Tank Game & Super Rainbow Reef Documentation

Amelie Cameron

Due August 13th, 2017

CSC 413
Professor Souza

Table of Contents

Project Information:
Tank Game & Super Rainbow Reef Game
CSC 413 Summer 2017 Professor Souza
Version 0.1
Amelie Cameron

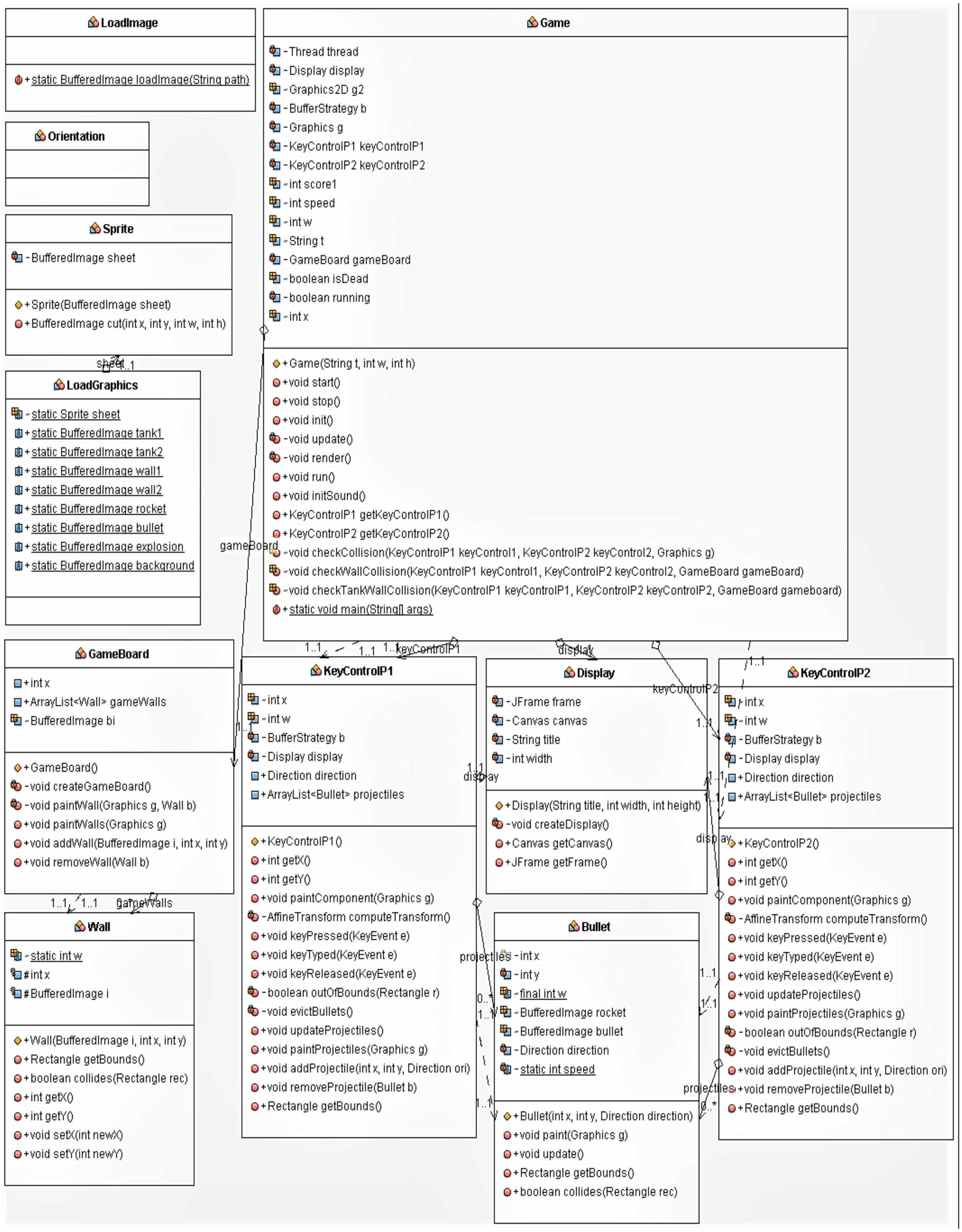
Introduction:
This is a developer's guide for the CSC 413 Summer 2017 Java games. I implemented the Tank Game and Super Rainbow Reef Game. The developer's guide contains information on how to use the games and how they function internally. The Tank Game is a two player shooting game, while the Super Rainbow Reef game is a one player brick breaker game. This guide will showcase the similarities as well as the differences in the code and how the games function.

Scope of Work:
As a starting point for these projects, we were given a zip file for the Plane Game to use as a template. The scope of work for these projects included setting up the hierarchy of classes and writing all necessary packages and subclasses. The two games were tested and documentation created.

