=====Ryan’s Code=====

package menu;  
  
import dungeonQuest.Tabloid;  
  
import resources.\*;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
  
import java.util.Random;  
  
import spacevader.SpaceGame;  
import pong.PongTable;  
  
*/\*\*  
 \* By Ryan  
 \*/*public class Menu extends JFrame{  
  
 static private Random *gen* = new Random();  
  
 private JLabel title;  
 private JLabel sel;  
 private JLabel option1;  
 private JLabel option2;  
 private JLabel option3;  
 private JLabel exit;  
 private JLabel copyright;  
 private JLabel names;  
  
 private int selLocation = 1;  
  
 public Menu() {  
 super("Atari Emulator 1040");  
 this.setBounds(0, 0, 1500, 1000);  
 this.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 this.getContentPane().setBackground(Color.*BLACK*);  
 this.setLayout(null);  
  
 addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch(e.getKeyCode()) {  
 case KeyEvent.*VK\_DOWN*: {  
 if (selLocation == 1 || selLocation == 2){  
 sel.setLocation(sel.getX(), sel.getY() + 100);  
 selLocation++;  
 }  
 else if(selLocation==3){  
 sel.setLocation(sel.getX(), sel.getY() + 200);  
 selLocation++;  
 }  
 else{  
 sel.setLocation(425, 375);  
 selLocation = 1;  
 }  
 break;  
 }  
 case KeyEvent.*VK\_UP*: {  
 if (selLocation == 2 || selLocation == 3){  
 sel.setLocation(sel.getX(), sel.getY() - 100);  
 selLocation--;  
 }  
 else if(selLocation==4){  
 sel.setLocation(sel.getX(), sel.getY() - 200);  
 selLocation--;  
 }  
 else{  
 sel.setLocation(425, 775);  
 selLocation = 4;  
 }  
 break;  
 }  
 case KeyEvent.*VK\_ENTER*: {  
 if (selLocation == 1){  
 //Instantiate Space Invaders Object  
 new SpaceGame();  
 dispose();  
 }  
 else if(selLocation == 2){  
 //myJoke joke=new myJoke();  
 //Instantiate Jeff's Quest Object  
 new Tabloid();  
 dispose();  
  
 }  
 else if(selLocation == 3){  
 //Instantiate Pong Object  
 new PongTable();  
 dispose();  
 }  
 else if(selLocation == 4){  
 dispose();  
 }  
 break;  
 }  
 }  
 }  
 });  
  
 this.createTitle();  
 this.createOptions();  
 this.addSel();  
  
 this.createStars();  
  
 this.setVisible(true);  
  
 }  
  
 private void createTitle(){  
 title = new JLabel("");  
 title.setSize(new Dimension(800,300));  
 ImageIcon theImage = new ImageIcon(this.getClass().getResource("/resources/images/LogoResized.png"));  
 title.setIcon(theImage);  
 title.setLocation(350,0);  
 title.setHorizontalAlignment(SwingConstants.*CENTER*);  
 title.setVisible(true);  
 this.add(title);  
  
 copyright = new JLabel("");  
 copyright.setSize(new Dimension(350,100));  
 ImageIcon copyrightImage = new ImageIcon(this.getClass().getResource("/resources/images/Copyright Stuff.png"));  
 copyright.setIcon(copyrightImage);  
 copyright.setLocation(25,875);  
 copyright.setHorizontalAlignment(SwingConstants.*LEFT*);  
 copyright.setVisible(true);  
 this.add(copyright);  
  
 names = new JLabel("");  
 names.setSize(new Dimension(350,100));  
 ImageIcon namesImage = new ImageIcon(this.getClass().getResource("/resources/images/Naems.png"));  
 names.setIcon(namesImage);  
 names.setLocation(1100,875);  
 names.setHorizontalAlignment(SwingConstants.*LEFT*);  
 names.setVisible(true);  
 this.add(names);  
 }  
 private void createOptions(){  
 option1 = new JLabel("");  
 option1.setSize(new Dimension(400,100));  
 option1.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/SpaceInvadersLogoWhite.png")));  
 option1.setLocation(550,350);  
 option1.setHorizontalAlignment(SwingConstants.*CENTER*);  
 option1.setVisible(true);  
  
 option2 = new JLabel("Jeff's Quest");  
 option2.setSize(new Dimension(400,100));  
 option2.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/JeffLogoWhite.png")));  
 option2.setLocation(550,450);  
 option2.setHorizontalAlignment(SwingConstants.*CENTER*);  
 option2.setVisible(true);  
  
 option3 = new JLabel("");  
 option3.setSize(new Dimension(400,100));  
 option3.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/Pong White.png")));  
 option3.setFont(new Font("Comic Sans MS", Font.*BOLD*, 36));  
 option3.setLocation(550,550);  
 option3.setHorizontalAlignment(SwingConstants.*CENTER*);  
 option3.setVisible(true);  
  
 exit = new JLabel("");  
 exit.setForeground(Color.*WHITE*);  
 exit.setSize(new Dimension(400,100));  
 exit.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/QuitWhiteLogo.png")));  
 exit.setLocation(550,750);  
 exit.setHorizontalAlignment(SwingConstants.*CENTER*);  
 exit.setVisible(true);  
  
 this.add(option1);  
 this.add(option2);  
 this.add(option3);  
 this.add(exit);  
 }  
  
 private void addSel(){  
 sel = new JLabel("");  
 sel.setSize(new Dimension(100,50));  
 sel.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/ArrowResizedWhite.png")));  
 sel.setLocation(425,375);  
 sel.setHorizontalAlignment(SwingConstants.*CENTER*);  
 sel.setVisible(true);  
 this.add(sel);  
 }  
  
 private void createStars(){  
 for(int i = 0; i<=50; i++){  
 GameRectangle star = new GameRectangle(*gen*.nextInt(1400)+50, *gen*.nextInt(850)+100, 3,3);  
 star.setBackground(Color.*WHITE*);  
 star.setVisible(true);  
 this.add(star);  
 }  
 }  
  
  
}



=====Sean’s Code=====

package spacevader;  
import player.highScoreSpaceInvaders;  
import resources.GameRectangle;  
import menu.Menu;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.util.\*;  
import java.util.Timer;  
  
*/\*\*  
 \* Created By Sean  
 \*/*public class SpaceGame extends JFrame{  
  
 private Alien[][] fleet;  
 private Ship ship;  
 private ArrayList<Bullet> bullets;  
 private Timer t = new Timer();  
 private int fps = 40;  
  
 private boolean leftPressed = false;  
 private boolean rightPressed = false;  
 private boolean spacePressed = false;  
 private final int speed = 20;  
  
 private int row = 10;//number per row  
 private int column = 4;//number of rows  
 private int aliensLeft;  
 private int alienTime = 5;  
 private int alienMove = alienTime;  
 private int alienPos = 0; //0-8, controls where aliens move to next  
  
 private int score=0;  
 private JLabel scoreBoard;  
  
 public SpaceGame()  
 {  
 super("Space Invaders");  
 setBounds(0,0,1500, 1000);  
 getContentPane().setBackground(Color.*black*);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(null);  
  
 fleet = new Alien[row][column];  
 makeAliens();  
  
 ship = new Ship(getWidth()/2+107/2, getHeight()-30-107-30);  
 add(ship);  
  
 bullets = new ArrayList<>();  
  
