BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	Cleosa	Carmen	2502009601

Course Code : COMP6699001 **Course Name** : Object Oriented Programming

Class : L2AC **Name of Lecturer(s)** : Jude Joseph Lamug Martinez

Major : Computer Science

Title of Assignment : Brick Breaker Game

Type of Assignment : Final Project

Submission Pattern

Due Date : **Submission Date** :

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Carmen Cleosa

Table of Contents

“Brick Breaker Game”

Name : Carmen Cleosa

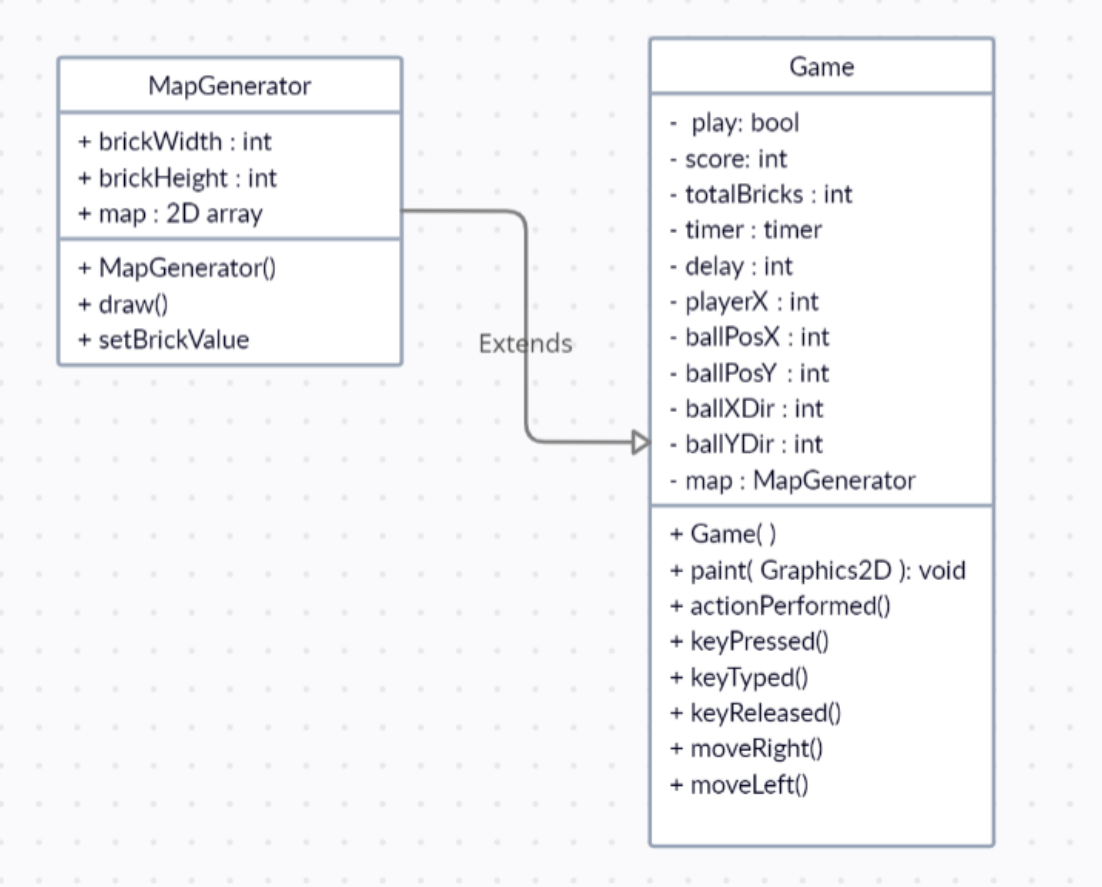
ID : 2502009601

I. Program Description

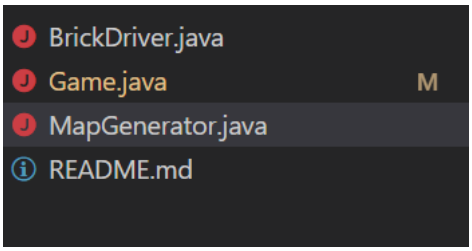
This is a simple Brick breaking game where the player must break/smash all the bricks on the screen before the win the game. Unlike traditional brick break games that allows players 3 lives, the players are only allowed 1 live as there is also only 1 level. This also gives a little more enjoyment because there are higher risks.

