T.C.
MALTEPE UNIVERSITY
FACULTY OF ENGINEERING AND NATURAL SCIENCES
SOFTWARE ENGINEERING DEPARTMENT

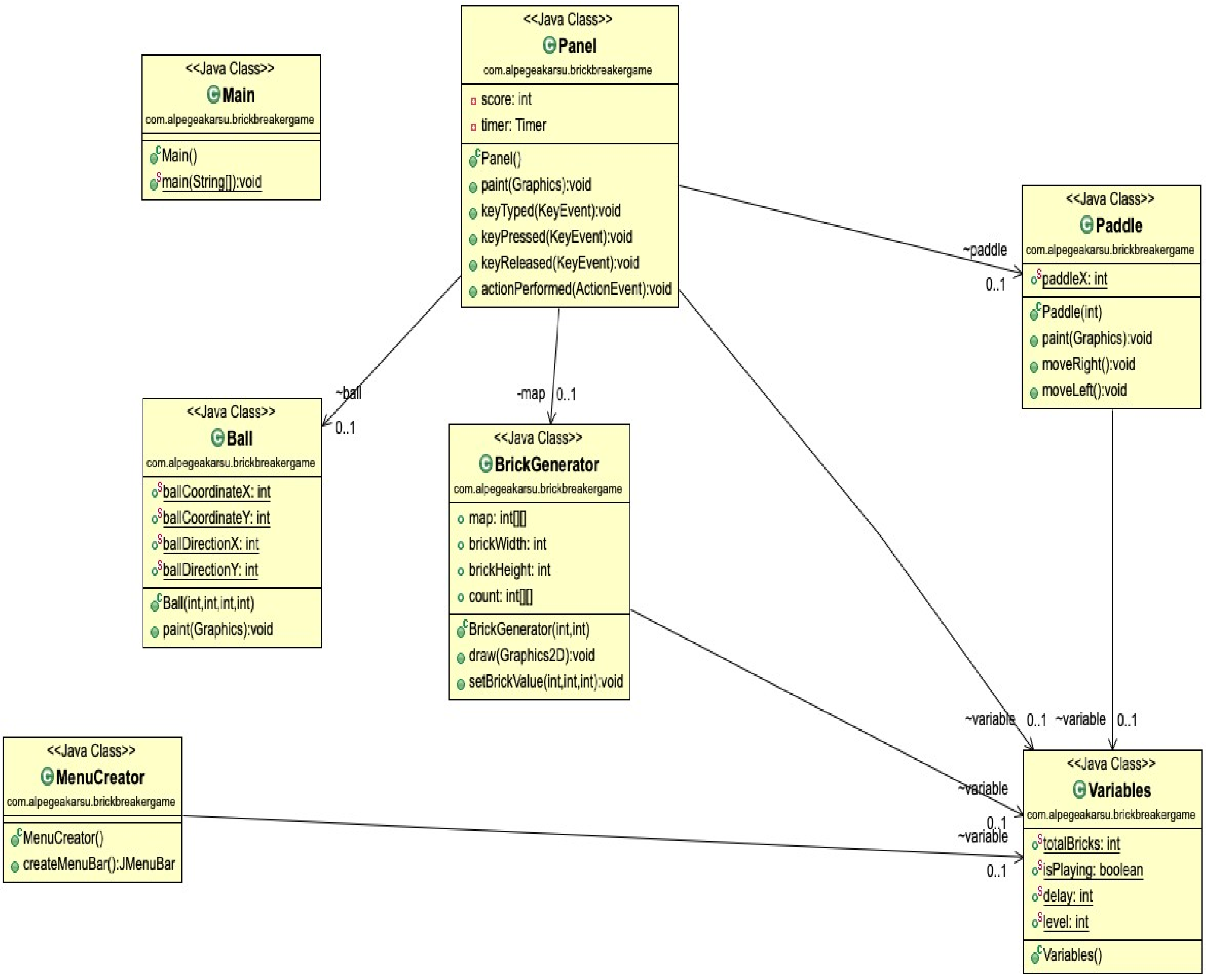
SOFTWARE CONSTRUCTION

ALP EGE AKARSU
160706029

MIDTERM PROJECT

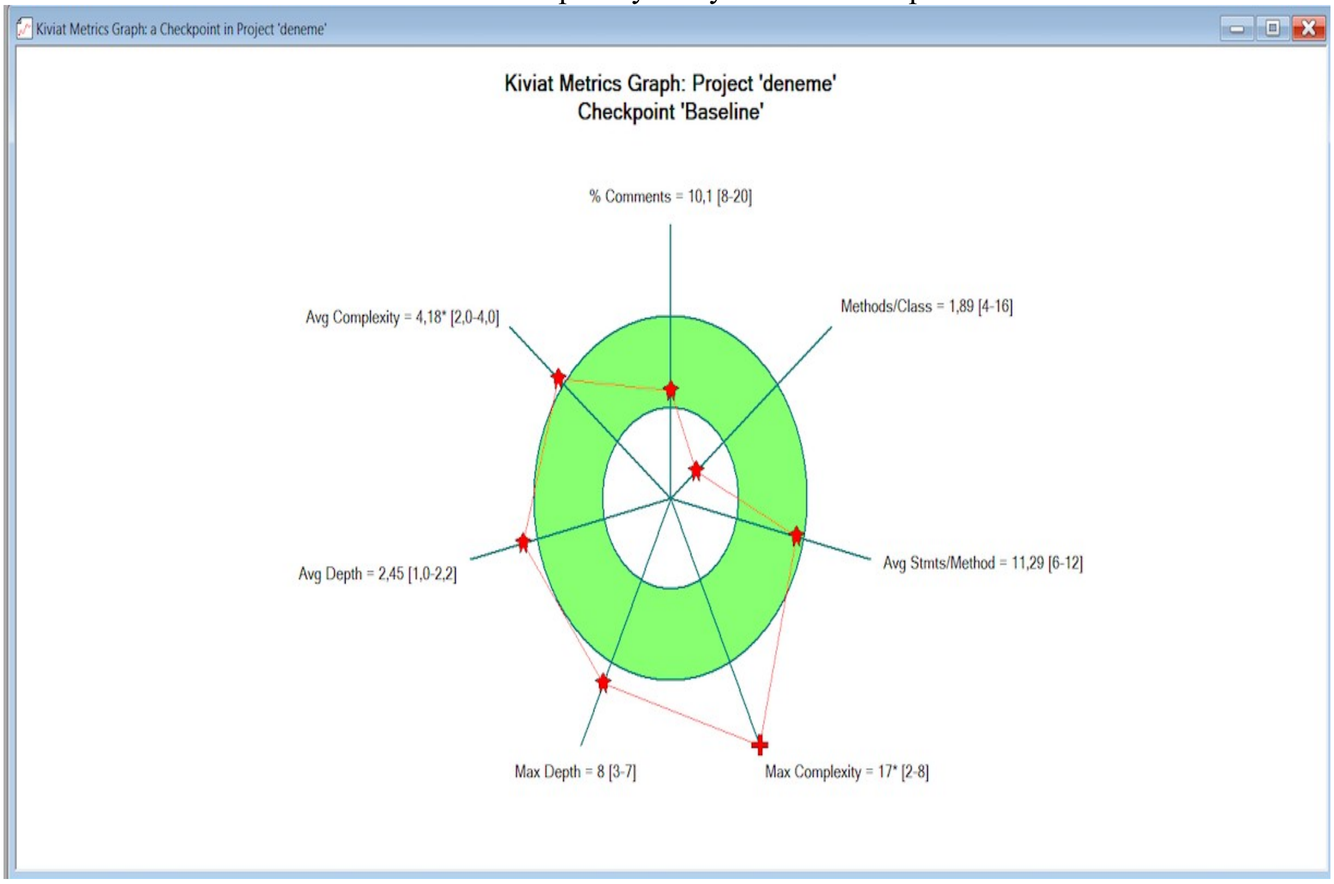
DESIGN

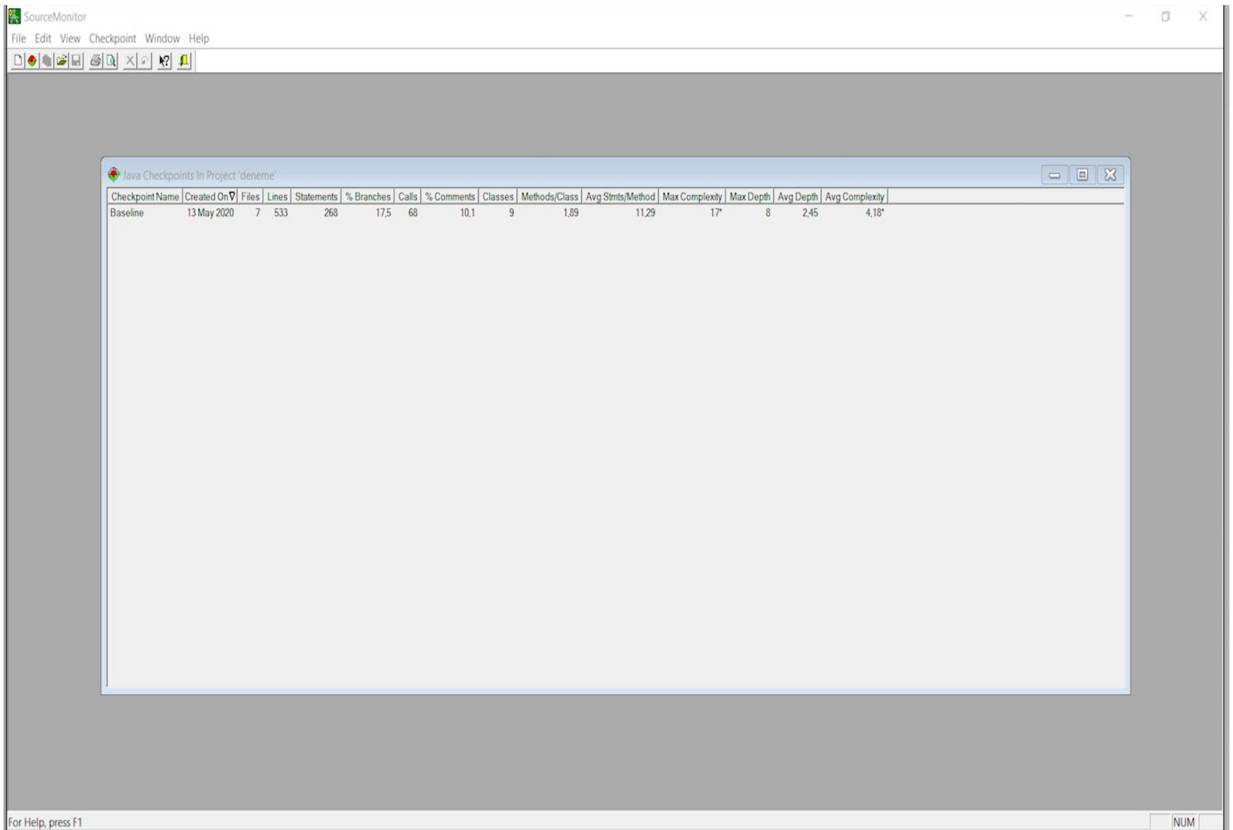
I first drew a UML diagram for my design. Then I started to write the codes by writing the classes I specified in the UML diagram. In addition to the UML diagram during the design phase, I had to design the game's graphical interface as I developed the game. For this, I tried on Java Swing library. The drawn UML diagram appears below. I explained every stage of this project in my project report.