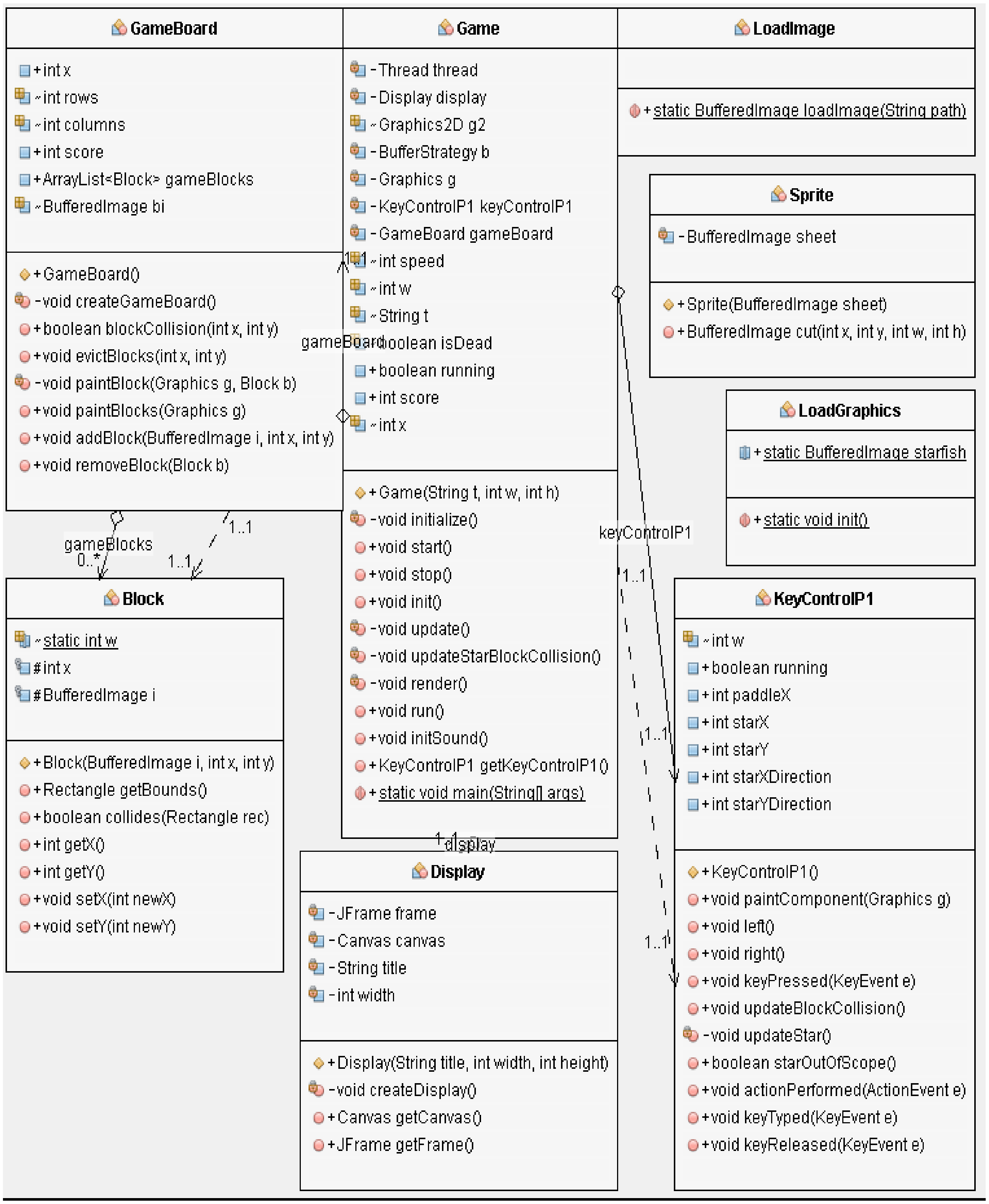
Background/Given Resources:
There was no given source code given for these projects and I have no prior experience programming java games. With that in mind, I studied the Plane Game and watched java game tutorials on YouTube. I included concepts and functions from these tutorials as a basis for the run() function in the Game class and the Graphics package classes. Similarly, I had difficulty with my initSound() function and showing a JOptionPane message in the render() function in the Game class. For these problems, I found solutions on StackOverflow and docs.oracle.com respectively. All source code is accurately cited within the projects.

Assumptions:
Development work was done a Lenovo ThinkPad laptop Intel Core i7-3520M CPU @ 2.90GHz, 8.0 GB RAM, running Windows 10 Pro 64-bit Operating System. NetBeans 8.1 was used as the development environment.

Tank Game Class Diagram: See the following page for the class diagram.


LoadImage
+static BufferedImage loadImage(String path)
Orientation
Sprite
-BufferedImage sheet
+Sprite(BufferedImage sheet)
+BufferedImage cut(int x, int y, int w, int h)
sheet 1..1
LoadGraphics
~static Sprite sheet
+static BufferedImage tank1
+static BufferedImage tank2
+static BufferedImage wall1
+static BufferedImage wall2
+static BufferedImage rocket
+static BufferedImage bullet
+static BufferedImage explosion
+static BufferedImage background
Game
-Thread thread
-Display display
~Graphics2D g2
-BufferStrategy b
-Graphics g
-KeyControlP1 keyControlP1
-KeyControlP2 keyControlP2
~int score1
~int speed
~int w
~String t
-GameBoard gameBoard
~boolean isDead
-boolean running
~int x
+Game(String t, int w, int h)
+void start()
+void stop()
+void init()
-void update()
-void render()
+void run()
+void initSound()
+KeyControlP1 getKeyControlP1()
+KeyControlP2 getKeyControlP2()
~void checkCollision(KeyControlP1 keyControl1, KeyControlP2 keyControl2, Graphics g)
~void checkWallCollision(KeyControlP1 keyControl1, KeyControlP2 keyControl2, GameBoard gameBoard)
~void checkTankWallCollision(KeyControlP1 keyControlP1, KeyControlP2 keyControlP2, GameBoard gameboard)
+static void main(String[] args)
gameBoard
1..1
1..1
1..1 keyControlP1
display
keyControlP2
1..1
GameBoard
+int x
+ArrayList<Wall> gameWalls
~BufferedImage bi
+GameBoard()
-void createGameBoard()
-void paintWall(Graphics g, Wall b)
+void paintWalls(Graphics g)
+void addWall(BufferedImage i, int x, int y)
+void removeWall(Wall b)
1..1 1..1 gameWalls
Wall
~static int w
#int x
#BufferedImage i
+Wall(BufferedImage i, int x, int y)
+Rectangle getBounds()
+boolean collides(Rectangle rec)
+int getX()
+int getY()
+void setX(int newX)
+void setY(int newY)
KeyControlP1
~int x
~int w
-BufferStrategy b
-Display display
+Direction direction
+ArrayList<Bullet> projectiles
+KeyControlP1()
+int getX()
+int getY()
+void paintComponent(Graphics g)
-AffineTransform computeTransform()
+void keyPressed(KeyEvent e)
+void keyTyped(KeyEvent e)
+void keyReleased(KeyEvent e)
-boolean outOfBounds(Rectangle r)
-void evictBullets()
+void updateProjectiles()
+void paintProjectiles(Graphics g)
+void addProjectile(int x, int y, Direction ori)
+void removeProjectile(Bullet b)
+Rectangle getBounds()
1..1 display
projectiles
0..*
1..1
1..1
Display
-JFrame frame
-Canvas canvas
-String title
-int width
+Display(String title, int width, int height)
-void createDisplay()
+Canvas getCanvas()
+JFrame getFrame()
1..1
1..1
display
Bullet
-int x
-int y
~final int w
~BufferedImage rocket
~BufferedImage bullet
-Direction direction
-static int speed
+Bullet(int x, int y, Direction direction)
+void paint(Graphics g)
+void update()
+Rectangle getBounds()
+boolean collides(Rectangle rec)
1..1
1..1
projectiles
0..*
KeyControlP2
~int x
~int w
-BufferStrategy b
-Display display
+Direction direction
+ArrayList<Bullet> projectiles
+KeyControlP2()
+int getX()
+int getY()
+void paintComponent(Graphics g)
-AffineTransform computeTransform()
+void keyPressed(KeyEvent e)
+void keyTyped(KeyEvent e)
+void keyReleased(KeyEvent e)
+void updateProjectiles()
+void paintProjectiles(Graphics g)
-boolean outOfBounds(Rectangle r)
-void evictBullets()
+void addProjectile(int x, int y, Direction ori)
+void removeProjectile(Bullet b)
+Rectangle getBounds()