As someone who barely play games this game stick with you. This simple game was played by a lot of kids back then because it was simple enough that kids of any age could play and enjoy the game. And remembering that is what motivates me to try and attempt making this game.

II. Class Diagram



III. Project Technical Description



This Project contains 3 files:

- BrickDriver.java : This is the main driver file which will run the game program.
- MapGenerator.java: This file is to generate the map and creates the arrays for the bricks and sets values to them.
- Game.java: This file contains the “game” like how the scoring system would work, creating the ball and paddles and how it moves etc.

IV. Code Explanation

First, we will start with the MapGenerator file

MapGenerator.java > ...

```java
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.BasicStroke;
/**
 * MapGenerator
 */
public class MapGenerator{
    public int map [][];
    public int brickWidth;
    public int brickHeight;

    public MapGenerator(int row, int col) {
        map = new int[row][col];
        for (int i = 0; i < map.length; i++) {
            for (int j = 0; j < map[0].length; j++) {
                map[i][j] = 1;
            }
        }
        brickWidth = 540/col;
        brickHeight = 150/row;
    }
```

The MapGenerator() function creates a 2D array and would later split into the size determine by the number of column and row that is later inputted in the game.java file.

```java
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) {
                g.setColor(Color.white);
                g.fillRect(j*brickWidth+80, i*brickHeight+50, brickWidth, brickHeight);

                // this is to show separate brick
                g.setStroke(new BasicStroke(width: 3));
                g.setColor(Color.black);
                g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
            }
        }
    }
}
```

The draw() function draw the bricks that players would break sets the color and border to separate the bricks

```java
public void setBrickValue(int value,int row,int col)
{
    map[row][col] = value;

}
```

The setBricksValue() sets a value to each of the bricks that would be drawn. This would later help in making the brick disappear when it collides with the ball

Game.java > Game > actionPerformed(ActionEvent)

```java
import java.awt.event.*;

import javax.swing.*;

import java.awt.*;

import javax.swing.Timer;
/**
 * Game
 */
public class Game extends JPanel implements KeyListener, ActionListener{
    private boolean play = false;
    private int score = 0;

    private int totalbricks = 21;

    private Timer timer;
    private int delay = 8;

    private int playerX = 310;

    private int ballPosX = 120;
    private int ballPosY = 350;
    private int ballXDir = -1;
    private int ballYDir = -2;

    private MapGenerator map;
```

```java
public Game() {
    map = new MapGenerator(row: 3,col: 7);
    addKeyListener(this);
    setFocusable(focusable: true);
    setFocusTraversalKeysEnabled(focusTraversalKeysEnabled: false);
    timer = new Timer(delay, this);
    timer.start();
}
```

The game() function is what creates the game by starting the timer which would then start the game. It also sets the number of rows and columns of bricks that will be generated. Even if there is technically only 1 level the number of bricks is customizable should you wish to add difficulty.

```java
public void paint(Graphics g) {
    // background
    g.setColor(Color.black);
    g.fillRect(x: 1, y: 1, width: 692, height: 592);

    // drawing map
    map.draw((Graphics2D) g);

    // borders
    g.setColor(Color.yellow);
    g.fillRect(x: 0, y: 0, width: 3, height: 592);
    g.fillRect(x: 0, y: 0, width: 692, height: 3);
    g.fillRect(x: 691, y: 0, width: 3, height: 592);

    // the scores
    g.setColor(Color.white);
    g.setFont(new Font(name: "serif", Font.BOLD, size: 25));
    g.drawString("" + score, x: 590, y: 30);

    // the paddle
    g.setColor(Color.white);
    g.fillRect(playerX, y: 550, width: 100, height: 8);

    //ball
    g.setColor(Color.yellow);
    g.fillOval(ballPosX, ballPosY, width: 20, height: 20);
```

