6.092 Introduction to Software Engineering in Java
January (IAP) 2009

6.092 - Assignment 6: Graphics strikes back!

Part Zero: Setup

1. Download the following classes from the Stellar website: `SimpleDraw, DrawGraphics, BouncingBox, BouncingDevice`
2. **Important: do not reuse your code from the previous assignment. The files have changed.**
3. Create a new project with these files as usual.
4. Make sure that the program runs normally. You should see a red box and a yellow box move in the window.

Part One: Inheritance

1. Make the class BouncingBox inherit from the class BouncingDevice (use keyword **extends**)
2. Move the following fields from the BouncingBox class to the BouncingDevice class:

```
int x;
int y;
Color color;
int xDirection = 0;
int yDirection = 0;
final int SIZE = 20;
```

3. **Checkpoint**: Make sure that your code runs normally.

Part Two: Constructors

1. Create a constructor in the class BouncingDevice that has the same signature as the BouncingBox one, i.e.:

```
public BouncingDevice(int startX, int startY, Color startColor);
```

2. Copy the body of the BouncingBox constructor into the BouncingDevice constructor.
3. Erase the body of the BouncingBox constructor and replace it by a call to the super-class constructor (use the keyword **super**).
4. **Checkpoint**: Make sure that your code runs normally.

Part Three: the BouncingBall class

1. Create a new class called BouncingBall based on this example:

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;

public class BouncingBall {
}
```

2. Make sure that you save the class in the correct filename (BouncingBall.java).

3. Make the BouncingBall class inherit from the BouncingDevice class (use the keyword **extends**).
4. Your code should not compile as is. Do you know why? If not, ask an instructor.
5. Add a constructor to the BouncingBall class that has the same signature as the one in BouncingDevice, that is:

```
public BouncingBall(int startX, int startY, Color startColor);
```

6. Make the constructor call the super-class constructor (use the keyword **super**).
7. **Checkpoint**: Make sure that your code runs normally.

Part Four: Mixing balls and boxes

1. Create a draw method in the BouncingBall class that has the same signature as the one in BouncingBox, that is:

```
public void draw(Graphics2D surface);
```

2. In this method, write code to draw and animate the ball using the `java.awt.Graphics2D` class (use `fillOval` for example). **Note: be simple here. Simply copy-paste the draw method from BouncingBox to BouncingBall, and replace fillRect by fillOval (and drawRect by drawOval)**.
3. Similarly, add a `moveInDirection` method to the BouncingBall class that does the same thing as in the BouncingBox class.
4. Finally, add a BouncingBall object to the `DrawGraphics` class. Make sure it is animated by calling `moveInDirection` at the right place.
5. **Checkpoint**: Make sure that your code runs normally and that the ball and the boxes are animated.

Part Five: Using inheritance

1. At this point, you should have noticed that your code has a lot of duplicates. For example, the `draw` methods in the BouncingBox class and in the BouncingBall class have a lot in common.
2. Create a method in the `BouncingDevice` class called `animate`. The method should have the following signature:

```
public void animate()
```

3. In this method, copy the code that is in common between the `draw` methods in the BouncingBox class and in the BouncingBall class (i.e. the code to move the center and to reverse direction when hitting an edge).
4. Remove the corresponding code from the `draw` methods in the BouncingBox class and in the BouncingBall class and replace it with a call to `animate`.
5. **Checkpoint**: Make sure that your code runs normally.
6. Similarly, remove the `moveInDirection` method from the BouncingBall and BouncingBox classes, and move it to the BouncingDevice class.
7. **Checkpoint**: Make sure that your code runs normally.

Optional: Crazy balls!

1. In the BouncingDevice class, make the SIZE field non-final.
2. In the BouncingBall class, make the size of the ball change when it hits an edge. You will need to use some code written in the `animate` method of the BouncingDevice class.
3. Play around with the parameters and add more fun features if you have the time. For example, you could make the color of the ball change over time, etc.