SOURCE CODE COMPLEXITY ANALYSIS

I used Source Monitor to calculate complexity analysis and the outputs like as below.





SourceMonitor
File Edit View Checkpoint Window Help
Java Checkpoints In Project 'deneme'
For Help, press F1
NUM

Checkpoint Name	Created On	Files	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
Baseline	13 May 2020	7	533	268	17,5	68	10,1	9	1,89	11,29	17*	8	2,45	4,18*

```
package com.alpegeakarsu.brickbreakergame;

import java.awt.Color;
import java.awt.Graphics;

public class Ball {
    public static int ballCoordinateX;
    public static int ballCoordinateY;
    public static int ballDirectionX;
    public static int ballDirectionY;

    public Ball(int ballCoordinateX,int
ballCoordinateY,int ballDirectionX,int
ballDirectionY) {
        this.ballCoordinateX = ballCoordinateX;
        this.ballCoordinateY = ballCoordinateY;
        this.ballDirectionX = ballDirectionX;
        this.ballDirectionY = ballDirectionY;


    }

    public void paint(Graphics g){
    g.setColor(Color.yellow);
    g.fillOval(ballCoordinateX, ballCoordinateY, 20,
20);

    }
}



package com.alpegeakarsu.brickbreakergame;

/* IMPORTS */
import java.awt.Color;
import java.awt.Graphics;
/* IMPORTS */

public class Paddle {
   /* ATTRIBUTES */
    public static int paddleX;

    /* ATTRIBUTES */

    /* CONSTRUCTOR */
    public Paddle(int paddleX){
        Paddle.paddleX = paddleX;
    }
    /* CONSTRUCTOR */

    public void paint(Graphics g){
     g.setColor(Color.green);
    g.fillRect(paddleX, 550, 100, 8);


    }
    public void moveRight(){
       if(Variables.level == 0){
         Variables.isPlaying = true;
           paddleX += 20;
       }
       else if(Variables.level == 1){
         Variables.isPlaying = true;
           paddleX += 30;
       }
       else if(Variables.level == 2){
         Variables.isPlaying = true;
           paddleX += 40;
       }
       else if(Variables.level == 3){
         Variables.isPlaying = true;
           paddleX += 60;
       }
       else {}
    }
    public void moveLeft(){
       if(Variables.level == 0){
         Variables.isPlaying = true;
           paddleX -= 20;
       }
       if(Variables.level == 1){
         Variables.isPlaying = true;
           paddleX -= 30;
       }
       if(Variables.level == 2){
         Variables.isPlaying = true;
           paddleX -= 40;
       }
        if(Variables.level == 3){
            Variables.isPlaying = true;
           paddleX -= 60;
       }

    }
}

public class Variables {
    public static int totalBricks = 21;
    public static boolean isPlaying = false;
    public static int delay = 8;
    public static int level = 0;

}

public class Main{
    public static void main(String[] args){
    JFrame frame = new JFrame();
    Panel panel = new Panel();
    frame.setBounds(10,10,700,605);
    frame.setTitle("SimpleTry");
    frame.setResizable(false);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
;
    MenuCreator menubar = new MenuCreator();
    frame.setJMenuBar(menubar.createMenuBar());
    frame.add(panel);
     frame.setVisible(true);
    }

}
```

