COSC1337 Programming Fundamentals II /ITSE2357 Advanced Object-Oriented Programming

The submission opens two days before the due day, and closes right at the due time on due day. One submission is allowed per project. It is students' responsibilities to submit timely and correctly in order to get credits. Students MUST start each project when it is first discussed in class in order to complete it on time. All projects are individual projects. Project due days may be changed to better align with lectures.

If one of the following is true, not credits will be given:

- The project is late.
- The algorithm or class design is missing.
- The project has syntax errors.
- There are no comments (Javadoc format required) at all in your code.
- Wrong files are submitted.
- **A project is a copy and modification of another student's project. (Both will receive 0.)**

Files to be submitted through Canvas:

- The UML design (UML class diagram)
- All source codes and driver programs
- All supporting files if any

Software Development Project Rubric: (Design is worth 20%; the rest is worth 80 %.)
Analysis: There will be no credit if the software doesn't meet customer specification at all.

- Does the software meet the exact customer specification?
- Does the software read the exact input data and display the exact output data as they are shown in sample runs?
- Does each class include all corresponding data and functionalities?

Design: There will be no credit if the design is missing.

- Is the design (a UML class diagram) an efficient solution?
- Is the design created correctly?

Code: There will be no credit if there are syntactic errors.

- Are there errors in the software?
- Are code conventions and name conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?

Debug:

- Are there bugs in the software?

Documentation: There will be no credit if comments are not included.

- Are there enough comments included in the software?
- Class comments must be included before a class header.
- Method comments must be included before a method header.
- More comments must be included inside each method.
- All comments must be written in Javadoc format.

Project 6 - Bouncing balls

Develop an application that displays balls bouncing in a frame. It is easier to take two steps:

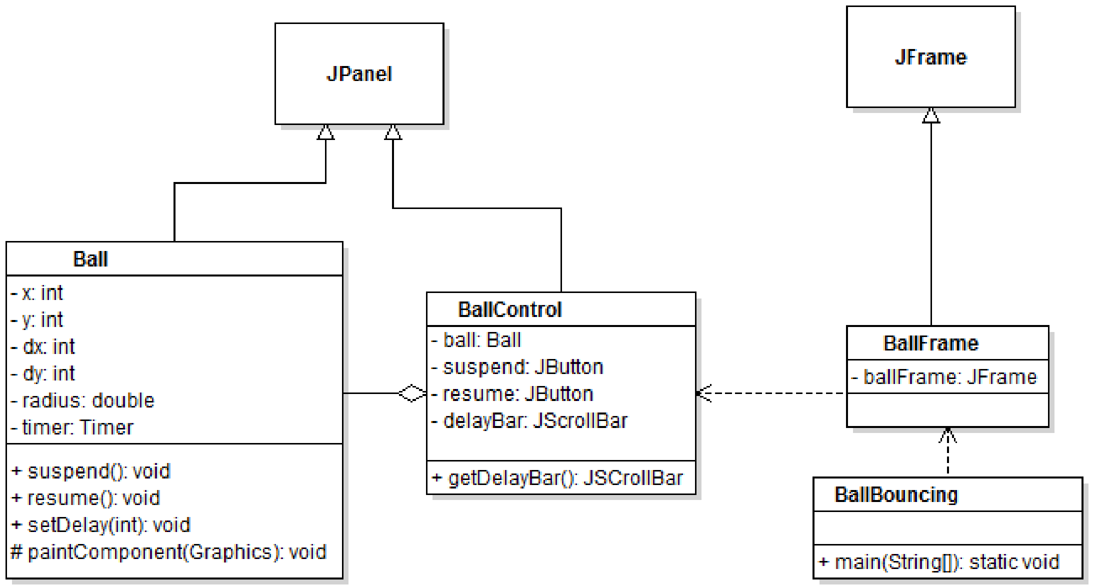
- bounce one ball

  Use two buttons to Suspend and Resume the movement, and use a scroll bar to control the bouncing speed. Run oneball.jar, a runnable jar file of this project, to get more information.

- bounce multiple balls

  Add button +1 and button-1 button to increase and decrease the number of balls respectively, and use the Suspend and Resume button to freeze the balls or resume bouncing. Run moreBalls.jar, a runnable jar file of this project, to get more information.

One Bounce Ball



Note: The diagram shows the design of each panel and the frame. All listeners must be outer classes. Add them into the design.