The paint() function paints the whole game on the window. Starting from the background, the map, the borders, ball, scores, and platform.

```java
public void actionPerformed(ActionEvent e) {
    timer.start();

    if (play) {
        if (new Rectangle(ballPosX, ballPosY, width: 20, height: 20).intersects(new Rectangle(playe
            ballYDir = -ballYDir;
            ballXDir = -2;
        }
        else if(new Rectangle(ballPosX, ballPosY, width: 20, height: 20).intersects(new Rectangle(p
        {
            ballYDir = -ballYDir;
            ballXDir = ballXDir + 1;
        }
        else if(new Rectangle(ballPosX, ballPosY, width: 20, height: 20).intersects(new Rectangle(p
        {
            ballYDir = -ballYDir;
        }
```

```java
// check map collision with the ball
App:
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 bricksWidth = map.brickWidth;
            int bricksHeight = map.brickHeight;

            Rectangle rect = new Rectangle(brickX, brickY, bricksWidth, bricksHeight);
            Rectangle ballRect = new Rectangle(ballPosX, ballPosY, width: 20, height: 20);
            Rectangle brickRect = rect;

            if (ballRect.intersects(brickRect)) {
                map.setBrickValue(value: 0, i, j);
                totalbricks--;
                score += 5;

                // when ball hit right or left of brick
                if (ballPosX + 19 <= brickRect.x || ballPosX + 1 >= brickRect.x + brickRect.wi
                    ballXDir = -ballXDir;
                }
                // when ball hits top or bottom of brick
                else {
                    ballYDir = -ballYDir;
                }
                break App;
            }
```

It checks the value of the bricks are 0 if so then it gets its perimeter and creates a rectangle based of the size of the perimeter. A rectangle is basically a hitbox and if the hitbox of the ball and the brick collide it would change the value of the brick and at your score by 5.

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

    if (e.getKeyCode() == KeyEvent.VK_ENTER) {
        if (!play) {
            ballPosX = 120;
            ballPosY = 350;
            ballXDir = -1;
            ballYDir = -2;
            score = 0;
            playerX = 310;
            totalbricks = 21;
            map = new MapGenerator(row: 3, col: 7);