Super Rainbow Reef Class Diagram:



<u>Descriptions of Classes Shared Between Two Games:</u>

Both games share the basic classes: Game, GameObject, KeyControlP1, LoadImage, LoadGraphics, Sprite, and Display. See "Detailed Class Explanation – Classes Shared Between Two Classes" for a more in depth discussion of each class.

<u>Descriptions of Tank Game Specific Classes:</u>

Game classes that are specific to the Tank Game are as follows: KeyControlP2, Orientation, and the Bullet class.

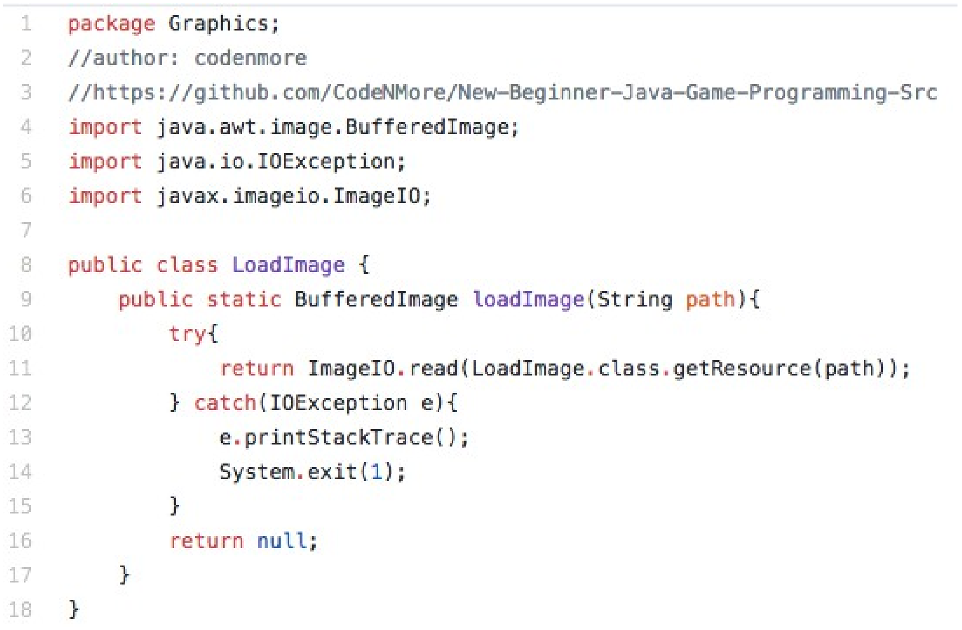
<u>Descriptions of SRR Game Specific Classes:</u>

Game class that is specific to the Super Rainbow Reef Game is the Block class. See page 12.

<u>Detailed Class Explanation – Classes Shared Between Two Classes:</u>

- **<u>LoadImage Class:</u>**

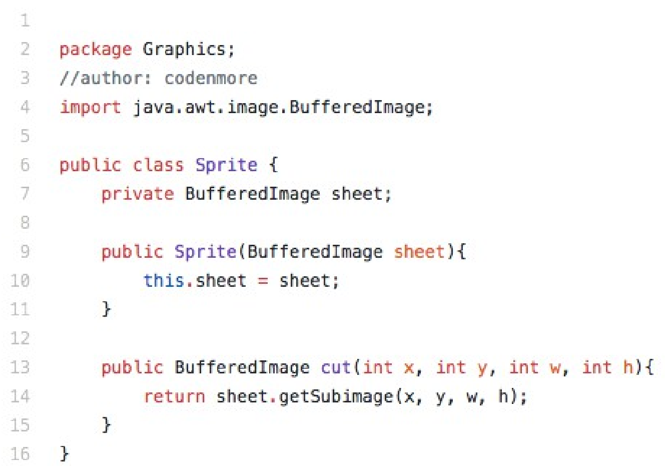
  The loadImage function takes in a String path and returns a buffered image or an error if the file is incorrect or inaccessible. Source code in the Graphics package is from codenmore tutorial.



```java
package Graphics;
//author: codenmore
//https://github.com/CodeNMore/New-Beginner-Java-Game-Programming-Src
import java.awt.image.BufferedImage;
import java.io.IOException;
import javax.imageio.ImageIO;

public class LoadImage {
    public static BufferedImage loadImage(String path){
        try{
            return ImageIO.read(LoadImage.class.getResource(path));
        } catch(IOException e){
            e.printStackTrace();
            System.exit(1);
        }
        return null;
    }
}
```

