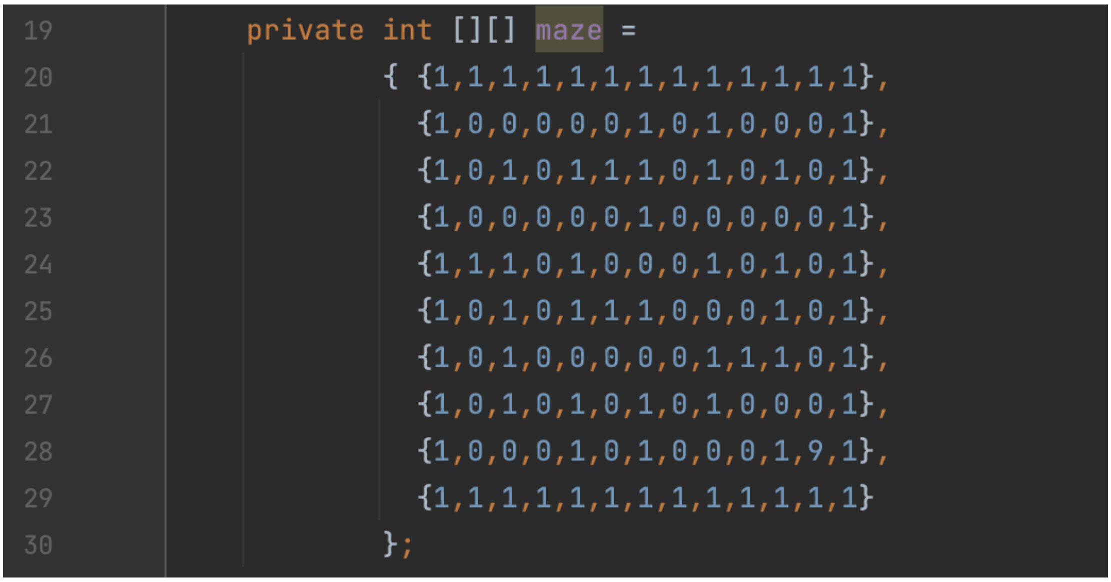
Name/NRP : Mardhatillah Shevy Ananti / 5025211070
Kirana Alivia Enrico / 5025211190
Talitha Hayyinas Sahala / 5025211263

