

**Masters in Applied Software Technology**

**Software Design & Testing**

**Assignment 2**

**Date : 02/12/2016**

**Student Name : Ciarán Sweeney**

**Student Number : C12317616**

**Chain Reaction Class**

This class sets up the layout for the chain reaction game and the main method is also in this class. I have also have a method which runs the secret drop down menu which is unseen by the user until he or she presses ‘d’ on the keyboard. The secret method allows the user to chooses which level he or she wants to play. The drop down options for this menu is created in the fill Levels method. I will not go much more in depth about this class as the class works basically the same as my draw Pad class from the last assignment.

**Game Area Class**

Game Area class is where the large about of the work is done. In this class there are three threads that can run at the same time. The gameThread will be always running in my code. This thread is used to chose which level of the game to play and also loads up the screen for winning and losing. This thread in a way acts like a finite state machine.

Paint Component Method

This method brings all the balls, text and explosions to the screen.

Game SetUp Method

This method is used to set up the values in th fields up. Balls are created here and so is the ballThread. The ballThread is used as a thread that deals with the balls movement by updating the positions of the balls with the updateBalls method.

Update Balls Method

This method as I said updates the positions of the balls but it also checks if the ball hits off the wall using the wallCollison method. In the updateBalls there is also a method for checking if a ball has hit into an explosion and if so the ball will explode too. This is done with the explosionCollision method.

Explosion method

This method is involved with all things that explodes. This method adds explosions to the explosion list to be drawn on screen. The timing of removing an explosion is done with setting the thread eThread to sleep.

Score Calculation method

This method is used to calculate the score of the game. The formula is score+=100\*(n\*n\*n). I found this formula out by using wolfram alpha.

Reset All Method

This method is used to reset all values for the next level

Play Level Method

This method is used as a waiting screen for the next level until the user clicks on the screen.

Game Over Method

This method is used to print game over on the screen as you ran out of levels.

Set Level method

This method is called from the secret menu in the Chain Reaction class. This method changes the level and allows the user to skip forward.

**Explosion class**

This class is used to create explosions and give the explosion colour, and position. The position of the explosion is passed into this class by a click on the screen which is done by the moues.

Area Of Explosion Method

This method uses the distance formula to work out the distance from the explosion and a ball that was passed into this method. If the ball distance from the explosion is less or equal of the radius of the explosion then the ball will explode too.

**Ball Class**

This ball creates a ball and gives its speed, direction, colour and so on.

Get Chain Method

Returns how many explosion there was before this one has hit into it.

Get First

Checks if this explosion was the first one.

**GameArea**

package chainReaction;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.RenderingHints;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.util.ArrayList;

import java.util.Random;

import javax.swing.\*;

public class GameArea extends JPanel{

private ArrayList<Ball> balls=new ArrayList<Ball>();

private ArrayList<Ball> removingballs=new ArrayList<Ball>();

private ArrayList<Explosion> explosionList=new ArrayList<Explosion>();

private int maxX;

private int maxY;

private int minX=0;

private int minY=0;

private double mouseX=0;

private double mouseY=0;

private double cMouseX=0;

private double cMouseY=0;

private int click=0;

private int level=0;

private int ballCount=0;

private int goal=0;

private int score=0;

private boolean enter=true;

private boolean endLevel=false;

private String messageA="";

private String messageB="";

//private String levelMessage="";

private Random rand = new Random();

public GameArea(int x, int y){

maxX=x;

maxY=y;

addMouseMotionListener(new MouseAdapter(){

//used to keep track of the cursors position

public void mouseMoved(MouseEvent e) {

cMouseX=e.getX();

cMouseY=e.getY();

}

});

addMouseListener(new MouseAdapter(){

//this keeps track of the mouse being pressed

public void mousePressed(MouseEvent e){

mouseX=e.getX();

mouseY=e.getY();

if(click==0){

explosion(new Explosion(mouseX,mouseY,new Color(1f,1f,1f,0.5f),50,1,true));

click++;