 setFocusable(true);  
 addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch (e.getKeyCode()) {  
 case KeyEvent.*VK\_LEFT*:  
 if(!leftPressed) {  
 leftPressed = true;  
 ship.dx += -speed;  
 }  
 break;  
 case KeyEvent.*VK\_RIGHT*:  
 if(!rightPressed) {  
 rightPressed = true;  
 ship.dx += speed;  
 }  
 break;  
 case KeyEvent.*VK\_SPACE*:  
 if(!spacePressed) {  
 spacePressed = true;  
 Bullet b = ship.shoot(true);  
 bullets.add(b);  
 add(b);  
 score-=100;  
 }  
 break;  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 switch (e.getKeyCode()) {  
 case KeyEvent.*VK\_LEFT*:  
 leftPressed = false;  
 ship.dx -= -speed;  
 break;  
 case KeyEvent.*VK\_RIGHT*:  
 rightPressed = false;  
 ship.dx -= speed;  
 break;  
 case KeyEvent.*VK\_SPACE*:  
 spacePressed = false;  
 break;  
 }  
 }  
 });  
  
 scoreBoard = new JLabel();  
 scoreBoard.setBounds(15,15,500, 25);  
 scoreBoard.setText(""+score);  
 scoreBoard.setForeground(Color.*WHITE*);  
 scoreBoard.setFont(new Font("Magneto", Font.*ITALIC*, 30));  
 add(scoreBoard, 0);  
  
 t.schedule(new MyTimerTask(), 0, 1000/fps);  
 setVisible(true);  
 }  
  
 private void checkCollision(Bullet b) {  
 Rectangle brec = new Rectangle(b.getX(), b.getY(), b.getWidth(), b.getHeight());  
 Rectangle arec;  
 AlienCollision:  
 if(b.isUp()) {  
 for (Alien[] ar : fleet) {  
 for (Alien a : ar) {  
 if (a != null) {  
 arec = new Rectangle(a.getX(), a.getY(), a.getWidth(), a.getHeight());  
 if (arec.intersects(brec)) {  
 score+=500;  
 aliensLeft--;  
 remove(a);  
 ar[Arrays.*asList*(ar).indexOf(a)] = null;  
 remove(b);  
 bullets.remove(b);  
 break;  
 }  
 }  
 }  
 }  
 }else  
 {  
 arec = new Rectangle(ship.getX(), ship.getY(), ship.getWidth(), ship.getHeight());  
 if (arec.intersects(brec)) {  
 gameOver();  
 break AlienCollision;  
 }  
 }  
 //checks to see if any aliens are left  
 if(aliensLeft==0)  
 newRound();  
 }  
  
 private void moveFleet()  
 {  
 score++;  
 boolean left;  
 switch (alienPos) {  
 case 0:  
 left = false;  
 break;  
 case 1:  
 left = false;  
 break;  
 case 2:  
 left = true;  
 break;  
 case 3:  
 left = true;  
 break;  
 case 4:  
 left = true;  
 break;  
 case 5:  
 left = true;  
 break;  
 case 6:  
 left = false;  
 break;  
 case 7:  
 left = false;  
 break;  
 default:  
 left = false;  
 break;  
 }  
 alienPos++;  
 alienPos%=9;  
 for (Alien[] ar : fleet)  
 {  
 for (Alien a : ar) {  
 if(a!=null) {  
 if(alienPos==0) {  
 a.shift(20, 0);  
 if((a.getY()+50)>ship.getY())//checks if aliens are too close  
 {  
 gameOver();  
 }  
 }else  
 {  
 a.shift(0, (left)? -25 : 25);  
 }  
  
 //also here's the code for the aliens to shoot  
 java.util.Random gen = new Random();  
 if(gen.nextInt(100)==0) {  
 Bullet b = a.shoot(false);  
 bullets.add(b);  
 add(b);  
 }  
 }  
 }  
 }  
 }  
  
 private void makeAliens()  
 {  
 int a;//space of alien  
 int s; //space between aliens  
 int w;//space between aliens plus space of aliens  
 int h = 50 + 35; //vertical spacing  
 for(int j = 0; j<fleet[0].length; j++) {  
 switch (j)  
 {  
 case 0:  
 a = 29;  
 break;  
 case 1:  
 a = 50;  
 break;  
 case 2:  
 a = 40;  
 break;  
 case 3:  
 a = 50;  
 break;  
 default:  
 a = 0;  
 break;  
 }  
 s = (getWidth()-row\*a)/(row+1);  
 w = s + a;  
 for (int i = 0; i < row; i++) {  
 fleet[i][j] = new Alien(i \* w + s, 50+j\*h, j);  
 add(fleet[i][j]);  
 }  
 }  
 aliensLeft = row\*column;  
 }  
  
 private void newRound()  
 {  
 score+=5000;  
 t.cancel();  
 try {  
 Thread.*sleep*(750);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 makeAliens();  
 for(Bullet b : bullets)  
 remove(b);  
 bullets.clear();  
 repaint();  
 alienTime--;  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 t = new Timer();  
 t.schedule(new MyTimerTask(), 0, 1000/fps);  
 }  
  
 private void gameOver()  
 {  
 t.cancel();  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 highScoreSpaceInvaders high = new highScoreSpaceInvaders();  
 boolean newHigh = high.readSpaceInvadersHighScore()<=score;  
 if(newHigh)  
 {  
 high.writeHighScoreSpaceInvaders(score);  
 }  
 int pane = JOptionPane.*showConfirmDialog*(null,  
  
 "Game Over!\nScore: "+score+ "\n" + ((newHigh)? "New high score!" : ("High score: "+ high.readSpaceInvadersHighScore())) +"\nWould you like to try again?", "Game Over", JOptionPane.*YES\_NO\_OPTION*);  
 if(pane==0)//if try again  
 {  
 for (Alien[] ar : fleet)//get rid of aliens  
 {  
 for (Alien a : ar) {  
 if(a!=null) {  
 remove(a);  
 ar[Arrays.*asList*(ar).indexOf(a)] = null;  
 }  
 }  
 }  
 makeAliens();//add new aliens  
 for(Bullet b : bullets)//remove bullets  
 {  
 remove(b);  
 }  
 bullets.clear();  
 alienTime=5;  
 score=0;  
 t = new Timer();  
 repaint();  
 try {  
 Thread.*sleep*(500);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 t.schedule(new MyTimerTask(), 0, 1000/fps);  
 }else  
 {  
 new Menu();  
 dispose();  
 }  
 }  
  
 private void act()  
 {  
 ship.move();  
 for(int i = 0; i<bullets.size(); i++) {  
 Bullet b = bullets.get(i);  
 b.move();  
 if(b.getY()>getHeight()||b.getY()<0)  
 {  
 remove(b);  
 bullets.remove(b);  
 i--;  
 }  
 checkCollision(b);  
 }  
 alienMove--;  
 if(alienMove==0)  
 {  
 alienMove = alienTime;  
 moveFleet();  
 }  
 scoreBoard.setText(""+score);  
 repaint();  
 }  
  
 public class MyTimerTask extends TimerTask  
 {  
 @Override  
 public void run()  
 {  
 act();  
 }  
 }  
}

