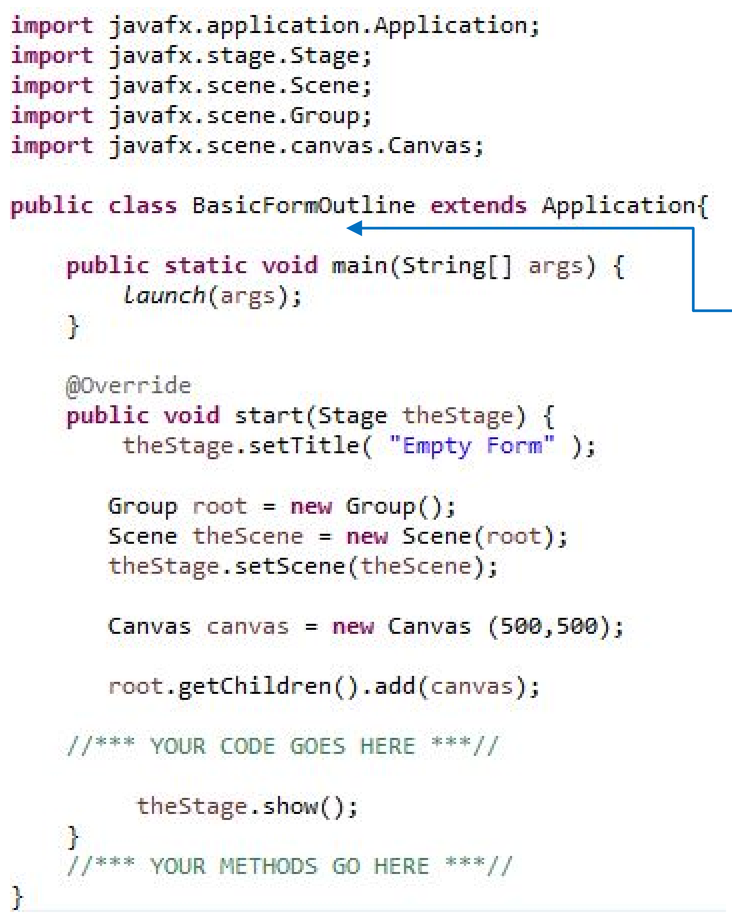
Java FX: Mouse Events

Learning Goal: Create a java program that reacts to the location of a mouse click

- We are going to create a mini-game in which the player earns a point every time the target is clicked.
- Our first goal is to display the coordinates of any mouse click by the user

PART A: Mouse Coordinates

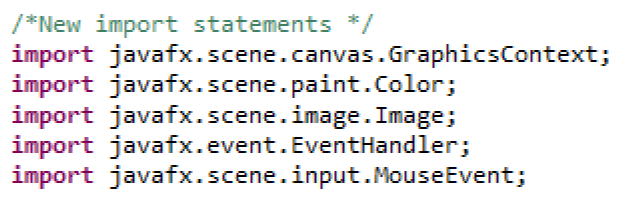
1) Create a new class called TargetPractice. Start with the general javafx form outline we have been using:



```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.canvas.Canvas;

public class BasicFormOutline extends Application{

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage theStage) {
        theStage.setTitle( "Empty Form" );

          Group root = new Group();
          Scene theScene = new Scene(root);
          theStage.setScene(theScene);

          Canvas canvas = new Canvas (500,500);

          root.getChildren().add(canvas);

    //*** YOUR CODE GOES HERE ***//

          theStage.show();
    }
    //*** YOUR METHODS GO HERE ***//
}
```

Change this class name to TargetPractice

2) Introduce the following additional import statements



```
/*New import statements */
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.image.Image;
import javafx.event.EventHandler;
import javafx.scene.input.MouseEvent;
```

All new code will be added in the `//**** YOUR CODE GOES HERE ****//` section

3) Add a new image called “bullseye.png” to your project, and then write the following three sections:

```
/* Import the bullseye image */
GraphicsContext gc = canvas.getGraphicsContext2D();
Image bullseye = new Image("bullseye.png");

/* Draw the blank background */
gc.setFill(new Color (0.85,0.85,1.0,1.0));
gc.fillRect(0, 0, 512, 512);


// draw the bullseye at x = 20, y = 20
gc.drawImage(bullseye, 20, 20);
```

- Run your program to make sure everything works!