}

else if(messageA.length()>1 || messageB.length()>1){

click++;

}

}

});

Thread gameThread=new Thread(new Runnable(){

//Thread for Levels. This thread picks a level

public void run(){

do{

//level 0

if(level==0){

welcome();

}

//level 1

else if(enter && level==1){

playLevel();

enter=false;

click=0;

gameSetUp(1,5);

}

//level 2

else if(enter && level==2){

playLevel();

enter=false;

click=0;

gameSetUp(2,10);

}

//level 3

else if(enter && level==3){

playLevel();

enter=false;

click=0;

gameSetUp(4,15);

}

//level 4

else if(enter && level==4){

playLevel();

enter=false;

click=0;

gameSetUp(6,20);

}

//level 5

else if(enter && level==5){

playLevel();

enter=false;

click=0;

gameSetUp(10,25);

}

//level 6

else if(enter && level==6){

playLevel();

enter=false;

click=0;

gameSetUp(15,30);

}

//level 7

else if(enter && level==7){

playLevel();

enter=false;

click=0;

gameSetUp(18,35);

}

//level 8

else if(enter && level==8){

playLevel();

enter=false;

click=0;

gameSetUp(22,40);

}

//level 9

else if(enter && level==9){

playLevel();

enter=false;

click=0;

gameSetUp(30,45);

}

//level 10

else if(enter && level==10){

playLevel();

enter=false;

click=0;

gameSetUp(37,50);

}

//level 11

else if(enter && level==11){

playLevel();

enter=false;

click=0;

gameSetUp(48,55);

}

//level 12

else if(enter && level==12){

playLevel();

enter=false;

click=0;

gameSetUp(54,60);

}

//Game over

else if(level==13)

gameOver();

try {

Thread.sleep(1);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}while(true);

}

});

gameThread.setDaemon(true);

gameThread.start();

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

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

g2d.setRenderingHint(RenderingHints.KEY\_RENDERING,RenderingHints.VALUE\_RENDER\_QUALITY);

g.setFont(new Font("TimesRoman", Font.PLAIN, 20));

//previw

if(click==0)

new Explosion(cMouseX,cMouseY,new Color(1f,1f,1f,0.5f),50,1,true).draw(g);

g.setColor(Color.white);

//draws messages

g.drawString(messageA, (maxX/2)-100, (maxY/2)-48);

g.drawString(messageB, (maxX/2)-100, (maxY/2));

//draws balls

for(int i=0 ; i<balls.size(); i++){

balls.get(i).draw(g);

}

//draws explodsions

for(int i=0 ; i<explosionList.size(); i++)

explosionList.get(i).draw(g);

}

//sets up the game

public void gameSetUp(int g, int numBalls){

goal=g;

//adds in balls

setBalls(numBalls);

repaint();

Thread ballThread=new Thread(new Runnable(){

public void run(){

click=0;

updateBalls();

resetAll();

}

});

ballThread.setDaemon(true);

ballThread.start();

}

public void setBalls(int numberBalls){

//randomly generates the vales ro create ball

int randomNumX;

int randomNumY;

int speed;

boolean randomBoolX;

boolean randomBoolY;

for(int j=0; j<numberBalls; j++){

randomBoolX=rand.nextBoolean();

randomBoolY=rand.nextBoolean();

randomNumX=rand.nextInt((maxX - minX) + 1) + minX;

randomNumY= rand.nextInt((maxY - minY) + 1) + minY;

speed= rand.nextInt((4 - 1) + 1) + 1;

balls.add(new Ball(randomNumX,randomNumY,randomBoolX,randomBoolY,randomColor(),10,speed));

}

}

public Color randomColor(){

//Generates random colours

float r = rand.nextFloat();

float g = rand.nextFloat();

float b = rand.nextFloat();

return new Color(r,g,b,0.5f);

}

public void updateBalls(){

do{

for(Ball b:balls){

b.updatePostions();

wallCollison(b);

explsionCossion(b);

}

for(Ball b:removingballs){

balls.remove(b);

ballCount++;