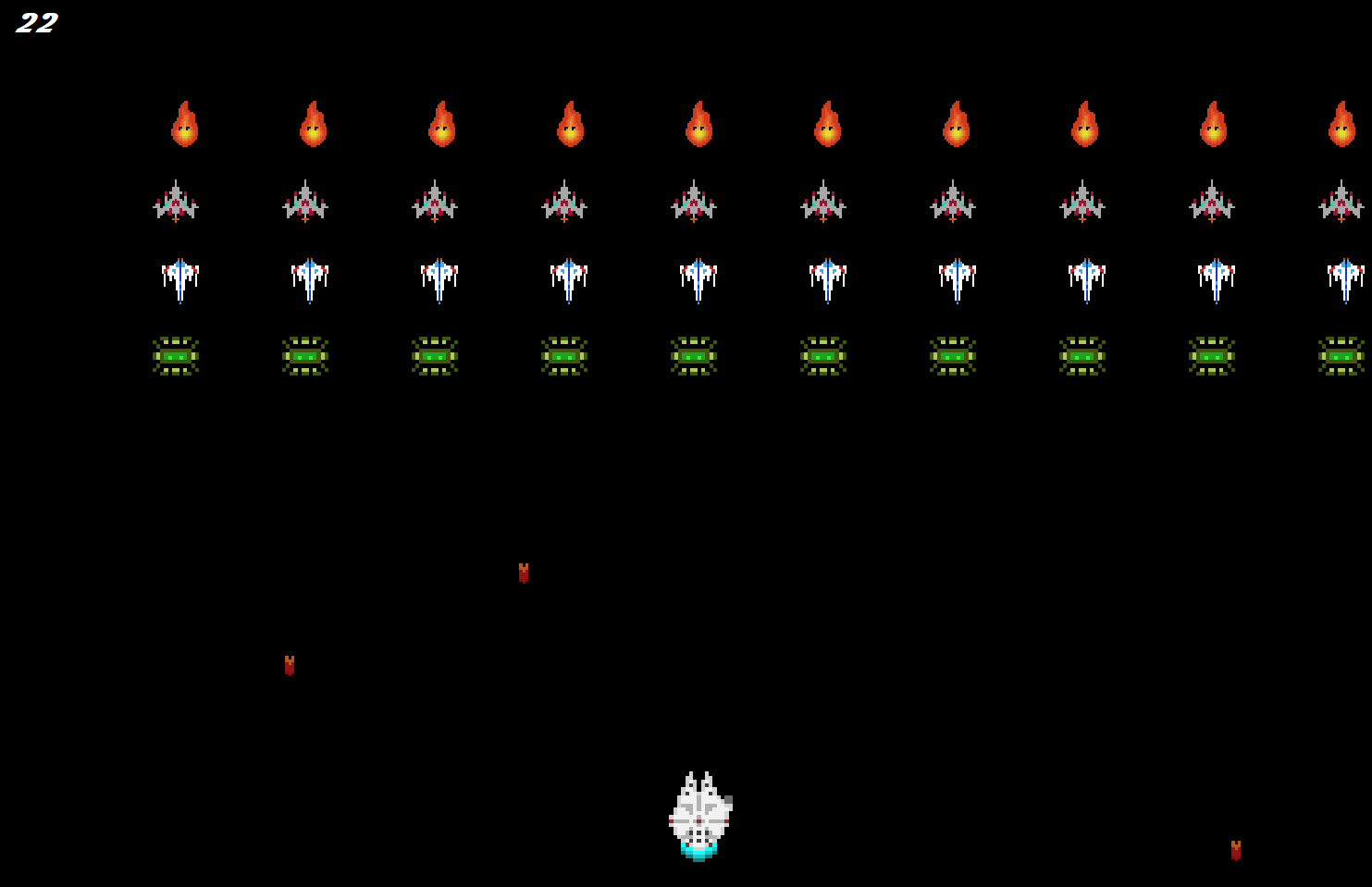
package spacevader;  
  
import resources.GameRectangle;  
  
public class SpaceEntity extends GameRectangle{  
  
 protected int x, y;  
 protected int dx = 0;  
 protected int dy = 0;  
 protected EzImage image;  
  
 protected SpaceEntity(int x, int y)  
 {  
 super(x,y,50,50);  
 this.x = x;  
 this.y = y;  
 }  
  
 protected void move()  
 {  
 x +=dx;  
 y+=dy;  
 setLocation(x,y);  
 }  
  
 protected Bullet shoot(boolean up)  
 {  
 Bullet b = new Bullet(getX()+getWidth()/2-5, getY(), up);  
 return b;  
 }  
}

package spacevader;  
  
public class Ship extends SpaceEntity{  
  
  
  
 public Ship(int x, int y)  
 {  
 super(x,y);  
 setSize(69,100);  
 image = new EzImage(-21,-2, 107, 107, "src/resources/space/MilleniumFalconResized.png");  
 add(image);  
 }  
}

package spacevader;  
  
//import java.awt.Toolkit.\*;  
import java.awt.\*;  
import javax.imageio.ImageIO;  
import java.io.\*;  
import javax.swing.JComponent;  
import java.awt.event.\*;  
import java.awt.Image;  
  
public class EzImage extends JComponent implements MouseListener {  
//implements java.awt.image.ImageObserver, MouseListener {  
 private Image content;  
  
// Constructor methods  
 public EzImage() {  
 super();  
 setBounds(0, 0, 10, 10);  
 addMouseListener(this);  
 }   
  
 public EzImage(int x, int y, int w, int h) {  
 super();  
 setBounds(x, y, w, h);  
 addMouseListener(this);  
 }   
  
 public EzImage(int x, int y, int w, int h, String s) {  
 super();  
 setBounds(x, y, w, h);  
 setImage(s);  
 addMouseListener(this);  
 }   
  
// ----- will set the image to a new picture named s -------------------------------   
 public void setImage(String s) {  
// java.net.URL url = getClass().getResource(s);   
// if (url == null) {  
// url = getClass().getResource("/"+s);  
// if (url == null)  
 try {   
 content = ImageIO.*read*(new File(s));  
 } catch(IOException ioe) {  
 ioe.printStackTrace();  
 }  
// else  
// content = getToolkit().getImage(url);  
// } else  
// content = getToolkit().getImage(url);  
  
 }  
  
 public void paint(Graphics g) {  
 g.drawImage(content, 0, 0, getWidth(), getHeight(), this);  
 paintChildren(g);  
 }   
   
 public void mouseClicked(MouseEvent e){}  
 public void mouseExited(MouseEvent e){}  
 public void mousePressed(MouseEvent e){}  
 public void mouseEntered(MouseEvent e){}  
 public void mouseReleased(MouseEvent e){}  
  
  
}

package spacevader;  
  
import java.awt.\*;  
  
public class Bullet extends SpaceEntity{  
  
 private boolean isUp;  
  
 public Bullet(int x, int y, boolean u)  
 {  
 super(x,y);  
 isUp = u;  
 setSize(11, 22);  
 String i;  
 if(u) {  
 dy = -20;  
 i="src/resources/space/LaserUP.png";  
 }  
 else {  
 dy = 20;  
 i = "src/resources/space/LaserDown.png";  
 }  
 image = new EzImage(-13, -3, 33, 33, i);  
 add(image);  
 }  
  
 protected boolean isUp()  
 {  
 return isUp;  
 }  
}

package spacevader;  
  
import java.awt.Color;  
  
  
public class Alien extends SpaceEntity{  
  
 protected Alien(int x, int y, int r)  
 {  
 super(x,y);  
 String i;  
 switch(r)  
 {  
 case 0:  
 setSize(29,50);  
 image = new EzImage(-10,0, 50, 50, "src/resources/space/FIREBALLResized.png");  
 break;  
 case 1:  
 setSize(50,47);  
 image = new EzImage(0,0, 50, 47, "src/resources/space/SPACESHIPResized.png");  
 break;  
 case 2:  
 setSize(40,50);  
 image = new EzImage(0,0, 40, 50, "src/resources/space/XWINGResized.png");  
 break;  
 case 3:  
 setSize(50,42);  
 image = new EzImage(0,0, 50, 42, "src/resources/space/BUGGYBUGResized.png");  
 break;  
 default:  
 i = "";  
 break;  
 }  
  
 add(image);  
  