4) Now for the important part! We want to know the current mouse click coordinates. Adding the following code will “handle” all mouse click events

```
/*Handle the mouse click events */
theScene.setOnMouseClicked(
       new EventHandler<MouseEvent>()
       {
           public void handle(MouseEvent e)
           {
               /*Everything in here will execute once
                * every time the mouse gets clicked
                */
               System.out.println("Mouse coordinates: "+e.getX()+ ", "+e.getY());


            }
        }
);
```

- Run your program and check that it displays each mouse click set of coordinates

PART B: Add the Target Practice!

- Update your class from part A to be more functional

1) Add just one more import statement:

```
/* new import statement */
import javafx.scene.shape.Circle;
```

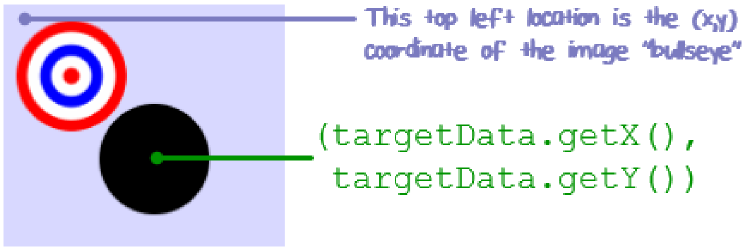
2) Create a global integer variable (place it outside and above all methods, but inside the class declaration) called points. Initialize it to 0 inside the start method.

3) It is much easier to interact with predefined objects like Circles, Rectangles, Triangles etc. in javafx than it is to interact with images. For this reason, we will always create an invisible shape that will follow our images around the screen. For our bullseye, the best choice is a circle.

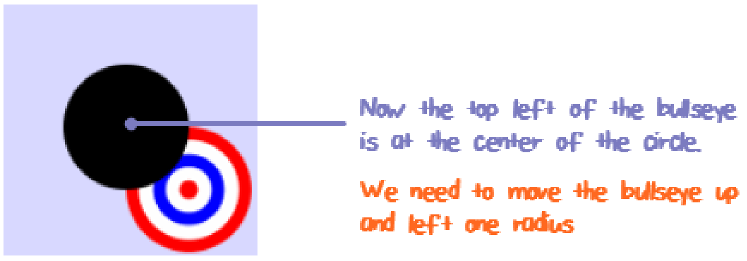
```
// create a circle with identical size to the
// bullseye (radius 32)
Circle targetData = new Circle (100,100,32);
```

There is some trial and error with this step. Use `root.getChildren().add(targetData);` in order to see the circle and tweak your object size

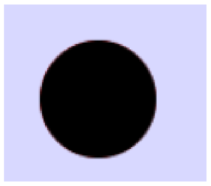
4) Now we need to move our bullseye image right behind our Circle called targetData


This top left location is the (x,y) coordinate of the image "bullseye"
(targetData.getX(), targetData.getY())

If we move the target to the same location as the targetData Circle this will happen:


Now the top left of the bullseye is at the center of the circle.
We need to move the bullseye up and left one radius

Finally...



```
gc.drawImage(bullseye,
targetData.getX()-targetData.getRadius(),
targetData.getY()-targetData.getRadius())
```

All of that to explain this little section of code:

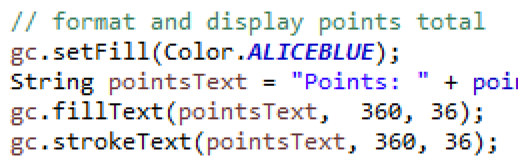
Replace:

```
// draw the bullseye at x = 20, y = 20
gc.drawImage(bullseye, 20, 20);
```

With:

```
// draw the bullseye right on top of our Circle
gc.drawImage(bullseye,
             targetData.getCenterX()-targetData.getRadius(),
             targetData.getCenterY()-targetData.getRadius());
```