}

removingballs.clear();

this.repaint();

try {

Thread.sleep(20);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}while(!endLevel);

}

//This is where and how the ball bounces off

public void wallCollison(Ball b){

if(b.getPosX()>=maxX-25){

b.setForward(false);

}

else if(b.getPosX()<=minX){

b.setForward(true);

}

if(b.getPosY()>=maxY-65){

b.setDown(false);

}

else if(b.getPosY()<=minY){

b.setDown(true);

}

}

public synchronized void explosion(Explosion ex){

//Creates a thread for creating and removing explosions

Thread eThread=new Thread(new Runnable(){

public void run(){

try {

explosionList.add(ex);

//sends the thread to sleep for 3 seconds

Thread.sleep(3000);

explosionList.remove(ex);

scoreCalculation(ex.getChain(),ex);

if(explosionList.isEmpty()){

repaint();

endLevel=true;

}

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

});

eThread.setDaemon(true);

eThread.start();

}

// deals with balls hitting with the explodsions then in turn turns into an explosion

public void explsionCossion(Ball b){

for(int i=0 ; i<explosionList.size(); i++){

if(explosionList.get(i).areaOfExplosion(b) && explosionList.get(i).getFirst()){

removingballs.add(b);

explosion(new Explosion(b.getPosX(),b.getPosY(),b.getColour(),50,explosionList.get(i).getChain(),false));

}

else if(explosionList.get(i).areaOfExplosion(b)){

removingballs.add(b);

explosion(new Explosion(b.getPosX(),b.getPosY(),b.getColour(),50,explosionList.get(i).getChain()+1,false));

}

}

}

//Calculates the score

public void scoreCalculation(int n,Explosion ex){

//the formula for adding to the score

if(ex.getFirst())

n--;

score+=100\*(n\*n\*n);

System.out.println(score);

}

//Checks to see if the play has beaten the level

public void winLevel(){

if(ballCount>=goal){

messageA="You won! your score was:"+score;

level++;

}

else

messageA="You lost, try again";

}

//resets every thing for next level or for replay level

public void resetAll(){

winLevel();

ballCount=0;

goal=0;

score=0;

balls.clear();

endLevel=false;

enter=true;

}

//Prints out welcome to the game until clicks the screen

public void welcome(){

click=1;

messageA="Welcome to the Game";

messageB="click anywhere to play";

do{

repaint();

}while(click<2);

messageA="";

messageB="";

click=0;

level++;

}

//prints a message at the start of each level

public void playLevel(){

click=1;

messageB="click anywhere to play level "+level;

do{

repaint();

}while(click<2);

messageA="";

messageB="";

click=0;

}

//prints game over

public void gameOver(){

click=1;

messageB="Gameover";

do{

repaint();

}while(true);

}

//picks the level

public void setLevel(int level) {

this.level = level;

endLevel=true;

balls.clear();

explosionList.clear();

messageA="level:"+level;

messageB="click anywhere to play";

repaint();

}

public int getScore() {

return score;

}

}

**ChainReaction**

package chainReaction;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Scanner;

import javax.swing.\*;

public class ChainReaction extends JFrame{

//Generating the fields

private int xSize=500;

private int ySize=500;

private int count=0;

private boolean secret=false;

private ArrayList<String> levelNames=new ArrayList<String>();

public ChainReaction(){

super(" JAVA Application!");

setLocation(300,200);

setSize(xSize,ySize);

setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

setContentPane(createConP());

setVisible(true);

}

private Container createConP(){

Container pane = new JPanel(new BorderLayout());

GameArea ga=new GameArea(xSize,ySize);

ga.setBackground(Color.black);

JMenuBar bar = new JMenuBar();

//checks to see if the menu will be hidden

String s="Score:"+ga.getScore();

bar.add(new JMenu(s));

if(secret){

JMenu secretMenu = new JMenu ("Secret Menu");

fillLevels(ga,secretMenu);

bar.add(secretMenu);

}

pane.add(bar,BorderLayout.NORTH);

pane.add(ga, BorderLayout.CENTER);

return pane;