 }  
  
 protected void shift(int u, int s)  
 {  
 x+=s;  
 y+=u;  
 setLocation(x,y);  
 }  
}



=====Blake’s Code=====

package dungeonQuest;  
  
  
import javafx.embed.swing.JFXPanel;  
import resources.GameRectangle;  
import spacevader.SpaceGame;  
import menu.Menu;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.Rectangle;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.util.ArrayList;  
import java.util.Random;  
import java.util.Timer;  
import java.util.TimerTask;  
  
import static java.lang.Math.*abs*;  
  
public class Tabloid extends JFrame {  
  
 static private Random *gen* = new Random();  
  
 private JLabel character;  
 private Arrow arrow;  
 private Grid grid;  
  
 private int shoot, amount, eAmount, dead, ex1, ex2, ex3, why1, why2, why3;  
  
 private Rectangle arec, grec, prec, crec, rec, herorec;  
  
 private Goat goaty;  
 private Pig piggy;  
 private Chicken chicky;  
  
  
 private ArrayList<Arrow> arrows;  
  
 private Timer t;  
 private int fps = 40;  
  
 private int arrowLocation = 1;  
  
 public Tabloid() {  
 super("Jeff's Quest");  
 ex1 = *gen*.nextInt(900);  
 ex2 = *gen*.nextInt(900);  
 ex3 = *gen*.nextInt(900);  
 why1 = *gen*.nextInt(900);  
 why2 = *gen*.nextInt(900);  
 why3 = *gen*.nextInt(900);  
 this.setBounds(0, 0, 1500, 1000);  
 this.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 this.getContentPane().setBackground(Color.*WHITE*);  
 this.setLayout(null);  
 arrows = new ArrayList<>();  
// t = new Timer();  
// System.out.println(arrows);  
// t.schedule(new Tabloid.MyTimerTask(),0,1000/fps);  
 this.createcharacter();  
 this.addGoat(ex1, why1);  
 this.addPig(ex2, why2);  
 this.addChicken(ex3, why3);  
 this.addGrid();  
 this.createGoat();  
 this.createChicken();  
 this.createPig();  
 herorec = new Rectangle(character.getX(), character.getY(),100, 100);  
 eAmount = 0;  
 grec = enemy("Goat");  
 prec = enemy("Pig");  
 crec = enemy("Chicken");  
 addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch(e.getKeyCode()) {  
 case KeyEvent.*VK\_DOWN*: amount += 1; character.setLocation(character.getX(),character.getY()+100); herorec.setLocation(character.getX(),character.getY()+100); ai(amount, eAmount); eAmount += 1; keepInBounds(); checkCollision(); break;  
 case KeyEvent.*VK\_UP*: amount += 1; character.setLocation(character.getX(),character.getY()-100);herorec.setLocation(character.getX(),character.getY()-100); ai(amount, eAmount); eAmount += 1; keepInBounds(); checkCollision(); break;  
 case KeyEvent.*VK\_LEFT*: amount += 1; character.setLocation(character.getX()-100,character.getY()); herorec.setLocation(character.getX()-100,character.getY()); ai(amount, eAmount); eAmount += 1; keepInBounds(); checkCollision(); break;  
 case KeyEvent.*VK\_RIGHT*: amount += 1; character.setLocation(character.getX()+100,character.getY());herorec.setLocation(character.getX()+100,character.getY()); ai(amount, eAmount); eAmount += 1; keepInBounds(); checkCollision(); break;  
// case KeyEvent.VK\_SPACE: shoot += 1; amount +=1; ai(amount, eAmount); eAmount += 1; addArrow(); arrows.add(arrow); shootArrow(); break;  
 }  
 }  
 });  
  
 this.setVisible(true);  
  
 }  
 private void keepInBounds(){  
 if(character.getX()<0)  
 {  
 character.setLocation(50, character.getY());  
 }  
 if (character.getY()<0)  
 {  
 character.setLocation(character.getX(), 15);  
 }  
 }  
  
 private void createcharacter(){  
 character = new JLabel("");  
 character.setSize(new Dimension(100,100));  
 ImageIcon theImage = new ImageIcon(this.getClass().getResource("/resources/dungeon/FarmerMan.png"));  
 character.setIcon(theImage);  
 character.setLocation(350,0);  
 character.setVisible(true);  
 this.add(character);  
 }  
 private void addGoat(int ex, int why){  
 if(ex + 100 < character.getX() && ex -100 > character.getX())  
 {  
 ex += 100;  
 }  
 if(why + 100 < character.getY() && why - 100 > character.getY())  
 {  
 why +=100;  
 }  
 goaty = new Goat(ex, why, 200, 200, "/resources/dungeon/DerpyGoat1.png");  
 goaty.setVisible(true);  
 this.add(goaty,0);  
 }  
 private Rectangle createGoat(){  
 grec = new Rectangle(goaty.getX(),goaty.getY(),goaty.getWidth(), goaty.getHeight());  
 return grec;  
 }  
 private Pig addPig(int ex, int why) {  
 if(ex + 100 < character.getX() && ex -100 > character.getX())  
 {  
 ex += 100;  
 }  
 if(why + 100 < character.getY() && why - 100 > character.getY())  
 {  
 why +=100;  
 }  
 piggy = new Pig(ex, why, 200, 200, "/resources/dungeon/Pig2.png");  
 piggy.setVisible(true);  
 this.add(piggy, 0);  
 return piggy;  
 }  
 private Rectangle createPig() {  
 prec = new Rectangle(piggy.getX(), piggy.getY(), piggy.getWidth(), piggy.getHeight());  
 return prec;  
 }  
 private Chicken addChicken(int ex, int why) {  
 if(ex + 100 < character.getX() && ex -100 > character.getX())  
 {  
 ex += 100;  
 }  
 if(why + 100 < character.getY() && why - 100 > character.getY())  
 {  
 why +=100;  
 }  
 chicky = new Chicken(ex, why, 150, 150, "/resources/dungeon/Chicken.png");  
 chicky.setVisible(true);  
 this.add(chicky, 0);  
 return chicky;  
 }  
 private Rectangle createChicken() {  
 crec = new Rectangle(chicky.getX(), chicky.getY(), chicky.getWidth(), chicky.getHeight());  
 return crec;  
 }  
  
