23-10 Bouncing Ball

\*\*\* 程式寫法 \*\*\*

整題被包在 BouncingBall 這個類別當中。

關於 GUI 的部份，我使用了一個 JFrame 物件與一個 MyCanvas 物件。

MyCanvas 其實是一個 JPanel 的子類別，並且 implement 了 MouseListener 這個 interface 因為題目要求在 (第一個) mouse click 的時候創造一顆藍色的球。

至於內部的邏輯部份，我使用自製的 Container 以及 Ball 類別，來 store 所有相關資訊。

Container 其實就是球所能移動的範圍，因此在 Ball 的 move() method 裡，就需要一個 Container 物件的資訊來判斷是否需要「彈回」。

Ball 這個類別是用來儲存所有關於一顆球應該有的資訊，比如位置、速度、顏色等。

(當然在此題當中球只有一個，而且一定是藍色。)

\*\*\* code 功能 \*\*\*

在第一下的 mouse click 發生時，會創建一個 Ball 物件在滑鼠點擊的位置，並且固定為藍色。

這個 Ball 物件的速度，不管是 x 分量還是 y 分量，都是 random 出來的。

而 random 的範圍為 1~10，也就是用 (new Random()).nextInt(10) + 1。

\*\*\* 執行流程 \*\*\*

(運用提供的 Makefile)

1. 使用 $make 指令來編譯

2. 使用 $make run1 指令來執行此題

\*\*\* 是否已完成 \*\*\*

已完成。

23-11 Bouncing Balls

\*\*\* 程式寫法 \*\*\*

大部分的程式碼與上一題極度相似。除了以下幾點：

1. 因為要支援多顆球，所以原本的 field 有一個 private Ball ball; 被改為 private ArrayList<Ball> balls;

2. 原本的 private boolean hasBall; 是用來判斷創造了球了沒，但是現在不需要了

(因為可以用 balls.size() == 0 來判斷是否已經有球)

3. 在 MyCanvas override 的 mouseClicked() 函式裡會支援每次的點擊都創造一顆球, i.e., calling addBall()

4. 新增了 addBall() method，基本上就是 balls.add("新的球"); 而這顆球的顏色會是各別 random R, G, and B values 而成。

\*\*\* code 功能 \*\*\*

在每一下的 mouse click 皆會創造一顆新的球，顏色為隨機，而初始速度的隨機方式與上一題相同。

\*\*\* 執行流程 \*\*\*

(運用提供的 Makefile)

1. 使用 $make 指令來編譯

2. 使用 $make run2 指令來執行此題

\*\*\* 是否已完成 \*\*\*

已完成。