- **<u>Sprite Class:</u>**

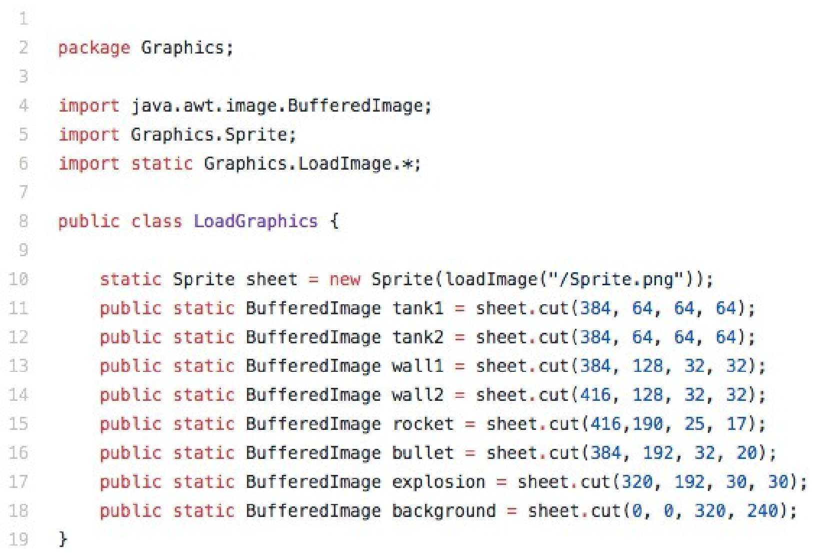
  This class was taken from the codenmore tutorial. It takes the sprite sheet and has a cut function that splits up the images in the sprite sheet based on the (x,y) coordinates of the upper left corner of the image and it's width and height.



```java

package Graphics;
//author: codenmore
import java.awt.image.BufferedImage;

public class Sprite {
    private BufferedImage sheet;

    public Sprite(BufferedImage sheet){
        this.sheet = sheet;
    }

    public BufferedImage cut(int x, int y, int w, int h){
        return sheet.getSubimage(x, y, w, h);
    }
}
```

- **<u>LoadGraphics Class:</u>**

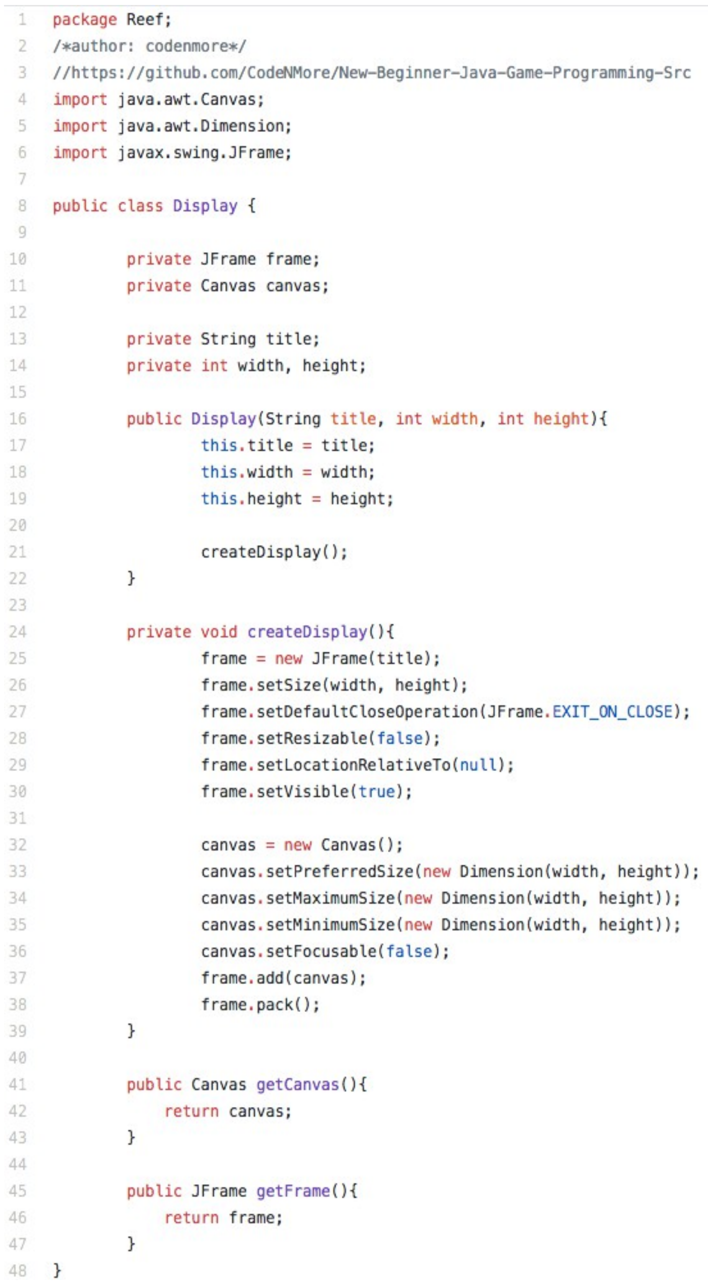
  This class loads the sprite sheet of graphics and cuts it into individual images to print to the game board. The sprite sheet(Sprite.png in Resources,) was created using Texture Packer.



```java

package Graphics;

import java.awt.image.BufferedImage;
import Graphics.Sprite;
import static Graphics.LoadImage.*;

public class LoadGraphics {

    static Sprite sheet = new Sprite(loadImage("/Sprite.png"));
    public static BufferedImage tank1 = sheet.cut(384, 64, 64, 64);
    public static BufferedImage tank2 = sheet.cut(384, 64, 64, 64);
    public static BufferedImage wall1 = sheet.cut(384, 128, 32, 32);
    public static BufferedImage wall2 = sheet.cut(416, 128, 32, 32);
    public static BufferedImage rocket = sheet.cut(416,190, 25, 17);
    public static BufferedImage bullet = sheet.cut(384, 192, 32, 20);
    public static BufferedImage explosion = sheet.cut(320, 192, 30, 30);
    public static BufferedImage background = sheet.cut(0, 0, 320, 240);
}
```

- **Display Class:**

  The display class uses a private JFrame and Canvas. It calls CreateDisplay() which sets the dimensions and properties of the JFrame and Canvas. Then it adds the canvas to the frame.
  * I also added a getJFrame() function in the Display class to help with the scoreboards.