 private Rectangle enemy(String i){  
 switch (i){  
 case "Goat":  
 rec = createGoat();  
 break;  
 case "Pig":  
 rec = createPig();  
 break;  
 case "Chicken":  
 rec = createChicken();  
 break;  
 }  
 return rec;  
 }  
 private void ai(int amount, int eAmount) {  
 if (amount != eAmount) {  
 if (*abs*(goaty.getX() - character.getX()) > *abs*(goaty.getY() - character.getY())) {  
 if (character.getX() - goaty.getX() > 0) {  
 goaty.setLocation(goaty.getX() + 100, goaty.getY());  
 grec.setLocation(goaty.getX() + 100, goaty.getY());  
 } else if (character.getX() - goaty.getX() < 0) {  
 goaty.setLocation(goaty.getX() - 100, goaty.getY());  
 grec.setLocation(goaty.getX() - 100, goaty.getY());  
 } else {  
 System.*out*.println("Error");  
 }  
 } else if (*abs*(goaty.getX() - character.getX()) < *abs*(goaty.getY() - character.getY())) {  
 if (character.getY() - goaty.getY() > 0) {  
 goaty.setLocation(goaty.getX(), goaty.getY() + 100);  
 grec.setLocation(goaty.getX(), goaty.getY() + 100);  
 } else if (character.getY() - goaty.getY() < 0) {  
 goaty.setLocation(goaty.getX(), goaty.getY() - 100);  
 grec.setLocation(goaty.getX(), goaty.getY() - 100);  
 } else {  
 System.*out*.println("Error");  
 }  
 }  
 if (*abs*(piggy.getX() - character.getX()) > *abs*(piggy.getY() - character.getY())) {  
 if (character.getX() - piggy.getX() > 0) {  
 piggy.setLocation(piggy.getX() + 100, piggy.getY());  
 prec.setLocation(piggy.getX() + 100, piggy.getY());  
 } else if (character.getX() - piggy.getX() < 0) {  
 piggy.setLocation(piggy.getX() - 100, piggy.getY());  
 prec.setLocation(piggy.getX() - 100, piggy.getY());  
 } else {  
 System.*out*.println("Error");  
 }  
 } else if (*abs*(piggy.getX() - character.getX()) < *abs*(piggy.getY() - character.getY())) {  
 if (character.getY() - piggy.getY() > 0) {  
 piggy.setLocation(piggy.getX(), piggy.getY() + 100);  
 prec.setLocation(piggy.getX(), piggy.getY() + 100);  
 } else if (character.getY() - piggy.getY() < 0) {  
 piggy.setLocation(piggy.getX(), piggy.getY() - 100);  
 prec.setLocation(piggy.getX(), piggy.getY() - 100);  
 }  
 }  
 if (*abs*(chicky.getX() - character.getX()) > *abs*(chicky.getY() - character.getY())) {  
 if (character.getX() - chicky.getX() > 0) {  
 chicky.setLocation(chicky.getX() + 100, chicky.getY());  
 crec.setLocation(chicky.getX() + 100, chicky.getY());  
 } else if (character.getX() - chicky.getX() < 0) {  
 chicky.setLocation(chicky.getX() - 100, chicky.getY());  
 crec.setLocation(chicky.getX() - 100, chicky.getY());  
 } else {  
 System.*out*.println("Error");  
 }  
 } else if (*abs*(chicky.getX() - character.getX()) < *abs*(chicky.getY() - character.getY())) {  
 if (character.getY() - chicky.getY() > 0) {  
 chicky.setLocation(chicky.getX(), chicky.getY() + 100);  
 crec.setLocation(chicky.getX(), chicky.getY() + 100);  
 } else if (character.getY() - chicky.getY() < 0) {  
 chicky.setLocation(chicky.getX(), chicky.getY() - 100);  
 crec.setLocation(chicky.getX(), chicky.getY() - 100);  
 }  
 }  
 }  
 }  
  
 private Arrow addArrow(){  
 arrow = new Arrow(character.getX(), character.getY(), 50,50, "/resources/dungeon/ArrowWithFletching.png");  
 arrow.setLocation(character.getX(),character.getY());  
 arrow.setHorizontalAlignment(SwingConstants.*CENTER*);  
 arrow.setVisible(true);  
 this.add(arrow);  
  
 return arrow;  
 }  
  
 private Grid addGrid(){  
 grid = new Grid(0,0,900,900, "/resources/dungeon/GridForBlake.png");  
 grid.setVisible(true);  
 this.add(grid);  
  
 return grid;  
 }  
 private void shootArrow() {  
 for (Arrow a : arrows) {  
 arec = new Rectangle(a.getX(), a.getY(), a.getWidth(), a.getHeight());  
 if (a.getCount() >= 1000) {  
 a.setVisible(false);  
 arrows.remove(a);  
 } else {  
 while (a.getCount() < 1000) {  
 a.setLocation(a.getX() + 1, a.getY());  
 a.setCount(a.getCount() + 1);  
 arec.setLocation(a.getX(), a.getY());  
 if (arec.intersects(grec)) {  
 remove(goaty);  
 dead += 1;  
 } else if (arec.intersects(prec)) {  
 remove(piggy);  
 dead += 1;  
 } else if (arec.intersects(crec)) {  
 remove(chicky);  
 dead += 1;  
 }  
 }  
 arrows.remove(a);  
 remove(a);  
 this.repaint();  
 }  
 }  
 }  
 private void checkCollision()  
 {  
  
 if(grec.intersects(herorec))  
 {  
 System.*out*.println("Hello");  
 dispose();  
 System.*out*.println("It took " + amount + " moves for the animals to get you");  
 HighScore();  
 new Menu();  
 }  
 else if(prec.intersects(herorec))  
 {  
 dispose();  
 System.*out*.println("It took " + amount + " moves for the animals to get you");  
 HighScore();  
 new Menu();  
 }  
 else if (crec.intersects(herorec))  
 {  
 dispose();  
 System.*out*.println("It took " + amount + " moves for the animals to get you");  
 HighScore();  
 new Menu();  
 }  
 }  
 public void HighScore()  
 {  
 player.highScoreJEFF high = new player.highScoreJEFF();  
 boolean newHigh = high.readJEFFHighScore()<=amount;  
 if(newHigh)  
 {  
 high.writeJEFFInvaders(amount);  
 JOptionPane.*showMessageDialog*(null, "New High Score Of: " + amount);  
 }  
 else  
 {  
 JOptionPane.*showMessageDialog*(null, "Your Score was " + amount);  
 }  
  
 }  
  
 // public void act(){  
// shootArrow();  
// checkCollision();  
// }  
//  
// public class MyTimerTask extends TimerTask {  
// @Override  
// public void run(){  
// act();  
// }  
// }  
  
}

package dungeonQuest;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
*/\*\* Rectangle Supplier Class   
 \* Author: David D. Riley  
 \* Date: April, 2004  
 \*/*public class Rectangle extends JComponent {  
  
 */\*\* post: getX() == x and getY() == y  
 \* and getWidth() == w and getHeight() == h  
 \* and getBackground() == Color.black  
 \*/* public Rectangle(int x, int y, int w, int h) {  
 super();  
 setBounds(x, y, w, h);  
 setBackground(Color.*black*);  
 }  
  
 */\*\* post: this method draws a filled Rectangle  
 \* and the upper left corner is (getX(), getY())   
 \* and the rectangle's dimensions are getWidth() and getHeight()  
 \* and the rectangle's color is getBackground()  
 \*/* public void paint(Graphics g) {  
 g.setColor( getBackground() );  
 g.fillRect(0, 0, getWidth()-1, getHeight()-1);  
 paintChildren(g);  
 }  
  
}

package dungeonQuest;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Pig extends JLabel {  
 public Pig(int x, int y, int w, int h, String i)  
 {  
 super("");  
 setBounds(x, y, w, h);  
 setSize(new Dimension(w, h));  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 }  
}