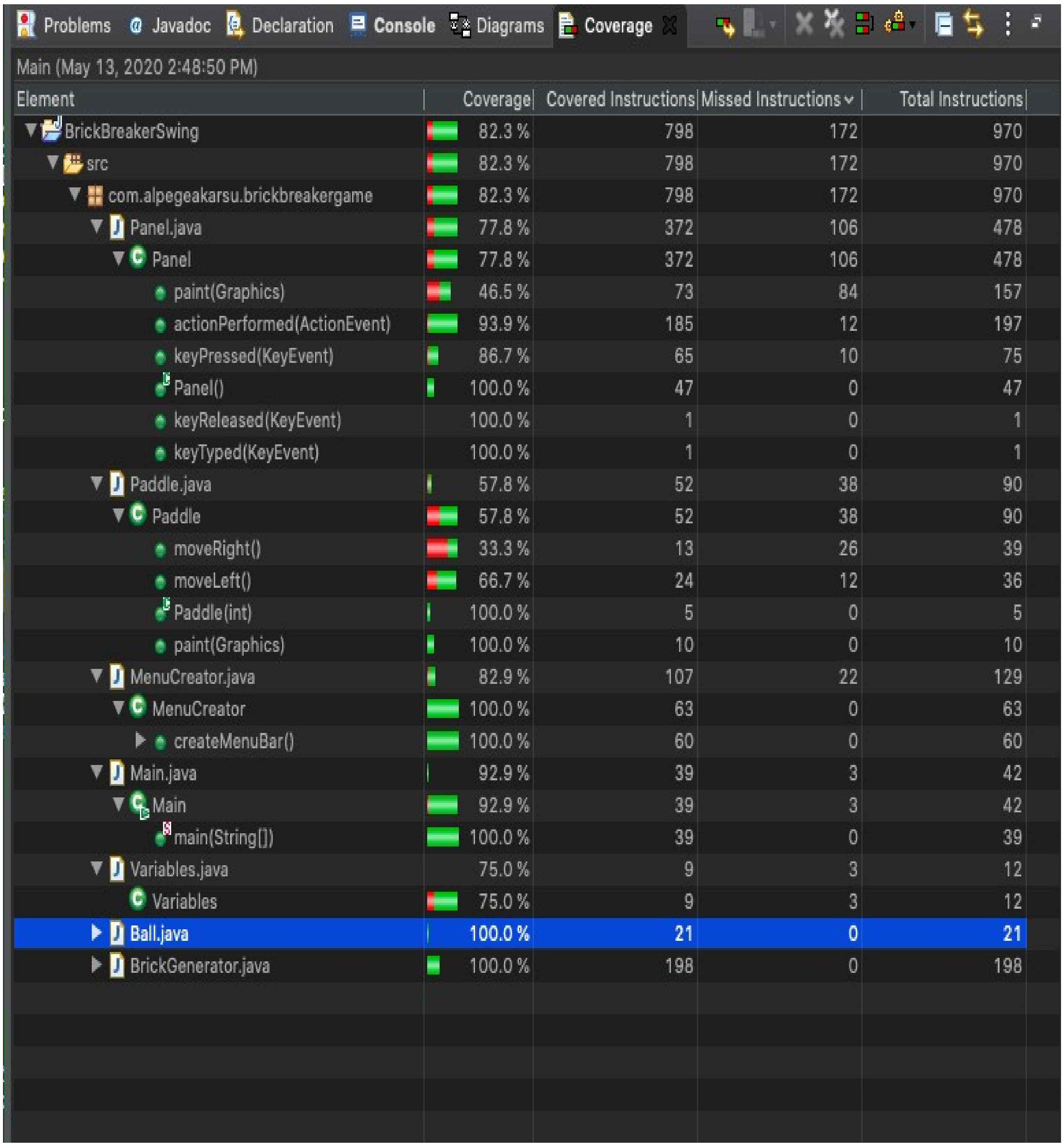
```
public class BrickGenerator {
   /* ATTRIBUTES */
    public int map[][];
    public int brickWidth;
    public int brickHeight;
    public int count[][];
    Variables variable;
    /* ATTRIBUTES */

public BrickGenerator(int row,int column){
        map = new int[row][column];
        count = new int[row][column];
        for(int i=0; i< map.length; i++){

            for(int j=0; j< map[0].length; j++){
                map[i][j] = 1;
                count[i][j] = 1;
            }

        }
        brickWidth = 540/column; // 540/7 = 77
brickWidth = 77
        brickHeight = 150/row;  // 150/3 = 50
brickHeight = 50

    }

    public void draw(Graphics2D g){
      for(int i=0; i< map.length; i++){

            for(int j=0; j< map[0].length; j++){
                if(map[i][j]>0){
               if(j % 2 == 0) {
                       g.setColor(Color.blue);
                    }else g.setColor(Color.white);

g.fillRect(j*brickWidth+80,i*brickHeight + 50 ,
brickWidth, brickHeight);
                 g.setStroke(new BasicStroke(3));
            g.setColor(Color.black);
            g.drawRect(j*brickWidth+80,i*brickHeight
+ 50 , brickWidth, brickHeight);

                }
            }

        }
    }
    @SuppressWarnings("static-access")
    public void setBrickValue(int value,int row,int
column){

         if(Variables.level == 0 || Variables.level
== 3){
         map[row][column] = value;
         Variables.totalBricks--;
         }
         if(Variables.level == 1 || variable.level ==
2){

                if(count[row][column] == 1){
                  map[row][column] = 1;
                  count[row][column] = 0;
                }
                 else{
                map[row][column] = value;
                Variables.totalBricks--;
                }
        }

    }


  public class MenuCreator{
         Variables variable;

    /* Returning a Menubar To Frame */
     public JMenuBar createMenuBar() {
         /* ATTRIBUTES */
        JMenuBar menuBar;
        JMenu menu,menu2;
        JMenuItem menuItem,menuItem2;
        /* ATTRIBUTES */
        menuBar = new JMenuBar(); //
        menu = new JMenu("Level");
        menu2 = new JMenu("Help");
       menuItem = new JMenuItem("Increase Level");
        menuItem2 = new JMenuItem("Start Page");
menuItem.setAccelerator(KeyStroke.getKeyStroke('L',
Toolkit.getDefaultToolkit ().getMenuShortcutKeyMask()));
        menuItem.addActionListener(new
java.awt.event.ActionListener() {
            @Override
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                menuItemActionPerformed(evt);
            }
            public void
menuItemActionPerformed(ActionEvent evt) {
                if(Variables.level <= 2){
                     Variables.level++;
                     Variables.delay = Variables.delay
-2;
                     Variables.isPlaying = false;
                  if(Variables.level == 1){
                   String message = "***LEVEL 2*** \
n"
                           + "In This Stage Ball Must
Hit 2 Times To Break Bricks..\n"
                           +"PRESS RIGHT-LEFT BUTTONS
TO CONTINUE ";

                   JOptionPane.showMessageDialog(null,
message);
                }

                   if(Variables.level == 2){
                   String message = "***LEVEL 3*** \
n"
                           + "In This Stage Ball Must
Hit 2 Times To Break Bricks..\n"
                           +"PRESS RIGHT-LEFT BUTTONS
TO CONTINUE ";
                   JOptionPane.showMessageDialog(null,
message);
                }
                     if(Variables.level == 3){
                   String message = "***LEVEL 4*** \
n"
                           + "This stage Similar To
Level 1 Because Ball Is Too Fast Good Luck..\n"
                           +"PRESS RIGHT-LEFT BUTTONS
TO CONTINUE ";
                   JOptionPane.showMessageDialog(null,
message);

                }

                }


            }

        });

         menuItem2.addActionListener(new
java.awt.event.ActionListener() {

            @Override

            public void
actionPerformed(java.awt.event.ActionEvent evt) {

                menuItem2ActionPerformed(evt);
            }

            public void
menuItem2ActionPerformed(ActionEvent evt) {
                String message = "To Pause : SPACE \
n"
                        + "To Continue : RIGHT-LEFT
BUTTONS\n"
                        + "To Restart : ENTER\n"
                        +"To Increase Level : CTRL or
COMMAND + L"   ;
                Variables.isPlaying = false;
                JOptionPane.showMessageDialog(null,
message);
           }
        });

        menu.add(menuItem);
       menu2.add(menuItem2);
        menuBar.add(menu);
        menuBar.add(menu2);
        return menuBar;

    }

}

}

// *****PANEL ****
public class Panel extends JPanel  implements
KeyListener,ActionListener {
    /*    ATTRIBUTES      */
    private int score = 0;
 private Timer timer;
    private BrickGenerator map;
    Variables variable;
    Paddle paddle;
    Ball ball;
    /*    ATTRIBUTES      */

   /* DEFAULT CONSTRUCTOR */
    public Panel(){
        map = new BrickGenerator(3,7);
        paddle = new Paddle(310);
