TP1 : Développement d'IHM avec JavaFX
« Premières fenêtres avec JavaFX

- Lisez et copier le code suivant correspondant à une classe «PremiereFenetre » après avoir créé un nouveau projet TP1 dans votre IDE.

```
package TP1;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
public class PremiereFenetre extends Application {
public void start(Stage premierStage)  {
     /*créer un objet Group qui sera la racine composant conteneur qui n'applique aucune disposition spéciale à ses enfants. Tous les composants enfants (nœuds) sont positionnés à 0,0*/
     Group root = new Group();
 //Créer une scène en passant l'objet Group, la hauteur et la largeur
   Scene scene = new Scene(root, 800, 600, Color.LIGHTBLUE);
 //Donner un titre au Stage (facultatif)
  premierStage.setTitle("Sample Application");
 //ajouter la scène au Stage
  premierStage.setScene(scene);
 //Rendre le contenu du Stage visible (montrer le Stage)
      premierStage.show();
  }
  public static void main(String args[]){
    launch(args);
  }
}
```

- Notre classe PremiereFenetre hérite de la classe Application, c'est la classe principale de notre application (de toute façon c'est la seule), c'est elle qui contient la fonction main() qui est le point d'entrée de notre programme. Pour mieux comprendre ce qu'elle contient, commençons par supprimer le contenu de la fonction start() en laissant juste la dernière ligne. Voici ce que devrait contenir maintenant notre classe Test :

```
public class PremiereFenetre extends Application {

   public static void main(String[] args) {
      Application.launch(args);
   }
   public void start(Stage premierStage) {
      premierStage.show();
   }
}
```

La classe **Test** contient deux fonctions :

• La fonction **main()**. Comme je l'ai dit c'est le point d'entrée de notre application. Elle appelle la fonction **launch()** qui lancera le reste du programme. C'est la seule instruction que doit contenir la fonction **main().**

• La fonction **start()**. Cette fonction est déclenchée par la fonction **launch(),** elle prend en argument un objet de type **Stage**. Vous vous souvenez... c'est le théâtre qui contiendra tout ce qui constitue l'application : la scène, les acteurs, etc... Pour l'instant la fonction **start()** ne fait que rendre visible l'objet **Stage**, c'est-à-dire la fenêtre de notre application.

Quand vous compilez, vous obtiendrez une fenêtre transparente, il n'y a même pas de fond. C'est à ça que se résume notre objet Stage : une fenêtre absolument vide.

Maintenant qu'on a créé notre théâtre on va pouvoir lui ajouter une scène, pour cela on va effectuer trois actions :

1. On crée le groupe **root** , c'est le groupe racine qui contiendra tous les autres objets et groupes d'objets graphiques. Un objet **Scene** ne contient qu'un seul groupe root. A partir de ce groupe root, on peut insérer et retirer tous les nœuds graphiques que l'on souhaite.

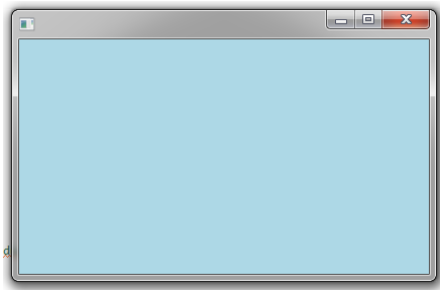
Un nœud graphique peut être :

- Un objet graphique comme un cercle, un rectangle, une image, etc.
- Un groupe contenant des objets graphiques, un groupe peut même contenir d'autres groupes.
- Ou un type d'objet graphique que nous avons nous-même définit.

2. On crée l'objet **Scene** qui contiendra le groupe **root**.
3. On ajoute l'objet **Scene** à l'objet **Stage** puis compiler.

```
@Override
public void start(Stage premierStage) {
        Group root = new Group();
   Scene scene = new Scene(root, 800, 600, Color.LIGHTBLUE);
   premierStage.setScene(scene);
   premierStage.show();
}
```

Notre fenêtre a maintenant un fond grâce à notre objet Scene qui va contenir tous les nœuds graphiques de notre application.



On peut créer et ajouter à notre scène tous les objets que l'on souhaite. Commençons par exemple par créer un cercle. Pour ça on procède en trois temps :

1. On déclare et on construit notre objet de type Circle et on règle les paramètres de cet objet comme on le souhaite (sa couleur, sa taille, etc...)
3. On ajoute notre objet au groupe root de notre objet Scene.