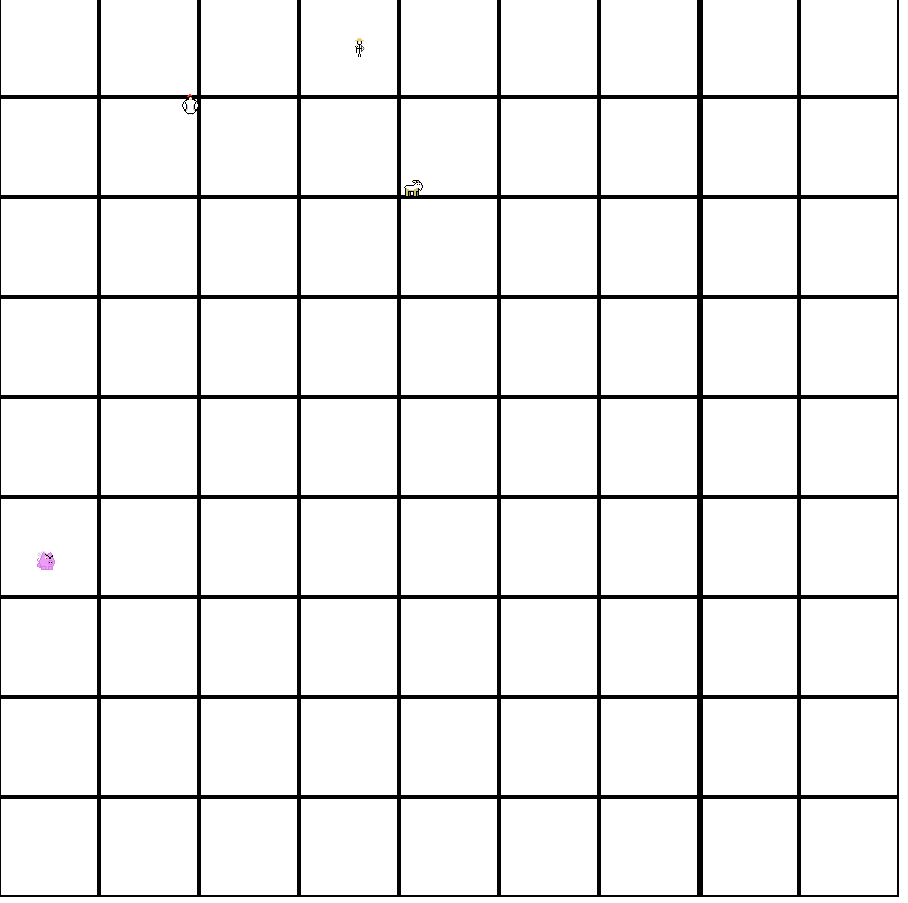
package dungeonQuest;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Grid extends JLabel{  
 public Grid(int x, int y, int w, int h, String i) {  
 super("");  
 setBounds(x, y, w, h);  
 setSize(new Dimension(w, h));  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 }  
}

package dungeonQuest;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Goat extends JLabel{  
 public Goat(int x, int y, int w, int h, String i)  
 {  
 super("");  
 setBounds(x, y, w, h);  
 setSize(new Dimension(w, h));  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 }  
}

package dungeonQuest;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Chicken extends JLabel {  
 public Chicken( int x, int y, int w, int h, String i){  
 super("");  
 setBounds(x, y, w, h);  
 setSize(new Dimension(w, h));  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 }  
}

package dungeonQuest;  
  
import javax.swing.\*;  
  
public class character extends JLabel{  
 protected int x, y, shoot, count, tcount;  
 public character(int x, int y, int w, int h, String i){  
 super("");  
 setBounds(x,y,w,h);  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 setVisible(true);  
 shoot = 15;  
 System.*out*.println(count);  
 }  
 public void moved(int x, int y, int count) {  
 this.setLocation(x, y);  
 tcount += 1;  
 if (count == tcount) {  
 System.*out*.println("done");  
 }  
  
 }  
  
  
}

package dungeonQuest;  
  
import resources.\*;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Arrow extends JLabel{  
  
 private int count;  
  
 private JLabel a;  
  
 public Arrow(int x, int y, int w, int h, String i){  
 super("");  
 count = 0;  
 setBounds(x, y, w, h);  
 setSize(new Dimension(w,h));  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
 setVisible(true);  
 }  
  
 public int getCount(){  
 return count;  
 }  
 public void setCount(int x){  
 count = x;  
 }  
  
 //public int shoot(int x)  
 //{  
 // this.setLocation(x,this.getY());  
 // x += 10;  
 // return x;  
  
 //}  
}



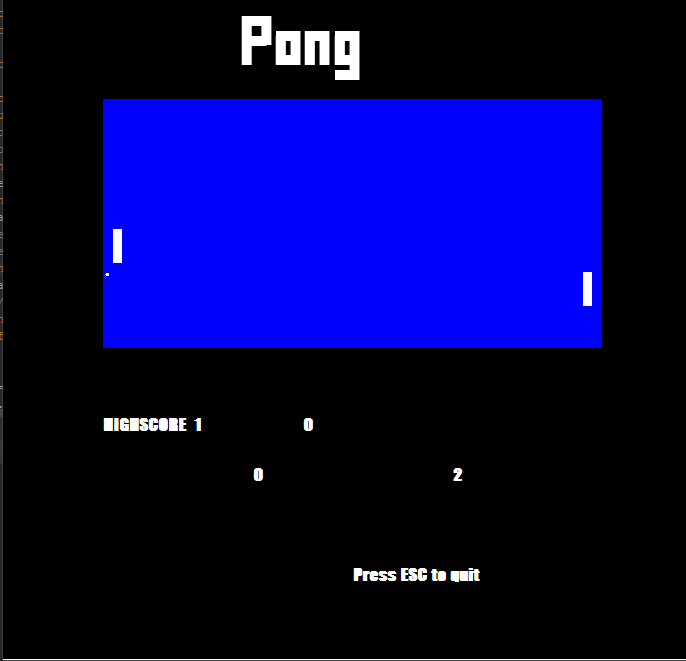
=====Garrett’s Code=====

package pong;  
import javax.swing.\*;  
import java.util.ArrayList;  
import java.util.Random;  
import java.util.Timer;  
import java.util.TimerTask;  
import java.awt.event.\*;  
import java.awt.\*;  
import resources.\*;  
import menu.Menu;  
import player.highScorePong;  
  
  
*/\*\*  
 \* Pong game board  
 \* Made By Garrett  
 \*/*public class PongTable extends JFrame {  
 private int score=0;  
 private int fps = 50;  
 private Timer t = new Timer();  
 private GameRectangle rec;  
 private Pong ball;  
 private Paddle paddle;  
 private Paddle paddle2;  
 private int counter=0;  
 private int counter2=0;  
 private int high;  
 private boolean up;  
 private boolean down;  
 private boolean s;  
 private boolean w;  
 private boolean decision;  
 private boolean aim;  
 private JLabel J;  
 private JLabel J2;  
 private JLabel test;  
 private JLabel test2;  
 private JLabel test3;  
 private highScorePong whatever;  
  
 private JLabel pongImage;  
  