        ball = new Ball(120,350,-1,-2);
        addKeyListener(this);
        setFocusable(true);
        setFocusTraversalKeysEnabled(false);
        timer = new Timer(Variables.delay,this);
        timer.start();
    }

    /* DEFAULT CONSTRUCTOR */


    /*    PAINT METHOD TO PREPARE PANEL      */
    @Override
    public void paint(Graphics g){
    // background
    g.setColor(Color.black);
    g.fillRect(1,1, 692, 592);
    // drawing map
    map.draw((Graphics2D)g);
    // borders
    g.setColor(Color.black);
    g.fillRect(0,0,3,592);
    g.fillRect(0,0,692,3);
    g.fillRect(691,0,3,592);
    // scores
    g.setColor(Color.white);
    g.setFont(new Font("serif",Font.BOLD,25));
    g.drawString(""+score,590,30);
    // the paddle
    paddle.paint(g);
    // the ball
    ball.paint(g);


    // If all the bricks are breaked
    if(Variables.totalBricks <= 0){

        Ball.ballDirectionX = 0;
        Ball.ballDirectionY = 0;
        g.setColor(Color.RED);
        g.setFont(new Font("serif",Font.BOLD,30));
        g.drawString("You Won,Total score:" +
score,260,300);

        g.setFont(new Font("serif",Font.BOLD,20));
        g.drawString("Press enter the
Restart",230,350);

    }


    // If the paddle out of borders
    if(Ball.ballCoordinateY > 570){

        Ball.ballDirectionX = 0;
        Ball.ballDirectionY = 0;
        g.setColor(Color.RED);
        g.setFont(new Font("serif",Font.BOLD,30));
        g.drawString("Game over,Total score:"+
score,190,300);

        g.setFont(new Font("serif",Font.BOLD,20));
        g.drawString("Press enter the
Restart",230,350);
    }

    g.dispose();

    }

    @Override

    public void keyTyped(KeyEvent e) {



    }


    @Override

    public void keyPressed(KeyEvent e) {
        if(e.getKeyCode() == KeyEvent.VK_RIGHT){
            if(Paddle.paddleX >= 600){
                Paddle.paddleX = 600;
            }else paddle.moveRight();
        }
        if(e.getKeyCode() == KeyEvent.VK_LEFT){
             if(Paddle.paddleX < 10){
                Paddle.paddleX = 10;
            }else paddle.moveLeft();

        }
        if(e.getKeyCode() == KeyEvent.VK_ENTER){
            if(!Variables.isPlaying){

                ball = new Ball(120,350,-1,-2);
                Paddle.paddleX = 310;
                score = 0;
                Variables.totalBricks = 21;
                map = new BrickGenerator(3,7);
                repaint();

            }


        }

        if(e.getKeyCode() == KeyEvent.VK_SPACE)
        setVisible(false);



        if(e.getKeyCode()== KeyEvent.VK_Q){
            timer.stop();
            System.exit(0);
        }
    }



    @Override
    public void keyReleased(KeyEvent e) {


    }



    @Override

    public void actionPerformed(ActionEvent e) {
        timer.start();
        timer.setDelay(variable.delay);
        repaint(); // its call paint method again and
draw everything again. to use moveright and moveleft
functions. (to see on program actually)
        if(Variables.isPlaying){
           if(new
Rectangle(Ball.ballCoordinateX,Ball.ballCoordinateY,2
0,20).intersects(new
Rectangle(Paddle.paddleX,550,100,8))){
               Ball.ballDirectionY = -
Ball.ballDirectionY;



          }
           A: for(int i=0; i<map.map.length; i++){
               for(int j=0; j<map.map[0].length; j++)
{
                   if(map.map[i][j] > 0 ){
                       int brickX = j* map.brickWidth
+ 80;
                       int brickY = i*
map.brickHeight + 50;
                       int brickWidth =
map.brickWidth;
                       int brickHeight =
map.brickHeight;

                       Rectangle compRect = new
Rectangle(brickX,brickY,brickWidth,brickHeight);
                       Rectangle ballRect = new
Rectangle(Ball.ballCoordinateX,Ball.ballCoordinateY,2
0,20);
                       Rectangle brickRect =
compRect;


if(ballRect.intersects(brickRect)){
                           map.setBrickValue(0, i,
j);
                           if(Variables.level == 0)
score+= 5;

                           if(Variables.level == 1)
score+=10;
                           if(Variables.level == 2)
score+=15;

                           if(Variables.level == 3)
score+=20;


                           if(Ball.ballCoordinateX +
19 <= brickRect.x || Ball.ballCoordinateX + 1 >=
brickRect.x + brickRect.width){

                          Ball.ballDirectionX  = -
Ball.ballDirectionX;

                           }else{

                               Ball.ballDirectionY =
-Ball.ballDirectionY;

                           }

                           break A;
                       }


                   }



               }



           }

               Ball.ballCoordinateX +=
Ball.ballDirectionX;

               Ball.ballCoordinateY +=
Ball.ballDirectionY;

               if(Ball.ballCoordinateX < 0){

                   Ball.ballDirectionX = -
Ball.ballDirectionX;

               }

               if(Ball.ballCoordinateY < 0){

                   Ball.ballDirectionY = -
Ball.ballDirectionY;

               }

               if(Ball.ballCoordinateX > 670){

                   Ball.ballDirectionX = -
Ball.ballDirectionX;

               }
           }

           repaint();

       }
```