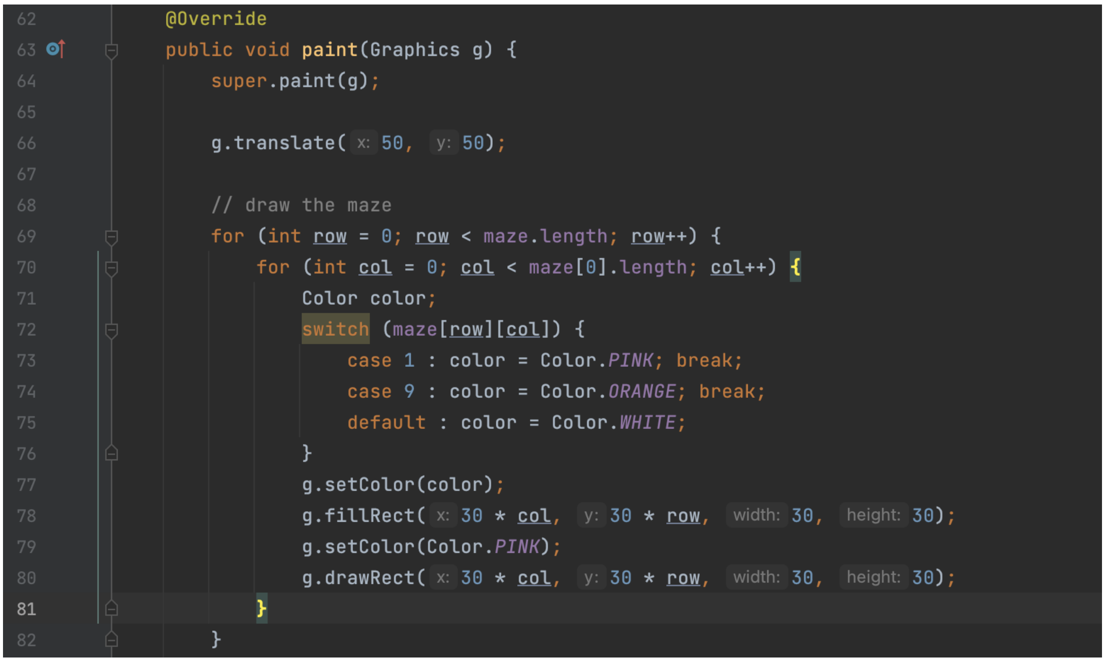
View class

1. View class is the main class. Use two dimensional array to represent the maze (graph). Maze [row][col] with the following values:
   0 = not-visited node
   1 = visited node
   2 = target node
   In this implementation, borders must be filled with “1” to avoid ArrayIndexOutOfBounds.
   Make the starting position as (1, 1) and define the target node as “9” at position (11,8)



```java
    private int [][] maze =
            { {1,1,1,1,1,1,1,1,1,1,1,1,1},
              {1,0,0,0,0,0,1,0,1,0,0,0,1},
              {1,0,1,0,1,1,1,0,1,0,1,0,1},
              {1,0,0,0,0,0,1,0,0,0,0,0,1},
              {1,1,1,0,1,0,0,0,1,0,1,0,1},
              {1,0,1,0,1,1,1,0,0,0,1,0,1},
              {1,0,1,0,0,0,0,0,1,1,1,0,1},
              {1,0,1,0,1,0,1,0,1,0,0,0,1},
              {1,0,0,0,1,0,1,0,0,0,1,9,1},
              {1,1,1,1,1,1,1,1,1,1,1,1,1}
            };
```

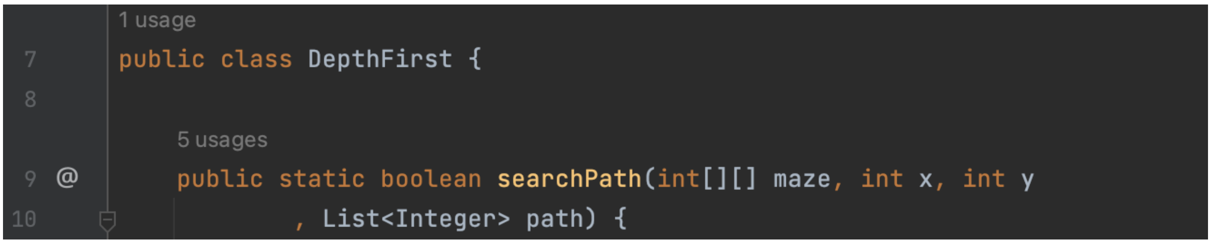
2. Draw the maze



```java
    @Override
    public void paint(Graphics g) {
        super.paint(g);

        g.translate( x: 50,  y: 50);

        // draw the maze
        for (int row = 0; row < maze.length; row++) {
            for (int col = 0; col < maze[0].length; col++) {
                Color color;
                switch (maze[row][col]) {
                    case 1 : color = Color.PINK; break;
                    case 9 : color = Color.ORANGE; break;
                    default : color = Color.WHITE;
                }
                g.setColor(color);
                g.fillRect( x: 30 * col,  y: 30 * row,  width: 30,  height: 30);
                g.setColor(Color.PINK);
                g.drawRect( x: 30 * col,  y: 30 * row,  width: 30,  height: 30);
            }
        }
```

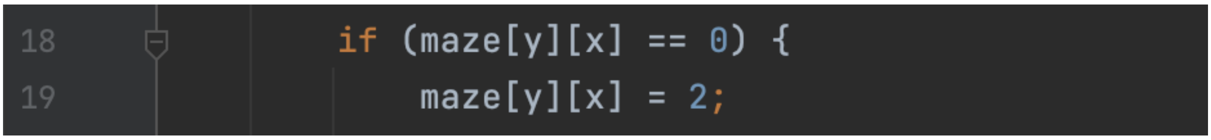
DepthFirst class

1. If path was found, this method will return true and the path list will be filled. Like this: { xn, yn, . . . , x2, y2, x1, y1}, so the order is inverted x and y are the start searching position



