**Example 1: Getting Started with One Bouncing Ball**

**BouncingBallSimple.java:** Writing a single ball bouncing inside a rectangular container box is straight forward, and can be accomplished with very few lines of codes, as follows:

import java.awt.\*;

import java.util.Formatter;

import javax.swing.\*;

/\*\*

\* One ball bouncing inside a rectangular box.

\* All codes in one file. Poor design!

\*/

// Extends JPanel, so as to override the paintComponent() for custom rendering codes.

public class **BouncingBallSimple extends JPanel** {

// Container box's width and height

private static final int BOX\_WIDTH = 640;

private static final int BOX\_HEIGHT = 480;

// Ball's properties

private float ballRadius = 200; // Ball's radius

private float ballX = ballRadius + 50; // Ball's center (x, y)

private float ballY = ballRadius + 20;

private float ballSpeedX = 3; // Ball's speed for x and y

private float ballSpeedY = 2;

private static final int UPDATE\_RATE = 30; // Number of refresh per second

/\*\* Constructor to create the UI components and init game objects. \*/

public BouncingBallSimple() {

this.setPreferredSize(new Dimension(BOX\_WIDTH, BOX\_HEIGHT));

// Start the ball bouncing (in its own thread)

Thread gameThread = new Thread() {

public void run() {

while (true) { // Execute one update step

// Calculate the ball's new position

ballX += ballSpeedX;

ballY += ballSpeedY;

// Check if the ball moves over the bounds

// If so, adjust the position and speed.

if (ballX - ballRadius < 0) {

ballSpeedX = -ballSpeedX; // Reflect along normal

ballX = ballRadius; // Re-position the ball at the edge

} else if (ballX + ballRadius > BOX\_WIDTH) {

ballSpeedX = -ballSpeedX;

ballX = BOX\_WIDTH - ballRadius;

}

// May cross both x and y bounds

if (ballY - ballRadius < 0) {

ballSpeedY = -ballSpeedY;

ballY = ballRadius;

} else if (ballY + ballRadius > BOX\_HEIGHT) {

ballSpeedY = -ballSpeedY;

ballY = BOX\_HEIGHT - ballRadius;

}

// Refresh the display

repaint(); // Callback paintComponent()

// Delay for timing control and give other threads a chance

try {

Thread.sleep(1000 / UPDATE\_RATE); // milliseconds

} catch (InterruptedException ex) { }

}

}

};

gameThread.start(); // Callback run()

}

/\*\* Custom rendering codes for drawing the JPanel \*/

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g); // Paint background

// Draw the box

g.setColor(Color.BLACK);

g.fillRect(0, 0, BOX\_WIDTH, BOX\_HEIGHT);

// Draw the ball

g.setColor(Color.BLUE);

g.fillOval((int) (ballX - ballRadius), (int) (ballY - ballRadius),

(int)(2 \* ballRadius), (int)(2 \* ballRadius));

// Display the ball's information

g.setColor(Color.WHITE);

g.setFont(new Font("Courier New", Font.PLAIN, 12));

StringBuilder sb = new StringBuilder();

Formatter formatter = new Formatter(sb);

formatter.format("Ball @(%3.0f,%3.0f) Speed=(%2.0f,%2.0f)", ballX, ballY,

ballSpeedX, ballSpeedY);

g.drawString(sb.toString(), 20, 30);

}

/\*\* main program (entry point) \*/

public static void main(String[] args) {

// Run GUI in the Event Dispatcher Thread (EDT) instead of main thread.

javax.swing.SwingUtilities.invokeLater(new Runnable() {

public void run() {

// Set up main window (using Swing's Jframe)

JFrame frame = new JFrame("A Bouncing Ball");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setContentPane(new BouncingBallSimple());

frame.pack();

frame.setVisible(true);

}

});

}

}