UNIT TESTING

```
public class MenuCreatorIT {
    static MenuCreator menu;
    static JMenuBar menubar;
     static JFrame frame;
    public MenuCreatorIT() {

    }


    @BeforeClass
    public static void setUpClass() {
         menu = new MenuCreator();
        frame = new JFrame();
         frame.setBounds(5, 5, 400, 700);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


         frame.setVisible(true);
    }

    @AfterClass
    public static void tearDownClass() {

        frame.setJMenuBar(menubar);
    }



    @Test

    public void testCreateMenuBar() throws Exception
{

       menubar = menu.createMenuBar();

    }

}

public class PaddleIT {

    static Paddle paddle;

    static int paddleTestX = 20;


    @BeforeClass

    public static void setUpClass() {

        Variables.level = 1;



    }

    @Before

    public void setUp() {

        paddle = new Paddle(paddleTestX);



    }



    @After

    public void tearDown() {
        System.out.println(paddle.paddleX);

    }



    @Test

    public void testMoveLeft() {
        if(Variables.level == 0) {
            paddle.moveLeft();
        assertEquals(0,paddle.paddleX);
        }
    }


    @Test
    public void testMoveRight() {
       if(Variables.level == 0) {
       paddle.moveRight();
       assertEquals(40,paddle.paddleX);
       }

    }

    @Test
    public void testMoveLeftLevelOne(){
          if(Variables.level == 1){
              paddle.moveLeft();
            assertEquals(-10,paddle.paddleX);
            }

        }

      @Test
    public void testMoveRightLevelOne(){
          if(Variables.level == 1){
              paddle.moveRight();
            assertEquals(50,paddle.paddleX);
            }
        }

      @Test
    public void testMoveLeftLevelTwo(){
          if(Variables.level == 2){
              paddle.moveLeft();
            assertEquals(-20,paddle.paddleX);
            }

        }

      @Test
    public void testMoveRightLevelTwo(){
          if(Variables.level == 2){
              paddle.moveRight();
            assertEquals(60,paddle.paddleX);
            }

        }


}

public class VariablesIT {
    Variables variables;
    static int testBrick;
    static int testDelay;
    static int testLevel;
    static boolean testIsPlaying;
    public VariablesIT() {
    }
    @BeforeClass
    public static void setUpClass() {
        testBrick = Variables.totalBricks;
        testDelay = Variables.delay;
        testLevel = Variables.level;
        testIsPlaying = Variables.isPlaying;
    }
    @Test
    public void testVariables() {
        assertEquals(21,testBrick);
        assertEquals(8,testDelay);
        assertEquals(0,testLevel);
        assertEquals(false,testIsPlaying);

    }
}


public class BrickGeneratorIT {
     BrickGenerator brick;
     int row = 3;
     int column = 7;
     Graphics2D g;
    private BufferedImage img;
    JFrame frame = new JFrame();
    JPanel panel = new Jpanel();

public BrickGeneratorIT() {
        img = new
BufferedImage(400,600,BufferedImage.TYPE_INT_RGB);
        g = (Graphics2D) img.getGraphics();
         frame.setBounds(10,10,700,605);
    frame.setTitle("SimpleTry");
    frame.setResizable(false);


frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    brick = new BrickGenerator(row,column);
    panel.setBackground(Color.red);
    frame.add(panel);
     frame.setVisible(true);
    }
@Test
    public void testDraw() {

        brick.draw(g);

    }


 @BeforeClass
    public static void setUpClass() {
         menu = new MenuCreator();
        frame = new JFrame();
         frame.setBounds(5, 5, 400, 700);


frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setVisible(true);
    }
@AfterClass

 public static void tearDownClass()
{     frame.setJMenuBar(menubar);

    }

 @Test

    public void testCreateMenuBar() throws Exception
{

       menubar = menu.createMenuBar();

    }

}
```