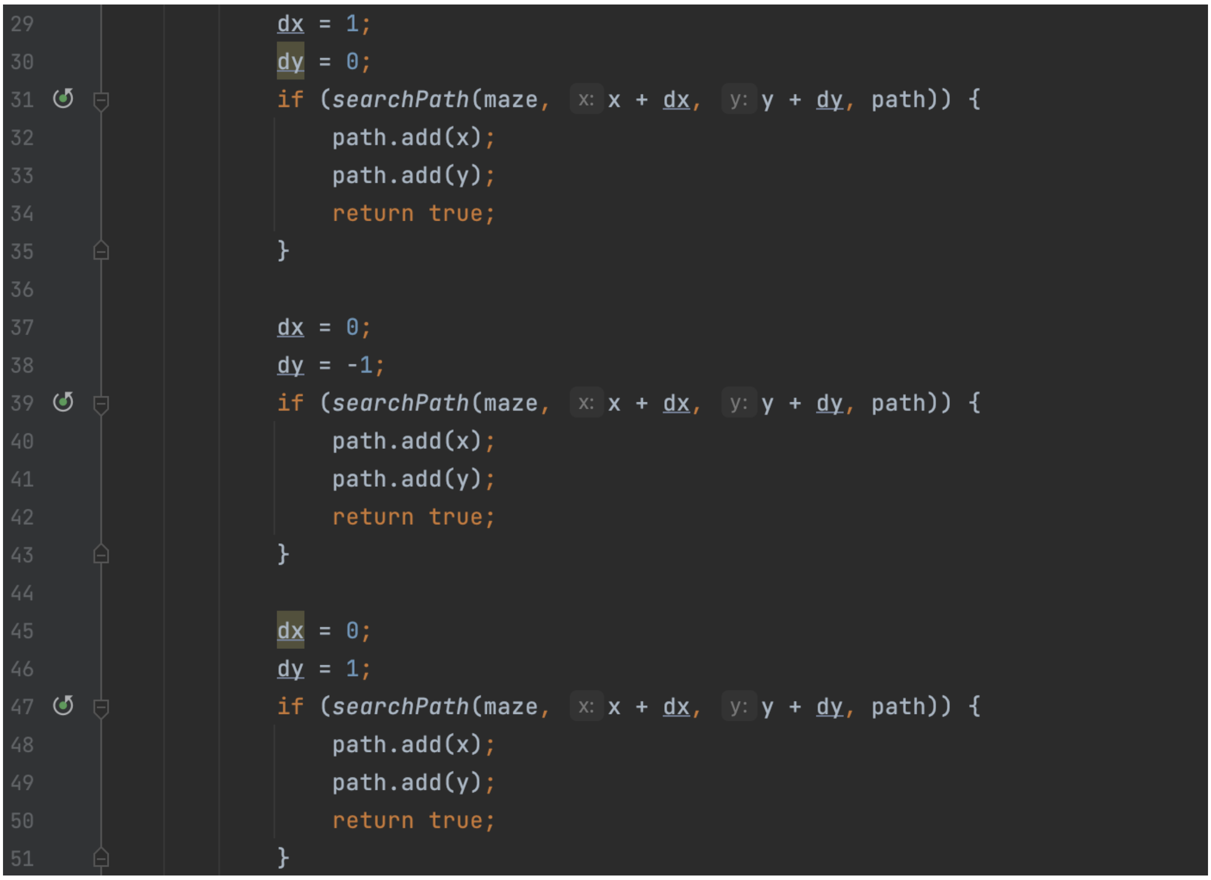
```
1 usage
public class DepthFirst {

    5 usages
    public static boolean searchPath(int[][] maze, int x, int y
            , List<Integer> path) {
```

2. When the current position (x and y) is a not-visited node (0) , then mark it as visited (2)



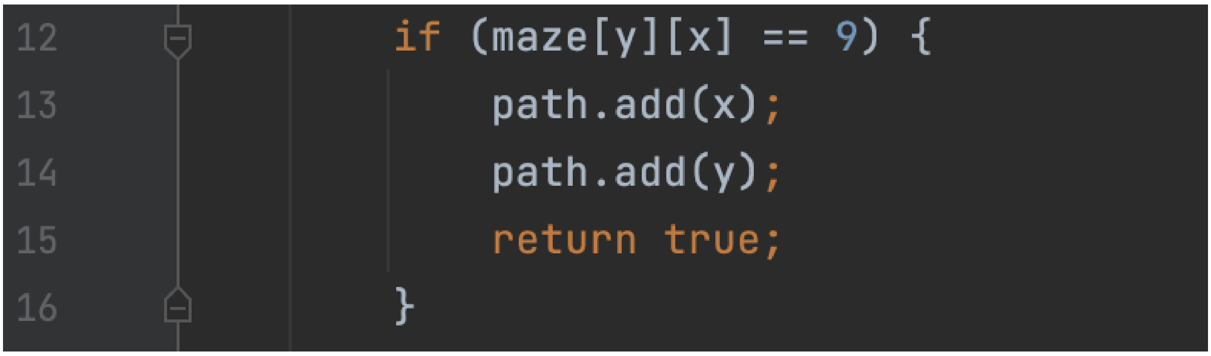
```
        if (maze[y][x] == 0) {
            maze[y][x] = 2;
```