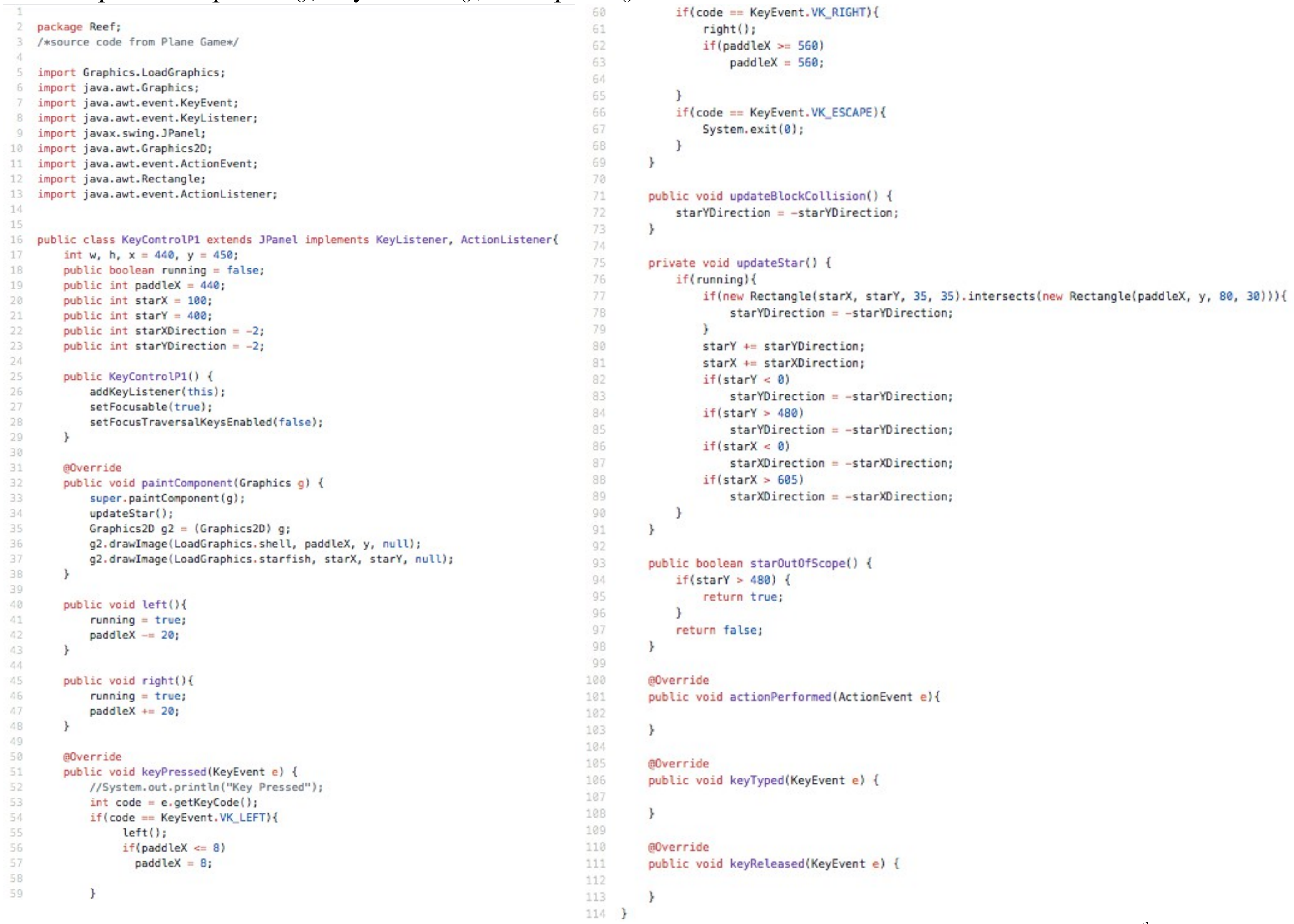


```java
package Reef;
/*author: codenmore*/
//https://github.com/CodeNMore/New-Beginner-Java-Game-Programming-Src
import java.awt.Canvas;
import java.awt.Dimension;
import javax.swing.JFrame;

public class Display {

        private JFrame frame;
        private Canvas canvas;

        private String title;
        private int width, height;

        public Display(String title, int width, int height){
                this.title = title;
                this.width = width;
                this.height = height;

                createDisplay();
        }

        private void createDisplay(){
                frame = new JFrame(title);
                frame.setSize(width, height);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setResizable(false);
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);

                canvas = new Canvas();
                canvas.setPreferredSize(new Dimension(width, height));
                canvas.setMaximumSize(new Dimension(width, height));
                canvas.setMinimumSize(new Dimension(width, height));
                canvas.setFocusable(false);
                frame.add(canvas);
                frame.pack();
        }

        public Canvas getCanvas(){
            return canvas;
        }

        public JFrame getFrame(){
            return frame;
        }
}
```

- **<u>KeyControlP1 Class:</u>**

  For the Tank Game, the KeyControl class creates an array list of projectiles of type Bullet. It also uses an AffineTransform to handle the rotation of objects. The other bullet functions that are implemented are addProjectile(), removeProjectile(), paintProjectiles(), updateProjectiles(), and evictBullets() which handle the bullet array list and their movement in the game. However, these functions are not used in the Super Rainbow Reef game. Functions that are specific to this game are updateBlockCollision(), updateStar(), actionPerformed(), and starOutofScope(). These handle collisions within the brick breaker game.

  In both games, a keyPressed() function is used to take input from the keyboard to control the player's game object. (Tank Game P1: W, A, S, D, Space, Escape/P2:arrow keys, Delete key; Super Rainbow Reef: Left arrow, Right arrow, Escape.) They also both implement the paintComponent(), keyPressed(), and update() functions.