CODE COVERAGE

I used a Eclipse tool to calculate code coverage. The output like as below



Main (May 13, 2020 2:48:50 PM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
BrickBreakerSwing	82.3 %	798	172	970
src	82.3 %	798	172	970
com.alpegeakarsu.brickbreakergame	82.3 %	798	172	970
Panel.java	77.8 %	372	106	478
Panel	77.8 %	372	106	478
paint(Graphics)	46.5 %	73	84	157
actionPerformed(ActionEvent)	93.9 %	185	12	197
keyPressed(KeyEvent)	86.7 %	65	10	75
Panel()	100.0 %	47	0	47
keyReleased(KeyEvent)	100.0 %	1	0	1
keyTyped(KeyEvent)	100.0 %	1	0	1
Paddle.java	57.8 %	52	38	90
Paddle	57.8 %	52	38	90
moveRight()	33.3 %	13	26	39
moveLeft()	66.7 %	24	12	36
Paddle(int)	100.0 %	5	0	5
paint(Graphics)	100.0 %	10	0	10
MenuCreator.java	82.9 %	107	22	129
MenuCreator	100.0 %	63	0	63
createMenuBar()	100.0 %	60	0	60
Main.java	92.9 %	39	3	42
Main	92.9 %	39	3	42
main(String[])	100.0 %	39	0	39
Variables.java	75.0 %	9	3	12
Variables	75.0 %	9	3	12
Ball.java	100.0 %	21	0	21
BrickGenerator.java	100.0 %	198	0	198

PROJECT REPORT

Before I started my project, I determined the classes I would create. Of course, I created a preliminary design even though some of these classes appeared after I started writing code. I researched examples of the game I wanted to design on the internet. My weak point in this project was that I didn't know Java Swing at all. I had never developed a Swing project before. In my project development process, I spent a while just learning the Java Swing library and its logic.

After making a preliminary design on paper, I started writing the codes of my project.

First, I created only one frame and one panel. All codes were included in the panel and frame. As I write new codes, new ideas began to emerge. As the number of codes in the panel class started to increase, I realized that I had to divide the project into classes. Since I was just learning the swing library, I was learning new things in every code I wrote, but this increased the complexity of the code. I first created the BrickGenerator class during classification. In this class, there was the creation of bricks that the ball would hit.

Although we were familiar with the rules we saw in the lesson, it took me a long time to do it on a project. In some cases, it could take me a day to transfer a routine from the panel class to another class. For example, it took me 4 days to transfer only the routine called MenuCreator to another class. To be realistic, this period is not convincing at all. However, even the operations that can be done in a simple way took me a lot of time since I had not developed a project in full and had not practiced enough object oriented language. Perhaps 80% of my creation of the project was spent by researching on the Internet.

If I continue with the project phases, I first wrote almost all of the classes I created in the Panel class. After running in the panel class, I tried to divide the routines into classes as much as I could to reduce the complexity of the panel class. The first 15-20 days of the project passed with creating the Panel class and the Main class (research, learning of the Java Swing library, object oriented programming course review). In the remaining time, I spent dividing the Panel class into other classes.

Towards the end of the project, after the game was fully operational, I focused on adding additional features to the game. As I learned more and more features of the Java Swing library every day, I decided to add a Menu to the game. While thinking about what to add for this Menu, I came to think of adding the Level variable. Whenever the player increased each level, a message should be displayed on the screen. In addition, after writing the code for this first, the Bricks had to be broken in two times when the player increased the level. Although this seems very easy from the outside, it has required me to do a day's research on it. At the same time, when the player increased the level, the speed of the ball had to increase, and when the speed of the ball increased, Paddle had to move quickly accordingly. After putting the level logic in the game, I added a new Menu and there should be information to help the player.

After this stage, in addition to the Panel class, the MenuCreator and BrickGenerator classes were added. At this stage, the game was working effectively. But the Panel code was still complicated and somehow I had to fix it. I had to pass some variables (such as Ball, Paddle) in the Components inside the panel code to other classes. In this process, while transferring these variables to other classes, I encountered a lot of errors and created the class called Variables. The project was running successfully when I ran some variables that were changed in the panel class in the Variables class. After the Variables, Panel and Ball classes were added to the project, the complexity of the Panel class decreased significantly.

My game was completely finished and after this process, I worked on correcting the variable names, creating comment lines, making the method names more meaningful.

PART 2 OF THE PROJECT REPORT

In this part of the project report, I will explain what I learned about making software and make self-criticism. I think this section will be very useful for the evaluation of the project. In this section, I will examine these situations in items.

1- First, I will start with self-criticism

Since I have never developed a GUI project before starting the project, I can admit that my learning phase took a little longer. I can also say that the requirements analysis and design stages we have seen in the course are not quite good in creating a project. It can be said that I did not create a professional design even though I had drawn the design in mind on a paper before I started writing my codes. To be honest, I created UML graphics after I finished my project. Because when I started the project, I really had no idea what to do. Since I am in the learning phase, I could not create a complete design and requirement analysis before starting the project.

2- About Defining the variables globally

If I continue with self-criticism, the second problem in my project may be to define some variables globally. Although I have done a lot of research to solve this problem, I could not fix it because I could not reach a complete solution.