3. Visit all neighbour nodes recursively. If path was found, fill the path list with current position



```
            dx = 1;
            dy = 0;
            if (searchPath(maze, x: x + dx, y: y + dy, path)) {
                path.add(x);
                path.add(y);
                return true;
            }

            dx = 0;
            dy = -1;
            if (searchPath(maze, x: x + dx, y: y + dy, path)) {
                path.add(x);
                path.add(y);
                return true;
            }

            dx = 0;
            dy = 1;
            if (searchPath(maze, x: x + dx, y: y + dy, path)) {
                path.add(x);
                path.add(y);
                return true;
            }
```

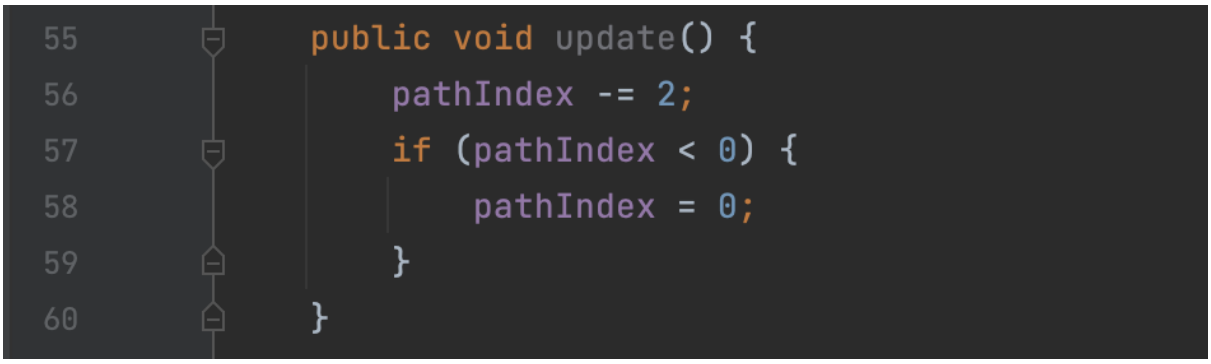
4. Check if the target node was reached



```
if (maze[y][x] == 9) {
    path.add(x);
    path.add(y);
    return true;
}
```