}

//Adds the levels to the hidden menu

private void fillLevels(GameArea ga,JMenuItem sm){

JMenuItem lv=new JMenuItem("Level 1");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(1);

}

});

sm.add(lv);

lv=new JMenuItem("Level 2");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(2);

}

});

sm.add(lv);

lv=new JMenuItem("Level 3");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(3);

}

});

sm.add(lv);

lv=new JMenuItem("Level 4");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(4);

}

});

sm.add(lv);

lv=new JMenuItem("Level 5");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(5);

}

});

sm.add(lv);

lv=new JMenuItem("Level 6");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(6);

}

});

sm.add(lv);

lv=new JMenuItem("Level 7");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(7);

}

});

sm.add(lv);

lv=new JMenuItem("Level 8");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(8);

}

});

sm.add(lv);

lv=new JMenuItem("Level 9");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(9);

}

});

sm.add(lv);

lv=new JMenuItem("Level 10");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(10);

}

});

sm.add(lv);

lv=new JMenuItem("Level 11");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(11);

}

});

sm.add(lv);

lv=new JMenuItem("Level 12");

lv.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

ga.setLevel(12);

}

});

sm.add(lv);

}

public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable(){

public void run(){

new ChainReaction();

}

});

}

}

**Explosion**

package chainReaction;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.geom.Ellipse2D;

public class Explosion {

//Generating the fields

private double posX;

private double posY;

private Color colour;

private double size;

private int chain;

private boolean first;

private double xCentre;

private double yCentre;

public Explosion(double x, double y,Color c,double s,int ch,boolean b){

posX=x;

posY=y;

colour=c;

size=s;

xCentre=posX-(size/2);

yCentre=posY-(size/2);

chain=ch;

first=b;

}

//Draws out the explosions =

public void draw(Graphics g){

Graphics2D g2d = (Graphics2D)g;

g.setColor(colour);

Ellipse2D.Double circle = new Ellipse2D.Double(xCentre, yCentre, size, size);

g2d.fill(circle);

//If this is not the first explosion in the demo then print the points

if(!(first)){

String s=""+point(chain);

g.drawString(s,(int)xCentre, (int)yCentre);

}

}

//Use the distance formula to check if the ball hits into the explsion

public boolean areaOfExplosion(Ball b){

//distance formaul

double xBall=b.getPosX();

double yBall=b.getPosY();

double x= Math.pow(posX-xBall,2);

double y= Math.pow(posY-yBall,2);

double distance=Math.sqrt(x+y);

//if the distance less than the radius

if(distance<=(size/2)){

return true;

}

return false;

}

public int getChain() {

return chain;

}

//checks if this is the first expodesion

public boolean getFirst(){

return first;

}

//calculates the point for this chain

public int point(int n){

return 100\*(n\*n\*n);

}

}

Ball

package chainReaction;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.geom.Ellipse2D;

public class Ball {

private double posX;

private double posY;

private boolean forward;

private boolean down;

private Color colour;

private double size;

private double speed;

public Ball(double x, double y, boolean f, boolean d, Color c,double s, double sp){

posX=x;

posY=y;

forward=f;

down=d;

colour=c;

size=s;

speed=sp;

posX=posX-(size/2);

posY=posY-(size/2);

}

public void draw(Graphics g){

Graphics2D g2d = (Graphics2D)g;

g.setColor(colour);

Ellipse2D.Double circle = new Ellipse2D.Double(posX, posY, size, size);//

g2d.fill(circle);//

//g.drawOval(0,0,0,0);//sort this out me

}

public double getPosX() {

return posX;

}

public double getPosY() {

return posY;

}

public Color getColour() {

return colour;

}

public void setForward(boolean forward) {

this.forward = forward;

}

public void setDown(boolean down) {

this.down = down;

}

public void updatePostions(){

if(forward==true){

posX+=speed;

}

else{

posX-=speed;

}

if(down==true){

posY+=speed;

}

else{

posY-=speed;

}

}

}