3- About Unit Testing

Unfortunately, I realized that it was my biggest mistake to start writing Unit Tests after I created my project when the project was finished. In addition to what we have learned in class, I have read quite a lot of articles about Unit Testing on the internet, but I cannot say that I have literally learned Unit Testing during this time. Because my project was to develop a game, I had to learn GUI Testing. Regardless of how much I research on the internet, I will continue to work on GUI Testing, unfortunately, since the project is nearing the deadline. I will also add the things I tried for Unit Testing in my project to my report, but I have to mention that these Unit Tests have nothing to do with Unit Test. I need to do additional research on Unit Tests of projects containing GUI. So honestly my Unit Tests are garbage. The reason I couldn't create Unit Tests beautifully might be because my codes are complicated. Finally, the unit testing part is missing in my project.

4- About the Graphical Interface

As I mentioned in other parts of my report, it took me a long time to learn the Java Swing library and its logic, since I was developing the GUI project for the first time. Therefore, I could not create the view that I designed in mind at the beginning of the project. For example, after hitting and breaking the ball Bricks, I also wanted to drop items from the Bricks, but I did not have time and enough knowledge to do this. Likewise, after putting Level logic in the game, after hitting the ball twice, the walls were falling and I wanted the wall to crack as soon as the ball hit the wall, but I did not succeed.

5- About What I Learned In This Project

First, starting to develop this project has shown me how difficult but enjoyable it is to develop games. I had to learn Java Swing to develop a graphical interface and I think I learned the basics of Java Swing. Although I cannot fully explain everything in the creation of the project in the report, I learned a lot from the time I started doing the project a month ago. I thought I knew object oriented programming because I passed the object oriented programming lesson. But I saw that this is not the case at all. Whenever I needed to add new features for the project, I noticed that there are rules or routines in the Java library that I need to learn. Although my project contains deficiencies, I think the difficulties I had while doing my project were very beneficial to me.

6- In Conclusion

I realize that I repeat it over and over again at the time of the report, but my project was very useful to me because I had no prior knowledge to create a GUI project. I learned a lot of new information, it was an introduction to game programming. I learned that the intricacies of object oriented programming can not only be learned by reading a book, but the developer can only learn these concepts as the project develops. I am aware of the missing parts of my project. Maybe if there was more time, I could add additional features to the project I developed.

REFERENCES AND TOOLS I USED

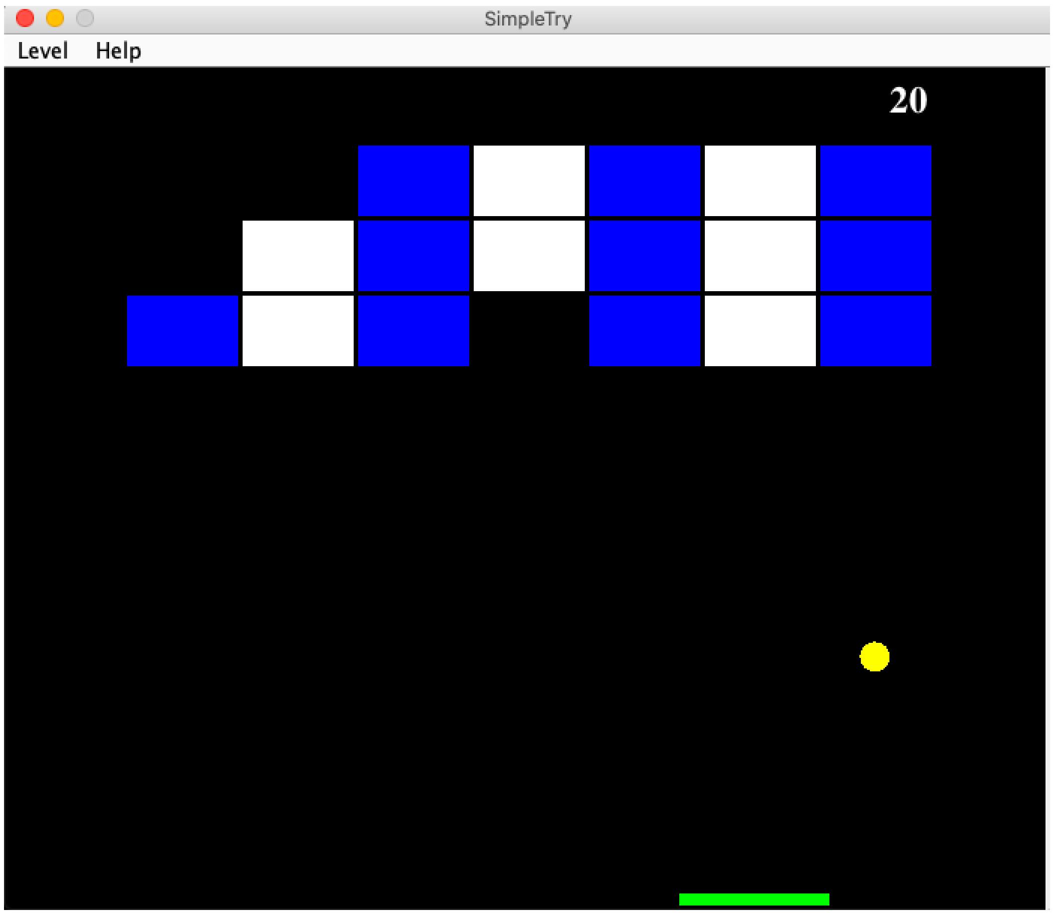
1- I used the Times New Roman font in size 12 in the writing section of my report.

2- I used the Courier New font in size 8 in the code section of my report.

3- I used NetBeans IDE while developing a project.

4- I do not think there is a part to be referenced in any part of my project. Because I only got ideas from the sites. And I researched a lot of websites such as StackOverFlow etc.

5- And the final image of this report will be game which is I created in this project.


SimpleTry
Level
Help
20