MI

Aix*Marseille université
Socialement engagée

Année 2023-24

Approfondissements en informatique (COO) : TP 3

Code UE : SIN6U28T

TP 3 : Adapter

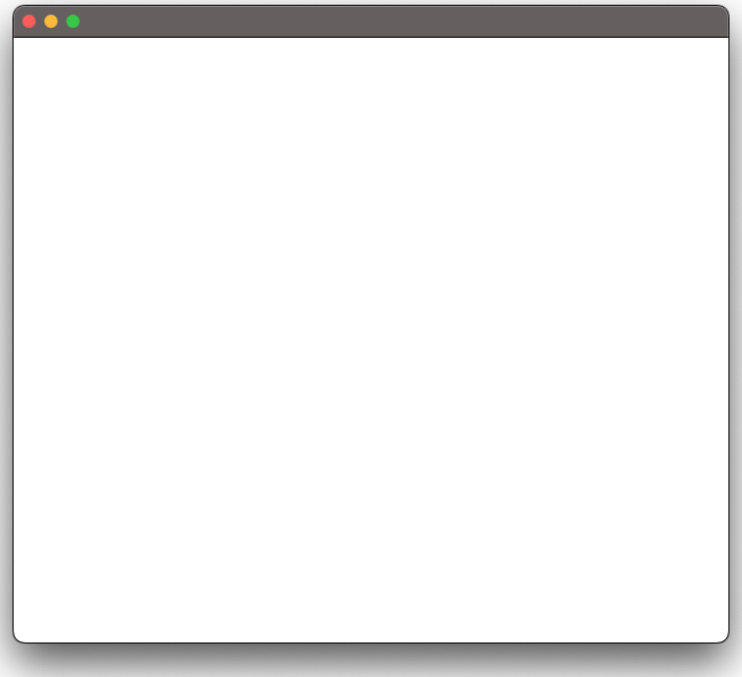
Dessin de camion

Consignes pour démarrer le TP

Comme pour les TP précédents, on va utiliser git pour la gestion de versions et vous allez devoir cloner un dépôt pour ce projet. Il vous faut donc vous reporter aux consignes du premier TP.

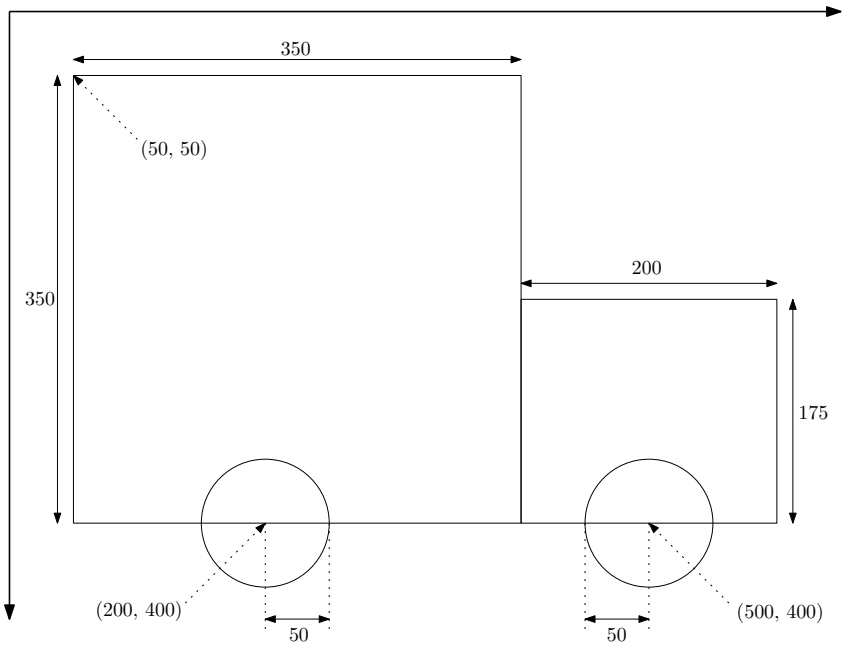
Lien vers le projet gitlab à forker pour le TP : https://etulab.univ-amu.fr/alaboure/graphic.

Une fois le dépôt téléchargé, vous pouvez compiler et exécuter le programme en cliquant deux fois sur `graphic` -> `application` -> `run`. Vous devriez obtenir l'affichage suivant.



