Concurrent Programming project

By:

Khaled Mohammed 201801275

Abdullah Nabil 201801353

Abdulrahman Alaa 201801336

Marwan Abdelaziz 201801411

To: Dr. Mohammed El kholy

Eng. Nada & Eng. Sahar

Shooting Game

Overview

The project consist of 3 classes: Ball, BounicingBalls, and ship.

The Ball class

The class extends Thread so we can create a thread for each ball.

```
Ball (int framewidth , int frameheight ,Color c ,int speed)
{
    this.c=c;
    fw =framewidth;
    fh =frameheight;
    x= ran.nextInt(fw-diameter);
    y=ran.nextInt(fh-diameter);
    this.speed=speed;
    //speed=ran.nextInt(10);
    up = ran.nextBoolean();
    right = ran.nextBoolean();

}
```

The constructor takes the frame width and height that we create in the BouncinBallas class and set each ball colour and speed as paramaters and creates random movement along the x and y axis inside the frame.

The draw method fills each ball and sets it to a colour.

```
public void run()
{
    while(true)
    {
    if(right) x+=speed;
    else x-=speed;

    if(up) y-=speed;
    else y +=speed;

    if(x<0) right=true;
    if(x>(fw-diameter)) right = false;

    if(y<25) up=false;
    if(y>(fh-diameter)) up = true;
        try {
            Thread.sleep(15);
        } catch (InterruptedException ex) {
            Logger.getLogger(Ball.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

Run method sets the edges for the ball movement and the reflection behaviour.

The ship class

This class creates the gun and bullet and score behaviour.

The constructor takes array of object ball to manage the behaviour when the bullet hits the ball, the attributes set the starting gun position.

```
public void draw(Graphics g){
    g.setColor(Color.blue);
    g.fillRect(x, y, 40, 10);
    g.fillRect(x+18, y-7, 4, 4);

    if(shot){
        g.setColor(Color.WHITE);
        g.fillRect(bullet.x, bullet.y, bullet.width, bullet.height);
    }
}
```

Draw sets the gun colour and the bullet colour.

The move method sets the behaviour of the gun movement along the x axis.

```
void hit ()
 {
     int i;

     for( i =0; i<ball.length;i++)
      if (bullet.intersects( ball[i].x,  ball[i].y,  ball[i].diameter, ball[i].diameter))
      {
          if(i==3)
          {
              score=0;
              s=Integer.toString(score);
               ball[i].y=ran.nextInt(500);
               ball[i].x=ran.nextInt(800);
          }
          else
          {
      ball[i].y=ran.nextInt(500);
      ball[i].x=ran.nextInt(500);
      score ++;
      s=Integer.toString(score);
          }
      }
```

Hit method has bullet.intersects which takes ball position and size as attributes.

If method to set the red ball which is the 4th array if hit sets score to 0 as a bomb ball and creates the thread of the ball again in a random position.Else any other ball adds 1 to the score.

```
public void keyPressed(KeyEvent e) {

    int key = e.getKeyCode();

    if (key == KeyEvent.VK_LEFT)
        dx--;
        //setdx(-1);
    if (key == KeyEvent.VK_SPACE)
    {
        if(bullet==null){
            readytofire=true;
        }
    if(readytofire){
        by=y-7;
        bx=x+18;
        bullet=new Rectangle(bx,by,3,5);
        shot=true;
    }
    }
    if (key == KeyEvent.VK_RIGHT)
         dx++;
```

keyPressed method creates object from keyevent library and when the designed button pressed moves the gun to the left or right and space key sets readytofire true which creates bullet that comes out of the middle of the gun.

```
public void keyReleased(KeyEvent e) {
    int key = e.getKeyCode();

    if (key == KeyEvent.VK_LEFT)
    dx=0;
setdx(0);

    if (key == KeyEvent.VK_RIGHT)
    dx=0;
setdx(0);
    if (key == KeyEvent.VK_SPACE){

            readytofire=false;
            if(bullet.y<= 0 ){
                bullet=new Rectangle(0,0,0,0);
                shot=false;
                readytofire=true;
            }

    }
```

KeyReleased methoed sets the dx to 0 so when pressed again the gun moves along the x axis as the user wishes.The space key when released sets readytofire to false so when pressed again the object is created to be shot.
Shoot method moves the bullet along the y axis.

```
public void run(){

    try{
        while(true){

            shoot();
            move();

          Thread.sleep(1);
        }
    }
    catch(Exception e){
        System.err.println(e.getMessage());
    }
}
```

The BouncingBalls class

This class implents runnable and create the frame of the box of the game(width =600 and height =600).

```
BouncingBalls()
{
    setBounds(100,100,framewidth,frameheigt);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setResizable(true);
    setVisible(true);
    addKeyListener(gun);


    ball[0] = new Ball(framewidth,frameheigt,Color.BLACK ,1);
    ball[1] = new Ball(framewidth,frameheigt,Color.BLUE ,2);
    ball[2] = new Ball(framewidth,frameheigt,Color.RED ,3);
    ball[3] = new Ball(framewidth,frameheigt,Color.WHITE ,4);
    ball[4] = new Ball(framewidth,frameheigt,Color.YELLOW ,5);
    ball[5] = new Ball(framewidth,frameheigt,Color.PINK,6);

    thread =new Thread(this);
    thread.run();
}
```

The constructer creates Array of objects from class Ball, runs the thread of this class,and adds attributes to the frame.

Paint method sets the color of the frame and the colour of each ball in the frame and the score.

```
public void run() {

    for(int i=0; i<ball.length;i++)
        ball[i].start();
     t.start();

  while(true)
  {

    //hit();

       repaint();

      try {
          Thread.sleep(20);
      } catch (InterruptedException ex) {
          Logger.getLogger(BouncingBalls.class.getName()).log(Level.SEVERE, null, ex);
      }

  }
```

Run method starts each ball thread.Repaint to keep refreshing to show the movement and sleep to prevent lagging.