**PROJECT2**

**Reminder**

* [Asking Questions for Assignments](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=content&rcode=uco-3257182)
* Plagiarism (Academic Integrity paragraph) at [Syllabus](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=content&rcode=uco-3197635)

**Create a project to write Java programs as follows:**

* VSCode project name: project2
* **If not given here, make your program run like this video demo:**[**https://youtu.be/kcm4bWY6n6M**](https://youtu.be/kcm4bWY6n6M)
* Create a Java window:
  + Window size: 700 x 700 pixels
  + Window title: "Project 2 - Car Collision"
* Four arrow keys: To move the Car by 10 pixels at a time
  + As the Car collides with circles or squares, the part of the Car that had collision becomes RED, and the all circles and squares that had collision with the Car disappear from the screen.
    - The damaged part of the Car changes its color to red.
    - If the damaged part makes collision one more time, the corresponding part is removed from the Car structure.
  + When all parts of the Car is destroyed, "Game Over" is displayed on the canvas.
  + **Program Tip**: Except the canvas, you must call **.setFocusable(false)** from ALL GUI component that you are going to click on. i.e., all Buttons. Otherwise, once a button is clicked, the button receives all Key events.
* "**Add 10**" button: To place 5 circles (size = 20) and 5 squares (size = 20) at random positions.
* "**Clear ALL**" button: To clear obstacles (circles and squares) except the Car
  + No changes on the car and "Game Over".
* "**Fix Car**": The Car is reset to its initial state (no damage) and position.
  + No changes on the obstacles, but remove "Game Over" since it restarts the game.

**The Program Structural Requirements 1 (No credits may be given if violated)**

* As implementation begins, enable git and add remote repo at GitHub (**private** repo)
  + GitHub remote repo naming requirement: *<YourUCOEmail>*-se3103-project2
  + E.g., my UCO email is hsung@uco.edu, then the remote repo name should be: **hsung-se3103-project2**
* **Java GUI** window and other components should be created and initialized as shown in Lessons videos using Java Swing (and some AWT). The use of JavaFX is not allowed.
* The use of **MVC architecture**: Separation codes among model, view, controller functions.
* **Use of "drawing canvas"** in the same way Lecture2 uses.
  + Use the canvas in the same way Lecture uses it especially the use of **paintComponent( )** method in **JPanel** class to implement the canvas.
  + Use of **JavaFX** or **java.awt.Canvas** class is not allowed.

**The Program Structural Requirements 2 (No credits may be given if violated)**

* **CarCollision.java**: **Use this with no changes**

public interface CarCollision {

     Rectangle2D.Double getCollisionBox();

}

* + getCollisionBox( ) is implemented at **Circle** and **Square** classes, not at MyShape. See below drawing for the definition of the CollisionBox
  + To detect collision between two rectangles (collisionbox), use **intersects( )** method of **Rectangle2D** class.
* **class MyShape implements CarCollision**
  + An abstract class to represent common behaviors and properties of shapes; Circle, Square
    - common properties: location, size, color
    - common behavior: draw (use polymorphism)
  + Note: An abstract method for detecting collision is not provided here. It will be implemented at sub classes of MyShape
  + Refer to the partial code shown below.
* **class Circle extends MyShape**
  + Override the draw method to render a circle on the canvas.
  + size  == the diameter (radius x 2) of the circle
  + location ==  the center of the circle
  + implement **getCollisonBox( )**
    - Refer to the picture below for the size of the collision box.
  + Use size, location, color from MyShape. Do not declare them again in Circle.
  + Refer to the partial code shown below.
* **class Square extends MyShape**
  + Override the draw method to render a square on the canvas.
  + size  == one side of the square
  + location == the center of the square
  + implement **getCollisonBox( )**
    - The collision box as the same size as the rectangle represented by this Square object.
  + Use size, location, color from MyShape. Do not declare them again in Square.
  + Refer to the partial code shown below.
* **class MyCar**
  + frontTire: a Circle object of size 20
  + rearTire: a Circle object of size 20
  + bottomBodyFront: a Square object of size 30
  + bottomBodyRear: a Square object of size 30
  + topBody: a Square object of size 30
  + Refer to the partial code shown below for the relative positions of those car parts.
  + The car moves by 10 pixels at each arrow key press
* **Use polymorphism**
  + to render all shapes on the canvas
  + to get the collision box and to detect collisions

**Submission:**

* Download the source code from GitHub
* Submit the zip file to [Project2](https://learn.uco.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=376620&type=dropbox&rcode=uco-2008091)
  + Late submission of up to 12 hours is accepted with an hourly late penalty (2 points per hour out of 100)
* Post the video demo link at the corresponding Student Video Presentation Forum (Project2 demo)
  + The same video demo requirements as Video 1 of Lesson Assignments.
    - The running program demo only. Do **not** show/explain the source code
  + Using the running app, show that all project requirements (except program structural requirements) are met
  + Max video length: 3 min

**Grading Policy:**

* **Zero** **credits** if
  + compile errors or program crashes at runtime
  + the program produces trivial or no meaningful display on canvas
  + the program does not use MVC architecture as shown in the lecture (Program Structural Requirements 1)
  + the program does not comply with Program Structural Requirements 2
  + the submitted program is not a download from GitHub
  + the video demo is not submitted
* Partial credits are given if not all features are implemented as required.

**Properties (instance variables) and the constructors of MyShape, Circle, Square, and MyCar are shown below. Use them and add more properties and behaviors as needed.**

public abstract class **MyShape** implements CarCollision {

     // location = center of Circle or Square (not upper-left corner)

     private final Point2D.Double location;

     private Color color;

     // size = diameter (2 x radius) if Circle

     //        a side if Square

     private final int size;

     public MyShape(Point2D.Double location, Color color, int size) {

          this.location = location;

          this.color = color;

          this.size = size;

     }

     public abstract void draw(Graphics2D g2);

            . . .

            . . .

} // end of MyShape

public class **Circle** extends MyShape {

     public Circle(int x, int y, Color color, int size) {

          super(new Point2D.Double(x, y), color, size);

     }

     @Override

     public void draw(Graphics2D g2) {

          ... // implement here!

     }

     @Override

     public Rectangle2D.Double getCollisionBox() {

          ... // implement here!

     }

} // end of Circle

public class **Square** extends MyShape {

     public Square(int x, int y, Color color, int size) {

          super(new Point2D.Double(x, y), color, size);

     }

     @Override

     public void draw(Graphics2D g2) {

          ... // implement here!

     }

     @Override

     public Rectangle2D.Double getCollisionBox() {

          ... // implement here

     }

} // end of Square

Text

Description automatically generated

A picture containing text, athletic game

Description automatically generated