            repaint();
```

The keyPressed() function lets the program knows what happens when you press certain keys. Ex: pressing Enter restarts the game or pressing the left arrow key lets you move the platform left.

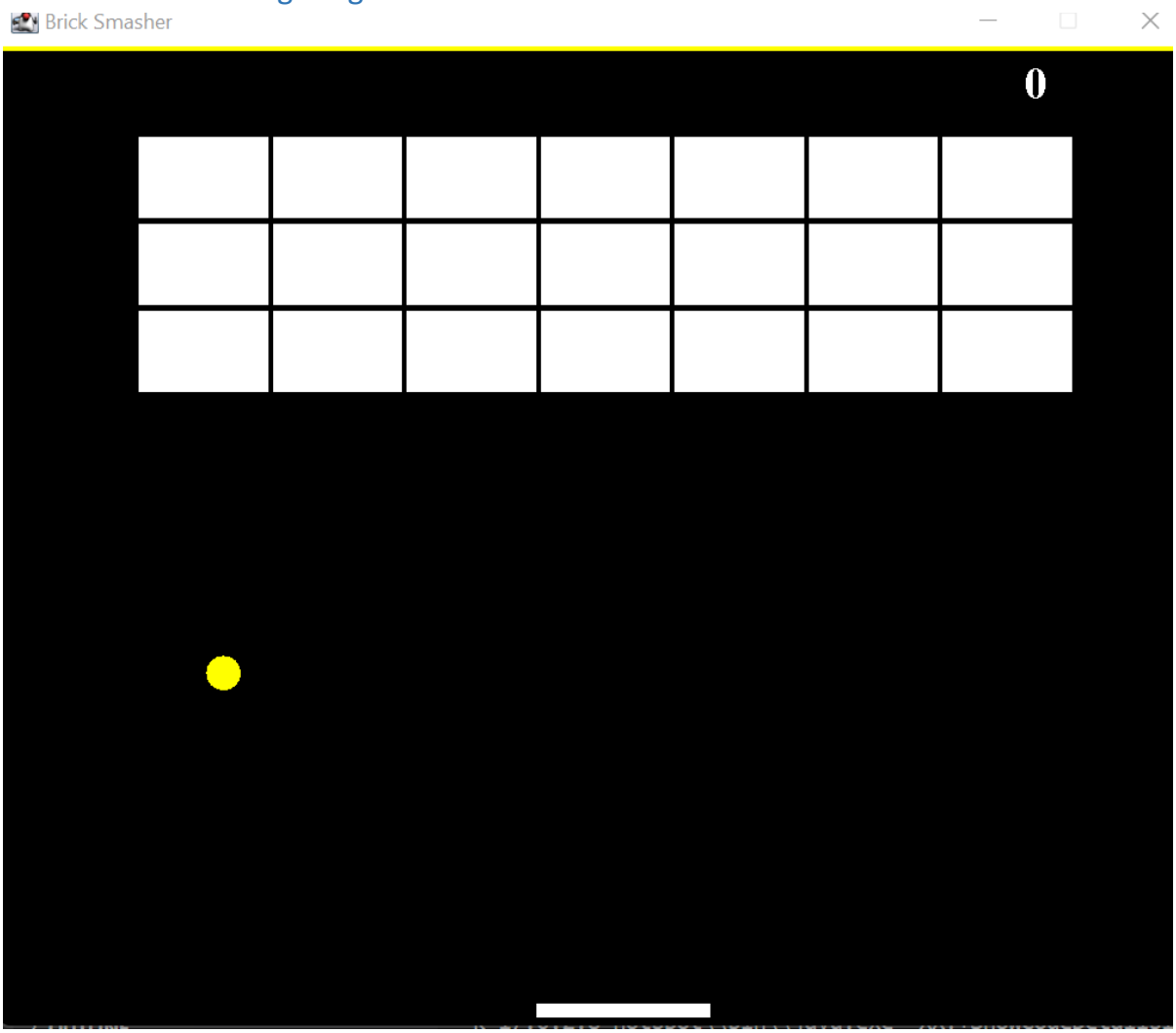
```
public void moveRight ()
{
    play = true;
    playerX += 20;
}
public void moveLeft ()
{
    play = true;
    playerX -= 20;
}
```

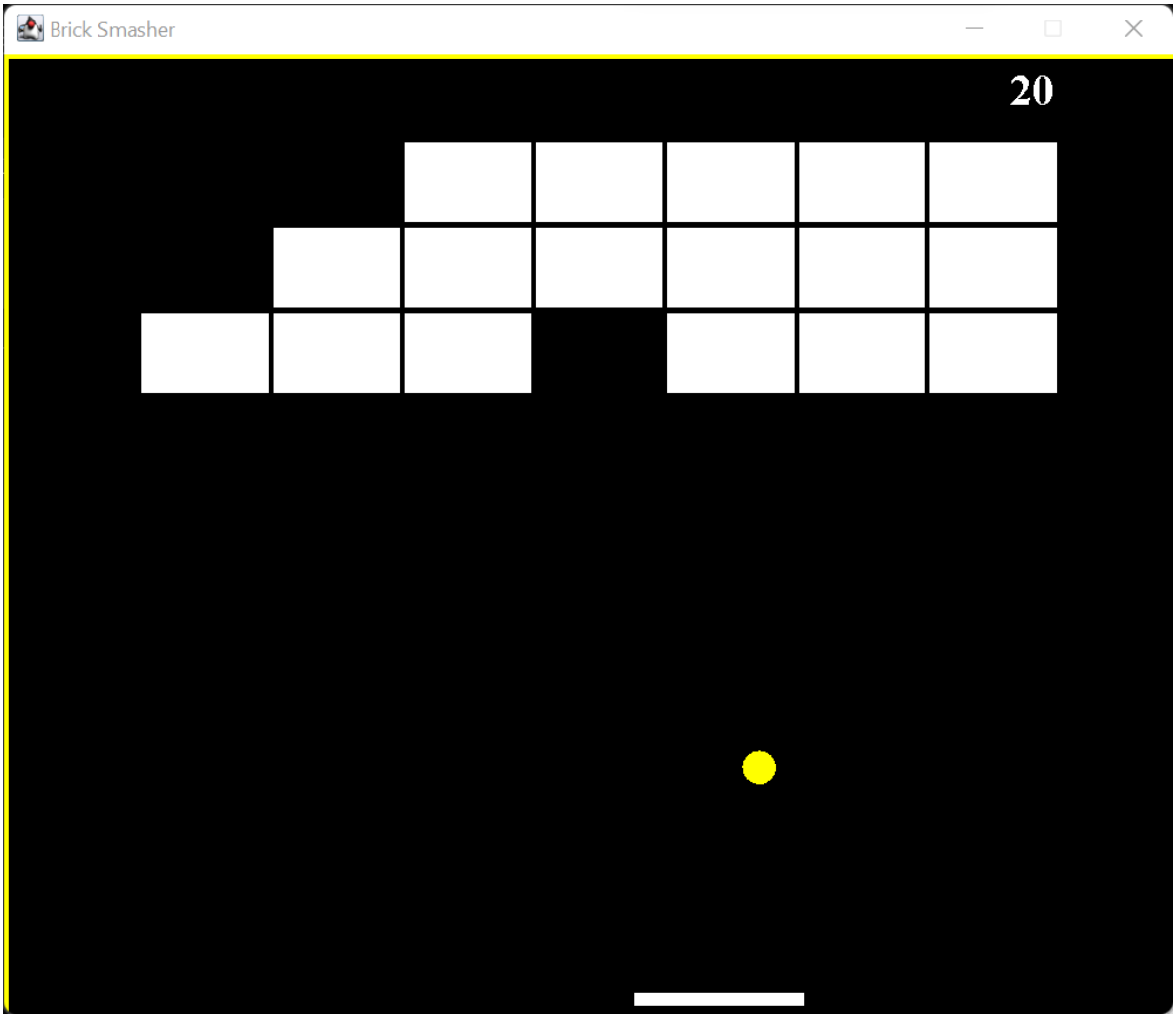
The moveLeft and moveRight function makes sure that the game is running or not, then it would move the player right or left when you press the arrow key.

```java
import javax.swing.JFrame;

/**
 * BrickDriver
 */
public class BrickDriver {
    Run | Debug
    public static void main(String[] args) {
        JFrame obj = new JFrame();
        Game gamePlay = new Game();
        obj.setBounds(x: 10,y: 10,width: 700,height: 600);
        obj.setTitle(title: "Brick Smasher");
        obj.setResizable(resizable: false);
        obj.setVisible(b: true);
        obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        obj.add(gamePlay);
    }

}
```

This is the main driver that would start the game by making the window where the game would start from. The window is set with its height and width

V. Evidence of Working Program




Brick Smasher
20

VI. Project Link

https://github.com/Carmen1104/OOP-FP

VII. References

https://stackoverflow.com/