Objectifs

Dans cette planche de TP, vous allez implémenter des classes pour réaliser des dessins de camion. Nous vous donnons une classe (classe `Truck` du package `viewer`) permettant de dessiner le camion suivant :



Le code de cette classe est le suivant :

```
1  public class Truck extends Canvas{
2      public Truck() {
3        super(650, 550);
4        GraphicsContext graphicsContext = this.getGraphicsContext2D();
5        Painter painter = new FrenchPainter(graphicsContext);
6        draw(painter);
7      }
8
9      private void draw(Painter painter) {
10       painter.drawRectangle(50.0, 50.0, 350.0, 350.0);
11       painter.drawRectangle(400.0, 225.0, 200.0, 175.0);
12       painter.drawCircle(200.0, 400.0, 50.0);
13       painter.drawCircle(500.0, 400.0, 50.0);
14     }
15 }
```

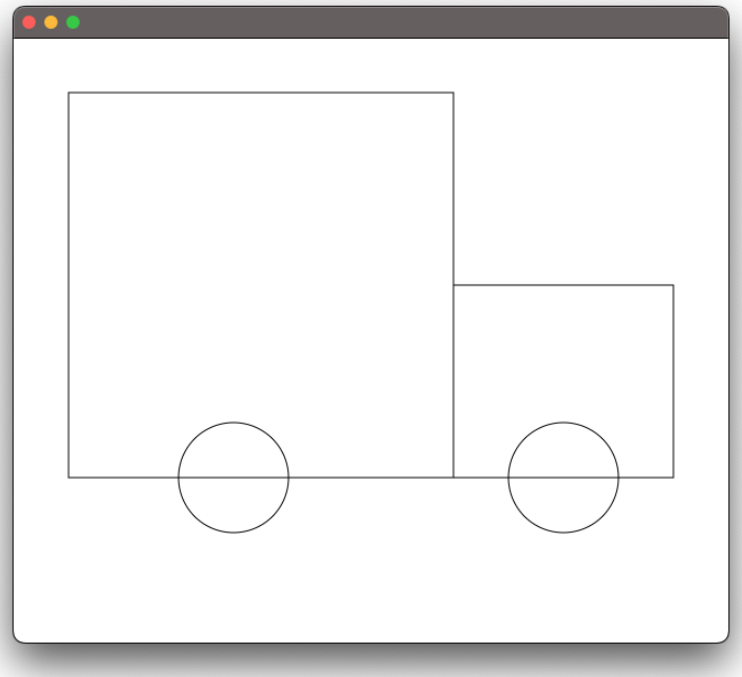
Pour le dessin, on utilise l'interface `Painter` suivante :

```
1  public interface Painter {
2    /**
3     * Draws a rectangle.
4     *
5     * @param xUpperLeft the X position of the upper left corner of the
6     * rectangle.
7     * @param yUpperLeft the Y position of the upper left corner of the
8     * rectangle.
9     * @param width the width of the rectangle.
10    * @param height the height of the rectangle.
11    */
12   void drawRectangle(double xUpperLeft, double yUpperLeft, double width,
        double height);
13
```

```java
    /**
     * Draws a circle.
     *
     * @param xCenter the X position of the center of the circle.
     * @param yCenter the Y position of the center of the circle.
     * @param radius the radius of the circle.
     */
    void drawCircle(double xCenter, double yCenter, double radius);
}
```

Le but est donc de dessiner des rectangles et des cercles. Pour cela il vous faudra utiliser les méthodes `strokeRect` et `strokeOval` de la classe `GraphicsContext` de JavaFX.

Tâche 1 : Complétez la classe `FrenchPainter` dans le package `french` qui implémente l'interface `Painter`. Le but est d'obtenir l'affichage ci-dessous :



Un anglais a développé une classe `EnglishPainter` qui permet de dessiner des cercles et des rectangles. L'objectif est d'utiliser la classe `EnglishPainter` sans la modifier. Elle possède deux méthodes utilisant la classe `Point2D` de JavaFX :

— `void paintRectangle(Point2D upperLeft, Point2D lowerRight)` (où `upperLeft` et `lowerRight` sont les deux coins opposés du rectangle) ;
— `void drawCircle(Point2D center, Point2D point)` (où center et point sont respectivement le centre et un point du cercle).

La classe `EnglishPainter` 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 de code :

```
1 Painter painter= new FrenchPainter(graphicsContext);
```

par

```
1 Painter painter = new EnglishPainterAdapter(graphicsContext);
```

dans la classe Truck.

> **Tâche 2 :** Proposez une implémentation de la classe `EnglishPainterAdapter` qui ne demande aucune modification du code des classes déjà présentes dans le code. Le but est d'obtenir l'affichage ci-dessous :