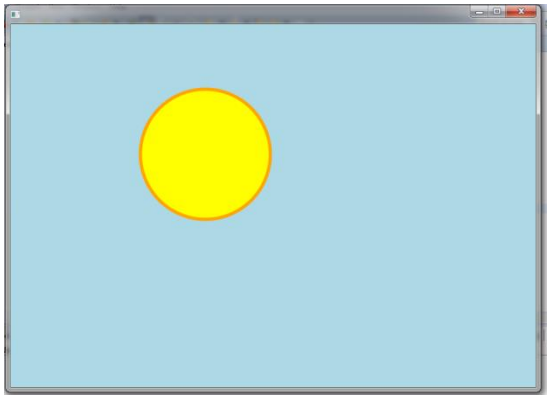
```
    @Override
    public void start(Stage premierStage) {
         Group root = new Group();
        Scene scene = new Scene(root, 800, 600, Color.LIGHTBLUE);
        premierStage.setScene(scene);
         Circle cercle = new Circle();
        cercle.setCenterX(300);//réglage de la position, de la taille et de la couleur du cercle
        cercle.setCenterY(200);
        cercle.setRadius(100);
        cercle.setFill(Color.YELLOW);
        cercle.setStroke(Color.ORANGE);//réglage de la couleur de la bordure et de son épaisseur
        cercle.setStrokeWidth(5);
        root.getChildren().add(cercle);//on ajoute le cercle au groupe root
        premierStage.show();
    }
}
```

Retenez bien la fonction groupe.getChildren().add(objet), c'est grâce à elle que l'on peut ajouter un nœud graphique à un groupe quelconque. Nous l'utiliseront très souvent.

Si vous faites un copier – coller de ce code, une erreur devrait apparaître. Cette erreur est due au fait que vous n'avez pas importé les bibliothèques nécessaires pour utiliser des objets de type Circle. Utilisez le raccourci Ctrl+Shift+M, qui détectera automatiquement les bibliothèques

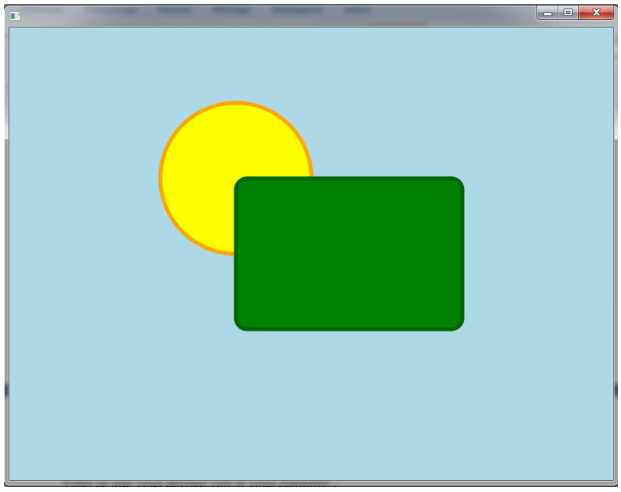
manquantes et les ajoutera en haut de votre feuille de code, vous pouvez vérifier.

Voici ce que vous devriez voir si vous compilez :



Ajoutons maintenant un rectangle vert à notre scène. Pour cela, créons un objet de type Rectangle et procédons de la même façon que pour le cercle :

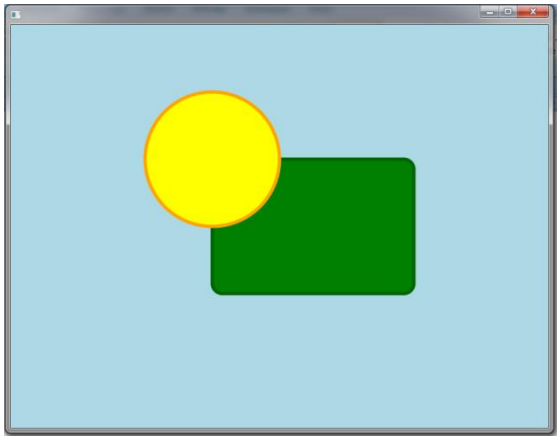
```
public void start(Stage premierStage) {    ....
    Rectangle rectangle = new Rectangle();
    rectangle.setX(300);
    rectangle.setY(200);
    rectangle.setWidth(300);
    rectangle.setHeight(200);
    rectangle.setFill(Color.GREEN);
    rectangle.setStroke(Color.DARKGREEN);
    rectangle.setStrokeWidth(5);
    rectangle.setArcHeight(30);
    rectangle.setArcWidth(30);
    root.getChildren().add(cercle);
    root.getChildren().add(rectangle);//on ajoute le rectangle après le cercle au root
...    }
```



On voit que notre rectangle apparaît devant notre cercle. C'est parce qu'on a inséré le rectangle au groupe root après avoir inséré le cercle. Si on inverse l'ordre dans lequel on insère ces deux nœuds graphiques :

```
root.getChildren().add(rectangle);//On ajoute dabord le rectangle
root.getChildren().add(cercle);//puis le cercle
```

voici le résultat de la compilation :



L'ordre d'apparition des nœuds dans notre scène correspond tout simplement à l'ordre dans lequel nous les avons insérés dans notre groupe root.