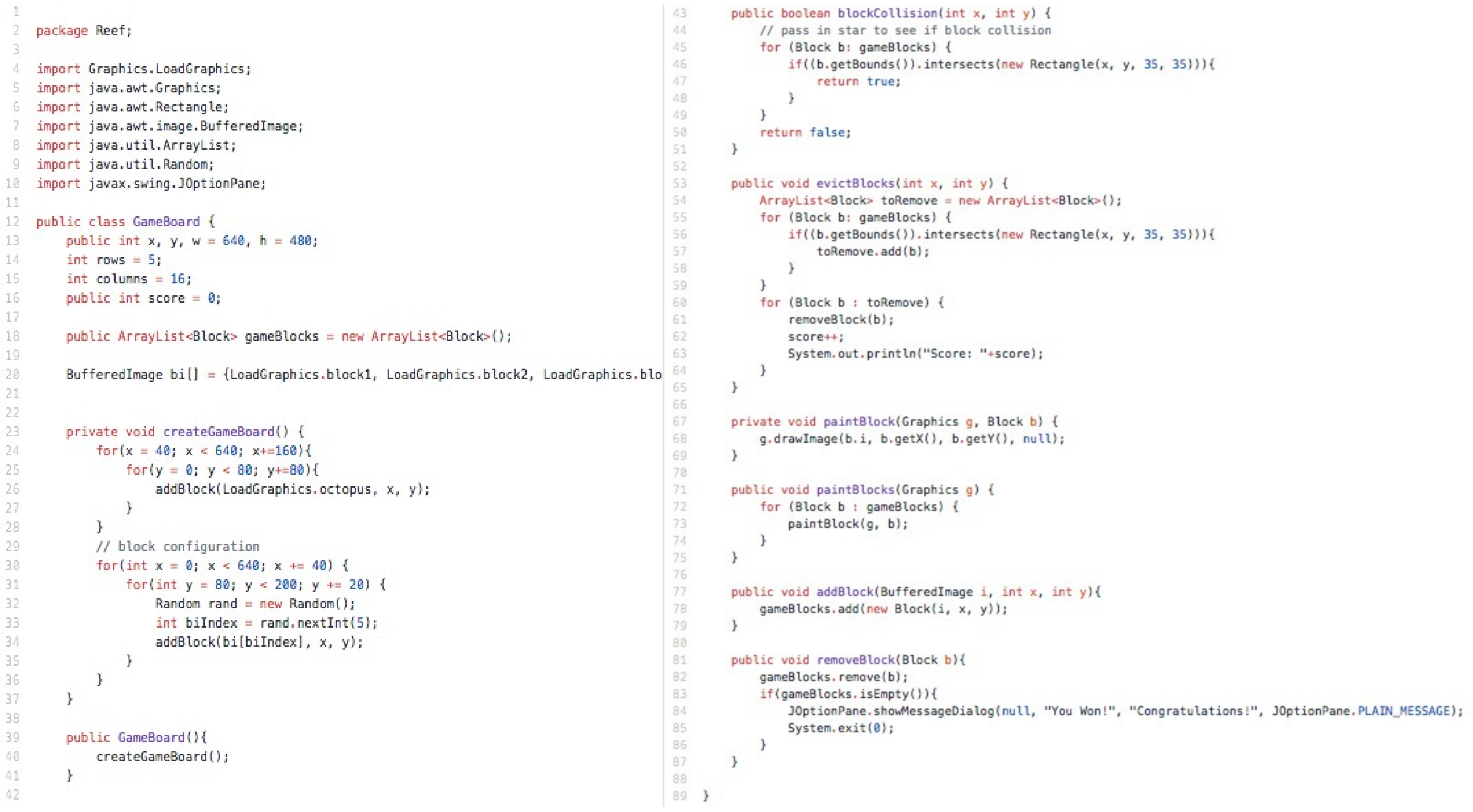
```java

package Reef;
/*source code from Plane Game*/

import Graphics.LoadGraphics;
import java.awt.Graphics;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JPanel;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.Rectangle;
import java.awt.event.ActionListener;


public class KeyControlP1 extends JPanel implements KeyListener, ActionListener{
    int w, h, x = 440, y = 450;
    public boolean running = false;
    public int paddleX = 440;
    public int starX = 100;
    public int starY = 400;
    public int starXDirection = -2;
    public int starYDirection = -2;

    public KeyControlP1() {
        addKeyListener(this);
        setFocusable(true);
        setFocusTraversalKeysEnabled(false);
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        updateStar();
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(LoadGraphics.shell, paddleX, y, null);
        g2.drawImage(LoadGraphics.starfish, starX, starY, null);
    }

    public void left(){
        running = true;
        paddleX -= 20;
    }

    public void right(){
        running = true;
        paddleX += 20;
    }

    @Override
    public void keyPressed(KeyEvent e) {
        //System.out.println("Key Pressed");
        int code = e.getKeyCode();
        if(code == KeyEvent.VK_LEFT){
             left();
             if(paddleX <= 8)
               paddleX = 8;

        }
        if(code == KeyEvent.VK_RIGHT){
            right();
            if(paddleX >= 560)
                paddleX = 560;

        }
        if(code == KeyEvent.VK_ESCAPE){
            System.exit(0);
        }
    }

    public void updateBlockCollision() {
        starYDirection = -starYDirection;
    }

    private void updateStar() {
        if(running){
            if(new Rectangle(starX, starY, 35, 35).intersects(new Rectangle(paddleX, y, 80, 30))){
                starYDirection = -starYDirection;
            }
            starY += starYDirection;
            starX += starXDirection;
            if(starY < 0)
                starYDirection = -starYDirection;
            if(starY > 480)
                starYDirection = -starYDirection;
            if(starX < 0)
                starXDirection = -starXDirection;
            if(starX > 605)
                starXDirection = -starXDirection;
        }
    }

    public boolean starOutOfScope() {
        if(starY > 480) {
            return true;
        }
        return false;
    }

    @Override
    public void actionPerformed(ActionEvent e){

    }

    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyReleased(KeyEvent e) {

    }
}
```

- **Game Class:**

  The game class is the main class in both games. It calls start(), stop(), init(), update(), render(), run(), initSound(), etc. Within the main function, a new game is created and started and a welcome message is printed using a JOptionPane. The Super Rainbow Reef's game class also calls updateStarBlockCollision(). In the Tank Game, the game class calls checkCollision() which handles collisions between tanks, walls, and bullets. It also increments the players' scores and establishes which blocks and bullets need to be removed from the game board as a result of collisions. JLabels were used to print the scores.