 public PongTable() {  
 super("Pong");  
 up=false;  
 down=false;  
 this.getContentPane().setBackground(Color.*WHITE*);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 this.setBounds(500, 200, 700, 700);  
 GameRectangle table = new GameRectangle(100, 100, 500, 250);  
 rec=new GameRectangle(0,0,700,700);  
 rec.setBackground(Color.*BLACK*);  
 this.add(rec,0);  
 table.setBackground(Color.*BLUE*);  
 //allows single or local multiplayer  
 int ting=JOptionPane.*showConfirmDialog*(null,"Are You playing with another human?(Local)","No is playing a computer",JOptionPane.*YES\_NO\_OPTION*);  
 if(ting==0) {  
 decision=true;  
 }  
 J=new JLabel(Integer.*toString*(counter));  
 J.setFont(new Font("Impact", Font.*BOLD*,16));  
 J.setForeground(Color.*WHITE*);  
 J.setBounds(250,450,50,50);  
 J2=new JLabel(Integer.*toString*(counter));  
 J2.setFont(new Font("Impact", Font.*BOLD*,16));  
 J2.setForeground(Color.*WHITE*);  
 J2.setBounds(450,450,50,50);  
 test=new JLabel("Press ESC to quit");  
 test.setFont(new Font("Impact", Font.*BOLD*,16));  
 test.setForeground(Color.*WHITE*);  
 test.setBounds(350,550,200,50);  
 rec.add(test);  
 if(!decision) {  
 whatever=new highScorePong();  
 test2 = new JLabel(Integer.*toString*(score));  
 test2.setFont(new Font("Impact", Font.*BOLD*, 16));  
 test2.setForeground(Color.*WHITE*);  
 test2.setBounds(300, 400, 50, 50);  
 rec.add(test2);  
 test3 = new JLabel("HIGHSCORE "+Integer.*toString*(high=whatever.readPongHighScore()));  
 test3.setFont(new Font("Impact", Font.*BOLD*, 16));  
 test3.setForeground(Color.*WHITE*);  
 test3.setBounds(100, 400, 150, 50);  
 rec.add(test3);  
  
 }  
 rec.add(J);  
 rec.add(J2);  
 pongImage = new JLabel("");  
 pongImage.setSize(new Dimension(400,100));  
 pongImage.setIcon(new ImageIcon(this.getClass().getResource("/resources/images/Pong White.png")));  
 pongImage.setFont(new Font("Comic Sans MS", Font.*BOLD*, 36));  
 pongImage.setLocation(100,0);  
 pongImage.setHorizontalAlignment(SwingConstants.*CENTER*);  
 pongImage.setVisible(true);  
  
  
 rec.add(pongImage);  
  
  
 rec.add(table, 0);  
 ball = new Pong(5, 5, 500, 250);  
 ball.setBackground(Color.*WHITE*);  
 paddle = new Paddle(10, 125, 10, 35);  
 paddle.setBackground(Color.*WHITE*);  
 paddle2=new Paddle(480,125,10,35);  
 paddle2.setBackground(Color.*WHITE*);  
 table.add(paddle2);  
 table.add(paddle, 0);  
 table.add(ball, 0);  
 setFocusable(true);  
 addKeyListener(new KeyAdapter() {  
 //code for when a key is pressed  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch (e.getKeyCode()) {  
 case KeyEvent.*VK\_UP*:  
 up=true;  
 if (!up  
 ) {  
  
  
 paddle.moveUP();  
 //code for moving paddle up  
 }  
 break;  
 case KeyEvent.*VK\_W*:  
 {  
 w=true;  
 if((!w)&&(decision))  
 {  
 paddle2.moveUP();  
 }  
 break;  
 }  
  
 case KeyEvent.*VK\_DOWN*:  
 down=true;  
 if(!down) {  
 paddle.moveDown();  
 }  
 //code for moving paddle down  
 break;  
 case KeyEvent.*VK\_S*:  
 {  
 s=true;  
 if((!s)&&(decision))  
 {  
 paddle2.moveDown();  
 }  
 break;  
 }  
  
 case KeyEvent.*VK\_ESCAPE*:  
 {  
 new Menu();  
 dispose();  
 }  
 }  
 }  
 //code for when key is lifted  
 @Override  
 public void keyReleased(KeyEvent e) {  
 switch (e.getKeyCode()) {  
 case KeyEvent.*VK\_UP*:  
 up=false;  
 //code for moving paddle up  
 break;  
  
 case KeyEvent.*VK\_DOWN*:  
 down=false;  
 //code for moving paddle down  
 break;  
 case KeyEvent.*VK\_W*:  
 w=false;  
 break;  
 case KeyEvent.*VK\_S*:  
 s=false;  
 break;  
 }  
 }  
 });  
 t.schedule(new myTimerTask(), 0, 1000 / fps);  
 setVisible(true);  
 }  
  
 public class myTimerTask extends TimerTask {  
 @Override  
 public void run() {  
  
 if(!decision) {  
 test2.setText(Integer.*toString*(score));  
  
 }  
 //allows gameplay if ball is not out of bounds  
 if((!ball.checkX1())&&(!ball.checkX2())) {  
 ball.move();  
 if (up) {  
 paddle.moveUP();  
  
 }  
 if (down) {  
 paddle.moveDown();  
 }  
 if((w)&&(decision))  
 {  
 paddle2.moveUP();  
 }  
 if((s)&&(decision))  
 {  
 paddle2.moveDown();  
 }  
  
 if(!decision) {  
 if(aim){  
 paddle2.setLocation(paddle2.getX(),ball.getY());  
  
 }  
 else  
 {  
 paddle2.setLocation(paddle2.getX(),ball.getY()+10);  
  
 }  
 }  
 if(collide(paddle.left(), paddle.right(), paddle.bottom(), paddle.top())) {  
 counter++;  
 if(counter%3==0)  
 {  
 ball.changeSpeed();  
 }  
 aim=paddle2.computer();  
  
  
 }  
 if(collide(paddle2.left(), paddle2.right(), paddle2.bottom(), paddle2.top())) {  
 counter2++;  
 }  
  
  
 J.setText(Integer.*toString*(counter));  
 J2.setText(Integer.*toString*(counter2));  
 }  
 else {  
 if(!ball.checkX2()) {  
 counter=0;  
 score=0;  
 }  
 if(!ball.checkX1()) {  
 counter2=0;  
 if(!decision) {  
 paddle2.changeValue();  
 score++;  
 }  
 }  
 //makes new highscore  
 if(!decision)  
 if(high<score) {  
 whatever.writePongInvaders(score);  
 }  
 //resets the bored  
 aim=paddle2.computer();  
 ball.reset();  
 paddle2.reset();  
 paddle.reset();  
  
  
 }  
 }  
 }  
  
//checks for paddle ball collision  
 public boolean collide(int l,int r,int b,int t)  
 {  
 int right=ball.right();  
 int bottom=ball.bottom();  
 int left=ball.left();  
 int top=ball.top();  
 int L2=l;  
 int R2=r;  
 int B2=b;  
 int T2=t;  
 if((right>L2)&&(right<R2)&&(bottom>T2)&&(bottom<B2)||  
 (right>L2)&&(right<R2)&&(top>T2)&&(top<B2)||  
 (left>L2)&&(left<R2)&&(bottom>T2)&&(bottom<B2)||  
 (left>L2)&&(left<R2)&&(top>T2)&&(top<B2)) {  
 ball.changeSpeedD();  
 return true;  
 }  
 else  
 {  
 return false; }  
 }  
  