Class *Ball*, a subclass of *JPanel*, represents a ball with a timer. Its radius is *radius*. It is located at (*x*,*y*). The timer's delay time is 10 milliseconds by default. Every delay interval, the timer fires an action event, and the ball should move by *dx* horizontally and by *dy* vertically. The timer should be started automatically when the ball is created. Its delay can be changed via method *setDelay*. The timer is stopped in method suspend; and it is started again in method resume. A ball is not allowed to move out of boundary.

timer: A timer of this ball with a delay in milliseconds and an action listener
x: x coordinate of this ball
y: y coordinate of this ball
radius: Radius of this ball
dx: Increment on this ball's x-coordinate
dy: Increment on this ball's y-coordinate

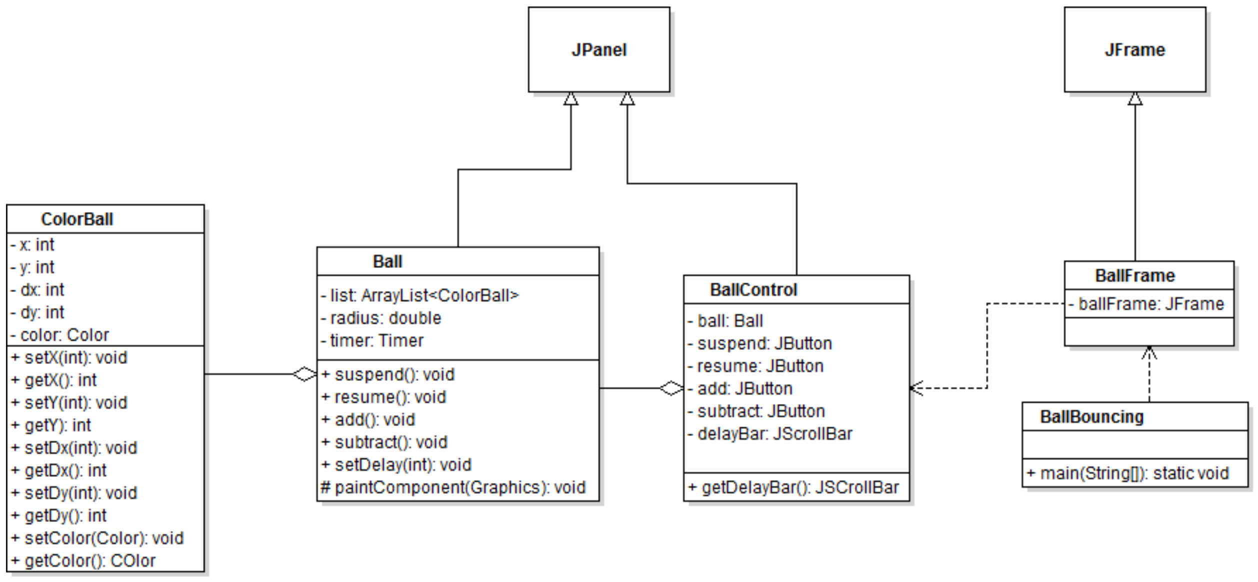
Class *BallControl*, a subclass of *JPanel*, contains the ball panel, a scroll bar, and two control buttons *suspend* and *resume*. When the program starts, a ball is moving. The ball stops when *suspend* is pressed; the ball continues to move when *resume* is pressed. The ball moves faster when the scroll bar is moved to the right; the ball moves slower when the scroll bar is moved to the left.

Class *BallFrame*, a subclass of *JFrame* , displays an instance of *BallControl* in a frame.

Class *BallBouncing* is a driver program that starts the entire program as follows:

```
public class BallBouncing{
      public static void main(String[] args){
            new BallFrame();
  }
}
```

Multiple Bounce Balls



Now class *Ball*, a subclass of *JPanel* must paint multiple balls with random colors. This class should have a reference to a list of colored balls that should be represented in a separate class.

Class *ColorBall* represents a ball that is located at (*x*,*y*), moves *dx* horizontally and by *dy* vertically, and has a random color.

Class *Ball*, a subclass of *JPanel*, represents a list of colored balls with a timer. All balls have the same radius, *radius*. The timer's delay time is 10 milliseconds by default. Every delay interval, the timer fires an action event, and each ball should move by *dx* horizontally and by *dy* vertically. The timer should be started automatically when the ball is created. Its delay can be changed via method *setDelay*. The timer is stopped in method suspend; and it is started again in method resume. A ball is not allowed to move out of boundary.

list: A array list of colored balss
timer: A timer of a ball with a delay in milliseconds and an action listener
radius: Radius of a ball

Class *BallControl* needs two more buttons, +1 and button-1 button to increase and decrease the number of balls respectively,