- **GameBoard Class:**

  The game board class creates an array list for bricks and randomly sets the color and where to put them on the screen. It also checks collisions between objects on the game board. It has the functions that add, remove, and paint the game objects. The only difference between the two games is that in the Tank Game they are called walls, while in the Super Rainbow Reef Game they are called blocks(which include octopus images at the top of the game board.)



```

package Reef;

import Graphics.LoadGraphics;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.JOptionPane;

public class GameBoard {
    public int x, y, w = 640, h = 480;
    int rows = 5;
    int columns = 16;
    public int score = 0;

    public ArrayList<Block> gameBlocks = new ArrayList<Block>();

    BufferedImage bi[] = {LoadGraphics.block1, LoadGraphics.block2, LoadGraphics.blo


    private void createGameBoard() {
        for(x = 40; x < 640; x+=160){
            for(y = 0; y < 80; y+=80){
                addBlock(LoadGraphics.octopus, x, y);
            }
        }
        // block configuration
        for(int x = 0; x < 640; x += 40) {
            for(int y = 80; y < 200; y += 20) {
                Random rand = new Random();
                int biIndex = rand.nextInt(5);
                addBlock(bi[biIndex], x, y);
            }
        }
    }

    public GameBoard(){
        createGameBoard();
    }

    public boolean blockCollision(int x, int y) {
        // pass in star to see if block collision
        for (Block b: gameBlocks) {
            if((b.getBounds()).intersects(new Rectangle(x, y, 35, 35))){
                return true;
            }
        }
        return false;
    }

    public void evictBlocks(int x, int y) {
        ArrayList<Block> toRemove = new ArrayList<Block>();
        for (Block b: gameBlocks) {
            if((b.getBounds()).intersects(new Rectangle(x, y, 35, 35))){
                toRemove.add(b);
            }
        }
        for (Block b : toRemove) {
            removeBlock(b);
            score++;
            System.out.println("Score: "+score);
        }
    }

    private void paintBlock(Graphics g, Block b) {
        g.drawImage(b.i, b.getX(), b.getY(), null);
    }

    public void paintBlocks(Graphics g) {
        for (Block b : gameBlocks) {
            paintBlock(g, b);
        }
    }

    public void addBlock(BufferedImage i, int x, int y){
        gameBlocks.add(new Block(i, x, y));
    }

    public void removeBlock(Block b){
        gameBlocks.remove(b);
        if(gameBlocks.isEmpty()){
            JOptionPane.showMessageDialog(null, "You Won!", "Congratulations!", JOptionPane.PLAIN_MESSAGE);
            System.exit(0);
        }
    }

}
```

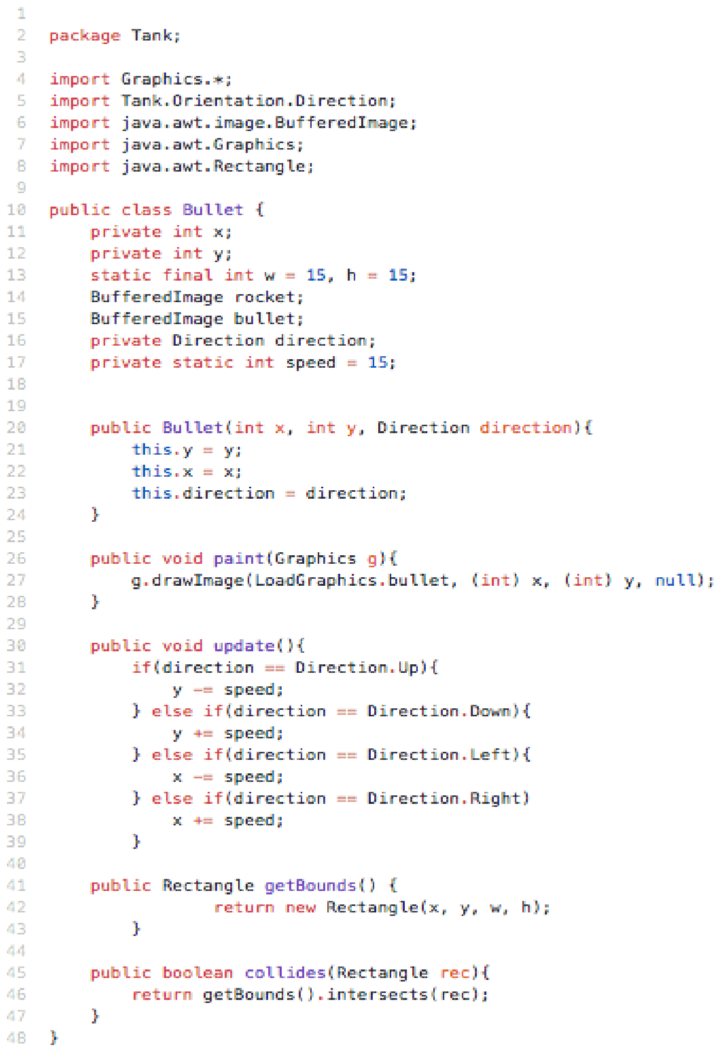
<u>Detailed Class Explanation – Tank Game Classes:</u>

- **<u>KeyControlP2 Class:</u>**

  Since the Tank Game class is for two players, there is a second key control class to handle another array list for the second tank's bullets and the functions that handle their creation, collision, movement, and removal form the game board. See KeyControlP1 for further information.

