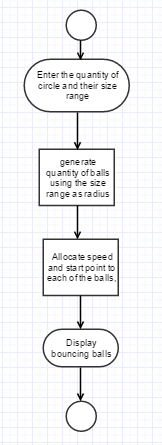
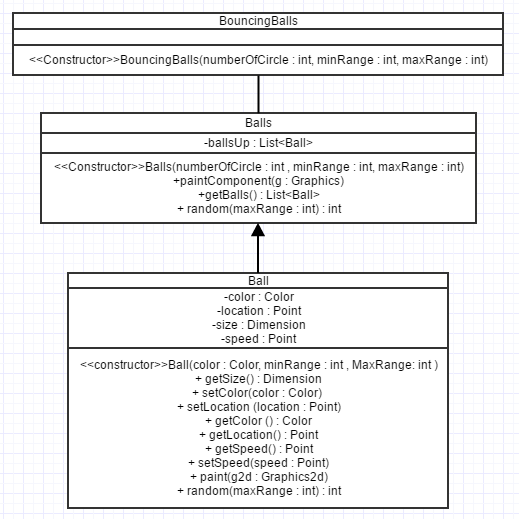
Assignment 4

This is a simple multiple Bouncing ball application.

Analysis

Design





Implementation

BouncingBalls.class

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

public class BouncingBalls {

public static void main(String[] args) {

JTextField numberOfCircles = new JTextField();

JTextField rangeOfCircleSize = new JTextField();

Object[] message = {

"Enter number of circles ", numberOfCircles,

"Enter the range of sizes ", rangeOfCircleSize

};

int option = JOptionPane.showConfirmDialog(null, message, "Input", JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if (option == JOptionPane.OK\_OPTION) {

String[] range = rangeOfCircleSize.getText().split("\\-");

BouncingBalls simpleBalls = new BouncingBalls(Integer.parseInt(numberOfCircles.getText()),

Integer.parseInt(range[0].trim()),

Integer.parseInt(range[1].trim()));

} else {

System.out.println("Canceled");

System.exit(0);

}

}

public BouncingBalls(final int numberOfCircle, final int minRange, final int maxRange) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException ex) {

} catch (InstantiationException ex) {

} catch (IllegalAccessException ex) {

} catch (UnsupportedLookAndFeelException ex) {

}

JFrame frame = new JFrame("Application Title");

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

frame.setLayout(new BorderLayout());

Balls balls = new Balls(numberOfCircle, minRange, maxRange);

frame.add(balls);

frame.setSize(400, 400);

frame.setVisible(true);

new Thread(new BounceEngine(balls)).start();

}

});

}

}

Balls.class

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.RenderingHints;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JPanel;

public class Balls extends JPanel {

/\*\*

\*

\*/

private static final long serialVersionUID = -1503263973313757150L;

private List<Ball> ballsUp;

public Balls(int numberOfCircle, int minRange, int maxRange) {

ballsUp = new ArrayList<Ball>(25);

//10 + random(90)

for (int index = 0; index < numberOfCircle; index++) {

ballsUp.add(new Ball(new Color(random(255), random(255), random(255)), minRange, maxRange));

}

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g.create();

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

for (Ball ball : ballsUp) {

ball.paint(g2d);

}

g2d.dispose();

}

public List<Ball> getBalls() {

return ballsUp;

}

public static int random(int maxRange) {

return (int) Math.round((Math.random() \* maxRange));

}

}

Ball.class

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics2D;

import java.awt.Point;

public class Ball {

private Color color;

private Point location;

private Dimension size;

private Point speed;

public Ball(Color color, int minRange, int MaxRange) {

setColor(color);

speed = new Point(10 - random(20), 10 - random(20));

// speed = new Point(10, 30);

int range = random(MaxRange);

size = new Dimension(range, range);

}

public Dimension getSize() {

return size;

}

public void setColor(Color color) {

this.color = color;

}

public void setLocation(Point location) {

this.location = location;

}

public Color getColor() {

return color;

}

public Point getLocation() {

return location;

}

public Point getSpeed() {

return speed;

}

public void setSpeed(Point speed) {

this.speed = speed;

}

protected void paint(Graphics2D g2d) {

Point p = getLocation();

if (p != null) {

g2d.setColor(getColor());

Dimension size = getSize();

g2d.fillOval(p.x, p.y, size.width, size.height);

}

}

public static int random(int maxRange) {

return (int) Math.round((Math.random() \* maxRange));

}

}

BounceEngine.class

import java.awt.Dimension;

import java.awt.Point;

import javax.swing.SwingUtilities;

public class BounceEngine implements Runnable {

private Balls parent;

public BounceEngine(Balls parent) {

this.parent = parent;

}

@Override

public void run() {

int width = getParent().getWidth();

int height = getParent().getHeight();

// Randomize the starting position...

for (Ball ball : getParent().getBalls()) {

int x = random(width);

int y = random(height);

Dimension size = ball.getSize();

if (x + size.width > width) {

x = width - size.width;

}

if (y + size.height > height) {

y = height - size.height;

}

ball.setLocation(new Point(x, y));

}

while (getParent().isVisible()) {

// Repaint the balls pen...

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

getParent().repaint();

}

});

// This is a little dangrous, as it's possible

// for a repaint to occur while we're updating...

for (Ball ball : getParent().getBalls()) {

move(ball);

}

// Some small delay...

try {

Thread.sleep(100);

} catch (InterruptedException ex) {

}

}

}

public Balls getParent() {

return parent;

}

public void move(Ball ball) {

Point p = ball.getLocation();

Point speed = ball.getSpeed();

Dimension size = ball.getSize();

int vx = speed.x;

int vy = speed.y;

int x = p.x;

int y = p.y;

if (x + vx < 0 || x + size.width + vx > getParent().getWidth()) {

vx \*= -1;

}

if (y + vy < 0 || y + size.height + vy > getParent().getHeight()) {

vy \*= -1;

}

x += vx;

y += vy;

ball.setSpeed(new Point(vx, vy));

ball.setLocation(new Point(x, y));

}

public static int random(int maxRange) {

return (int) Math.round((Math.random() \* maxRange));

}

}

Ouput Screenshot