}

package pong;  
import resources.GameOval;  
import java.util.Random;  
*/\*\*  
 \* Ball class for pong  
 \*  
 \** ***@Garrett*** *\** ***@version*** *(a version number or a date)  
 \*/*public class Pong extends GameOval {  
 private int width2;  
 private int width;  
 private int height;  
 private int height2;  
 private int speedY;  
 private int speedX;  
  
 public Pong(int w, int h, int w2, int h2) {  
 super(250, 125, w, h);  
 width2 = w2;  
 height2 = h2;  
 width = w;  
 height = h;  
 speedY = 4;  
 speedX = 4;  
 }  
  
 //allows ball movement  
 public void move() {  
 int x = this.getX();  
 int y = this.getY();  
 this.setLocation(x + speedX, y + speedY);  
 if ((this.getX() > (width2 - width)) || (this.getX() < 0)) {  
 speedX \*= -1;  
 }  
 if ((this.getY() > (height2 - height)) || (this.getY() < 0)) {  
 speedY \*= -1;  
 }  
  
 }  
  
 public int left() {  
 return getX();  
 }  
  
 public int right() {  
 return getX() + width;  
 }  
  
 public int top() {  
 return getY();  
 }  
  
 public int bottom() {  
 return getY() + height;  
 }  
  
 //allow ball to bounce off paddle  
 public void changeSpeedD() {  
 speedX \*= -1;  
 }  
  
 //allows ball to change speed  
 public void changeSpeed() {  
 Random gen = new Random();  
 speedX += gen.nextInt(3);  
 speedY += gen.nextInt(3);  
 }  
  
 //checks right side for ball out of bounds  
 public boolean checkX1()  
 {  
 if((this.getX()<5)) {  
 return true;  
 }  
 else  
 return false;  
  
 }  
 //checks left side for ball out of bounds  
 public boolean checkX2()  
 {  
 if(this.getX()>495) {  
 return true;  
 }  
 else  
 return false;  
  
 }  
 //allows ball reset  
 public void reset()  
 {  
 this.setLocation(250,125);  
 speedX=4;  
 speedY=4;  
 }  
}

package pong;  
import javax.swing.\*;  
import java.awt.\*;  
public class Photo extends JLabel{  
 public Photo(int x, int y, int g, int b,String i){  
 super("");  
 setVisible(true);  
 setSize(new Dimension(g,b));  
 setLocation(x,y);  
 setIcon(new ImageIcon(this.getClass().getResource(i)));  
  
 }  
}

package pong;  
import resources.\*;  
  
import java.util.Random;  
  
*/\*\*  
 \* Paddle class for Pong  
 \*  
 \* Garrett  
 \** ***@version*** *(a version number or a date)  
 \*/*public class Paddle extends GameRectangle{  
 private int W;  
 private int H;  
 private int X;  
 private int Y;  
 private int value;  
 public Paddle(int x, int y, int w, int h) {  
 super(x, y, w, h);  
 X=x;  
 Y=y;  
 W = w;  
 H = h;  
 setDefaultValue();  
 }  
 //makes basic computer bot  
 public boolean computer()  
 {  
 boolean ting=true;  
 Random gen=new Random();  
 int temp=gen.nextInt(100)+1;  
 if(value>temp){  
 ting=false;  
 }  
 return ting;  
 }  
  
 public void moveUP() {  
 this.setLocation(getX(), getY() - 5);  
 }  
  
 public void moveDown() {  
 this.setLocation(getX(), getY() + 5);  
 }  
  
 public int left() {  
 return getX();  
 }  
  
 public int right() {  
 return getX() + W;  
 }  
  
 public int top() {  
 return getY();  
 }  
 //reset computer difficulty  
 public void setDefaultValue()  
 {  
 value=50;  
 }  
 //make computer get more difficult  
 public void changeValue()  
 {  
 if(value>1) {  
 value -= 2;  
 }  
 else  
 {  
 value=1;  
 }  
 }  
  
 public int bottom() {  
 return getY() + H;  
 }  
 //resets paddles  
 public void reset()  
 {  
 this.setLocation(X,Y);  
 }  
 //was used for bug fixing  
 public int ReturnValue()  
 {  
 return value;  
 }  
}



=====Jake’s Code=====

NOTE: Jake was also in charge of all artwork for the game. All images in the Menu, Jeff’s Quest, and Space Invaders were created by him.

package player;  
  
import java.io.\*;  
  
/\*  
\* Made by Jake  
\* \*/  
  
public class highScoreJEFF {  
  
 int previousScore;  
  
 public int readJEFFHighScore(){  
 try {  
 //Create an object that opens a file for writing data  
 //FileInputStream fileIn = new FileInputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataInputStream dataIn = new DataInputStream(new FileInputStream("highScoresJEFF.txt"));  
 previousScore = dataIn.readInt();  
 dataIn.close();  
  
 }catch(IOException e){  
 e.printStackTrace();  
 }  
 return previousScore;  
 }  
  
 public void writeJEFFInvaders(int n){  
 try{  
 //Create an object that opens a file for writing data  
 //FileOutputStream fileOut = new FileOutputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataOutputStream dataOut = new DataOutputStream(new FileOutputStream("highScoresJEFF.txt"));  
 if(n>previousScore){  
 dataOut.writeInt(n);  
 dataOut.close();  
 }  
 else{  
 dataOut.close();  
 }  
  
 }catch(IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

package player;  
  
import java.io.\*;  
  
*/\*\*  
 \* Made by Jake  
 \*/*public class highScorePong {  
  
 int previousScore;  
  
 public int readPongHighScore(){  
 try {  
 //Create an object that opens a file for writing data  
 //FileInputStream fileIn = new FileInputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataInputStream dataIn = new DataInputStream(new FileInputStream("highScoresPong.txt"));  
 previousScore = dataIn.readInt();  
 dataIn.close();  
  
 }catch(IOException e){  
 e.printStackTrace();  
 }  
 return previousScore;  
 }  
  
 public void writePongInvaders(int n){  
 try{  
 //Create an object that opens a file for writing data  
 //FileOutputStream fileOut = new FileOutputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataOutputStream dataOut = new DataOutputStream(new FileOutputStream("highScoresPong.txt"));  
 if(n>previousScore){  
 dataOut.writeInt(n);  
 dataOut.close();  
 }  
 else{  
 dataOut.close();  
 }  
  
 }catch(IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

package player;  
  
import java.io.\*;  
  
*/\*\*  
 \* Made by Jake  
 \*/*public class highScoreSpaceInvaders {  
  
 int previousScore;  
  
 public int readSpaceInvadersHighScore(){  
 try {  
 //Create an object that opens a file for writing data  
 //FileInputStream fileIn = new FileInputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataInputStream dataIn = new DataInputStream(new FileInputStream("highScoreSpaceInvaders.txt"));  
 previousScore = dataIn.readInt();  
 dataIn.close();  
  
 }catch(IOException e){  
 e.printStackTrace();  
 }  
 return previousScore;  
 }  
  
 public void writeHighScoreSpaceInvaders(int n){  
 try{  
 //Create an object that opens a file for writing data  
 //FileOutputStream fileOut = new FileOutputStream("info.txt");  
 //Create a stream object that connects to the file  
 DataOutputStream dataOut = new DataOutputStream(new FileOutputStream("highScoreSpaceInvaders.txt"));  
 if(n>previousScore){  
 dataOut.writeInt(n);  
 dataOut.close();  
 }  
 else{  
 dataOut.close();  
 }  
  
 }catch(IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

import menu.\*;  
  
*/\*\*  
 \* Created by Jake on 12/18/2017.  
 \*/*public class Main {  
  
 public static void main(String[]args){  
  
 new Menu();  
  
 }  
  
}