- **<u>Bullet Class:</u>**

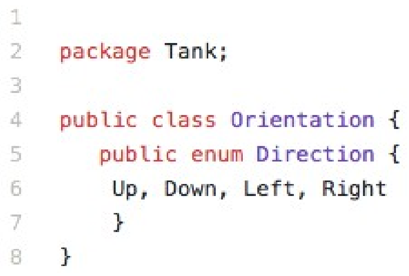
  The bullet class has paint(), update(), getBounds(), and collides() functions to handle the individual bullets and their location and orientation.



```java

package Tank;

import Graphics.*;
import Tank.Orientation.Direction;
import java.awt.image.BufferedImage;
import java.awt.Graphics;
import java.awt.Rectangle;

public class Bullet {
    private int x;
    private int y;
    static final int w = 15, h = 15;
    BufferedImage rocket;
    BufferedImage bullet;
    private Direction direction;
    private static int speed = 15;


    public Bullet(int x, int y, Direction direction){
        this.y = y;
        this.x = x;
        this.direction = direction;
    }

    public void paint(Graphics g){
        g.drawImage(LoadGraphics.bullet, (int) x, (int) y, null);
    }

    public void update(){
        if(direction == Direction.Up){
            y -= speed;
        } else if(direction == Direction.Down){
            y += speed;
        } else if(direction == Direction.Left){
            x -= speed;
        } else if(direction == Direction.Right)
            x += speed;
        }

    public Rectangle getBounds() {
                return new Rectangle(x, y, w, h);
        }

    public boolean collides(Rectangle rec){
        return getBounds().intersects(rec);
    }
}
```

- **<u>Orientation Class:</u>**

  The orientation class establishes possible directions of the game objects.

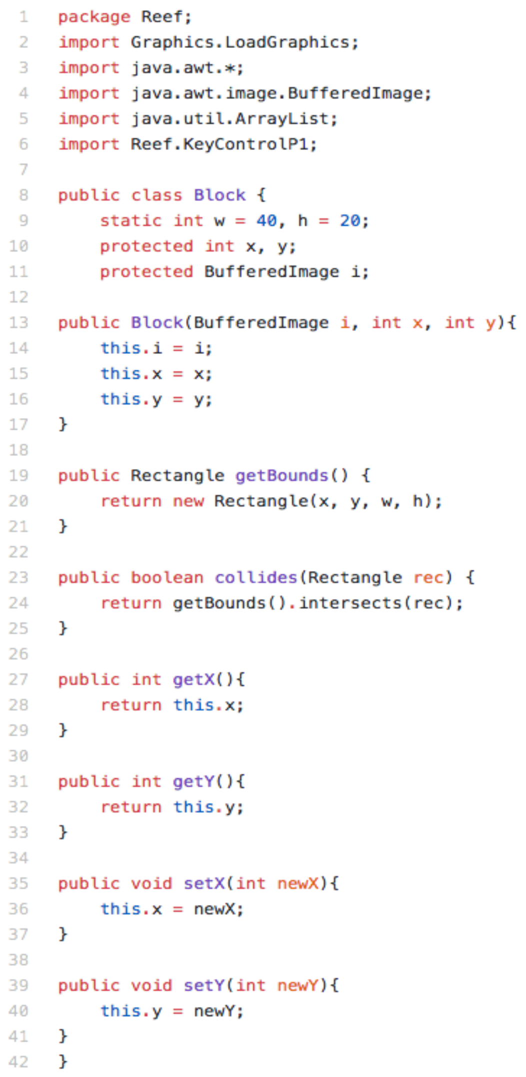


```java

package Tank;

public class Orientation {
   public enum Direction {
    Up, Down, Left, Right
    }
}
```

<u>Detailed Class Explanation – SRR Game Classes:</u>

- **<u>Block Class:</u>**

  The block class handles the game objects by checking the bounds and collisions.



```java
package Reef;
import Graphics.LoadGraphics;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import Reef.KeyControlP1;

public class Block {
    static int w = 40, h = 20;
    protected int x, y;
    protected BufferedImage i;

public Block(BufferedImage i, int x, int y){
    this.i = i;
    this.x = x;
    this.y = y;
}

public Rectangle getBounds() {
    return new Rectangle(x, y, w, h);
}

public boolean collides(Rectangle rec) {
    return getBounds().intersects(rec);
}

public int getX(){
    return this.x;
}

public int getY(){
    return this.y;
}

public void setX(int newX){
    this.x = newX;
}

public void setY(int newY){
    this.y = newY;
}
}
```

<u>Self-Reflection on Software Design, Development and Reusability:</u>

In retrospect, I would've liked to have a more structured format for my classes. Reusability did not become a pressing concern until I started my second game. Since many of my classes weren't general enough, I had to reformat a lot of my code for the Super Rainbow Reef Game. I had difficulties with the Tank Game e.g. collision detection and movement. The movement is not as smooth as I would like. The tanks can only move in four possible directions and the keyListener only allows one tank to move at a time. I was unable to rotate the bullets based on the angle they are shot from. In regards to collision detection, the edge detection is slightly off and causes walls to be removed early and bullets sometimes graze the edge of a tank without colliding and drawing an explosion. Similarly, in the Rainbow Reef Game sometimes the star will skip across the surface of the paddle if the paddle is still in motion when they collide. Unfortunately, the Super Rainbow Reef scoring system counts two bricks that are hit at the same time as one point, however, sequential bricks are counted correctly. I would've also liked to make the scoreboards for both games look cleaner and have high scores.

<u>Conclusion:</u>

Overall, this was my first experience building java games and I learned so much throughout this summer course. I was able to produce two functioning games with basic features. While they are not as polished as I would like, they both carry out simple gaming functions correctly. There are a few bugs listed in the self-reflection section, but other game functionalities have been tested and work correctly. For my next long-term programming project, I will have a more general set of packages and classes in order to maximize reusability and minimize time spent rewriting code.