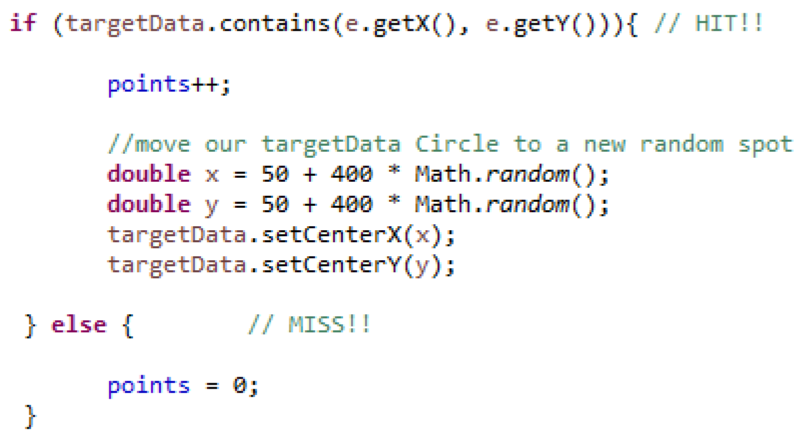
5) Since this is a game after all, we need to display the users points:



```
// format and display points total
gc.setFill(Color.ALICEBLUE);
String pointsText = "Points: " + points;
gc.fillText(pointsText,  360, 36);
gc.strokeText(pointsText, 360, 36);
```

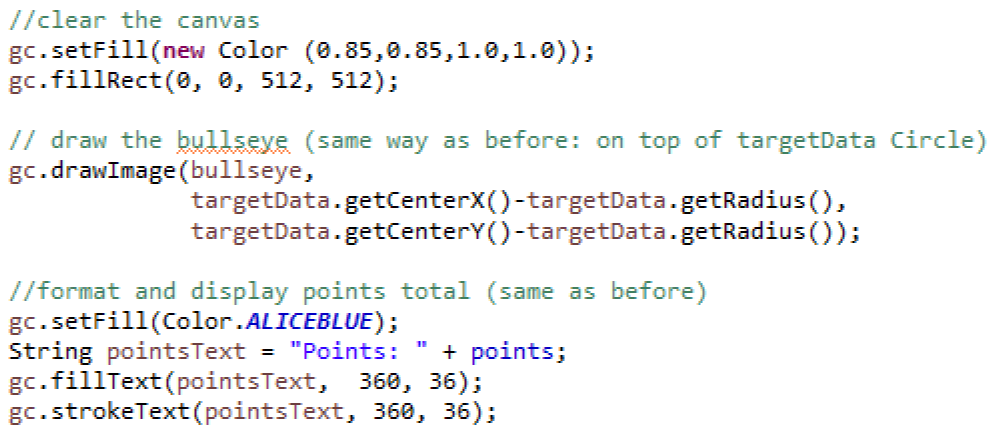
- Run your program – it should display the target and zero points

6) Now find your mouse click event where you displayed the mouse coordinates. Right underneath that line of code, add the following:



```
if (targetData.contains(e.getX(), e.getY())){ // HIT!!

       points++;

       //move our targetData Circle to a new random spot
       double x = 50 + 400 * Math.random();
       double y = 50 + 400 * Math.random();
       targetData.setCenterX(x);
       targetData.setCenterY(y);

 } else {        // MISS!!

       points = 0;
 }
```

7) And finally, right after the code for step 6, redraw the entire scene:



```
//clear the canvas
gc.setFill(new Color (0.85,0.85,1.0,1.0));
gc.fillRect(0, 0, 512, 512);

// draw the bullseye (same way as before: on top of targetData Circle)
gc.drawImage(bullseye,
             targetData.getCenterX()-targetData.getRadius(),
             targetData.getCenterY()-targetData.getRadius());

//format and display points total (same as before)
gc.setFill(Color.ALICEBLUE);
String pointsText = "Points: " + points;
gc.fillText(pointsText,  360, 36);
gc.strokeText(pointsText, 360, 36);
```

ICS3U JavaFX 06: Mouse Events

<u>Programming Exercise:</u>

- Open up Paint and create one new image and replace the target. The shape must be something other than a circle. Edit your code from TargetPractice to allow the user to get points for clicking on your new object.

<u>Hand this small project into the dropbox please!</u>