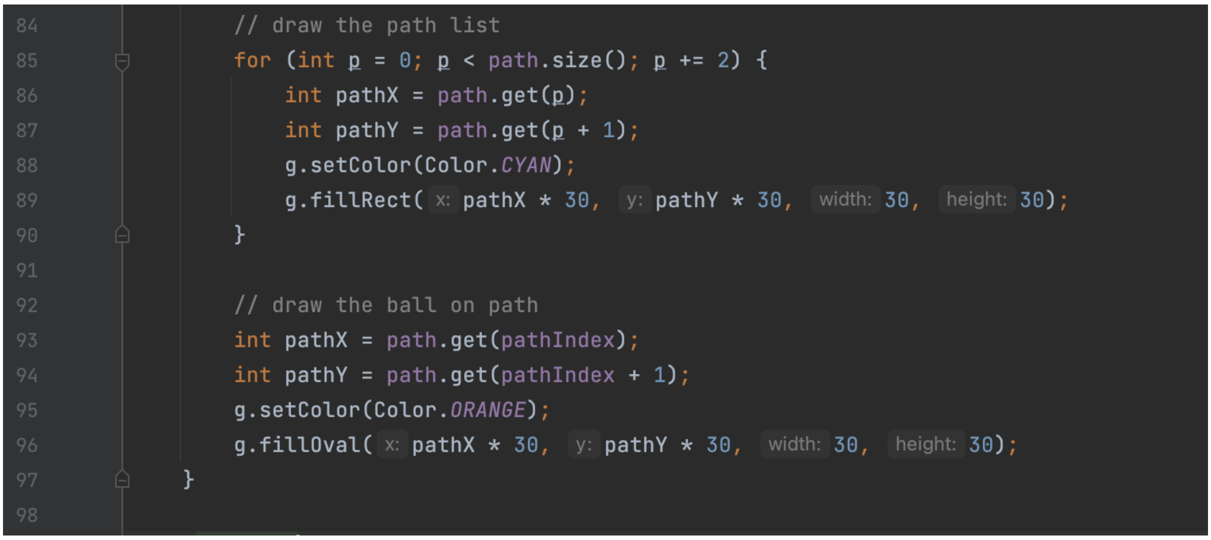
Implement DFS in View

1. Test DepthFirst, Checking the first and the last item of path list



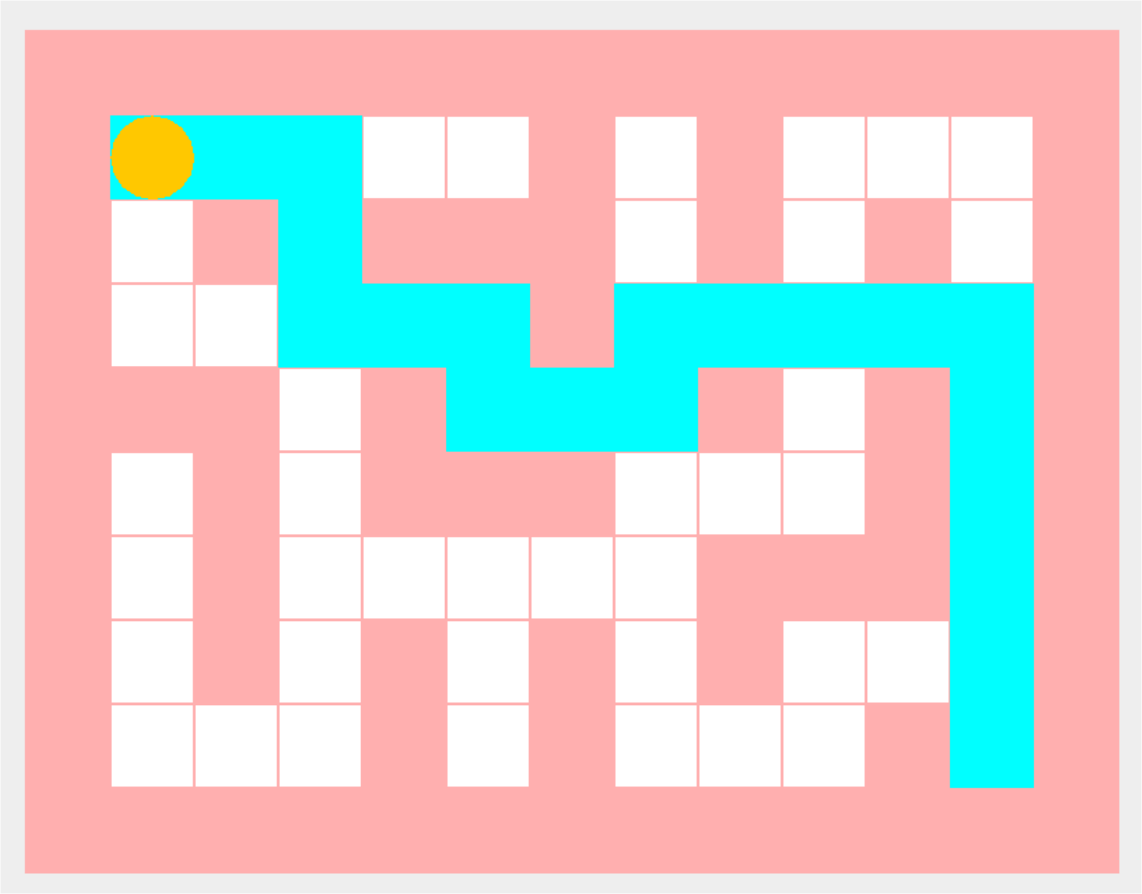
```
public void update() {
    pathIndex -= 2;
    if (pathIndex < 0) {
        pathIndex = 0;
    }
}
```

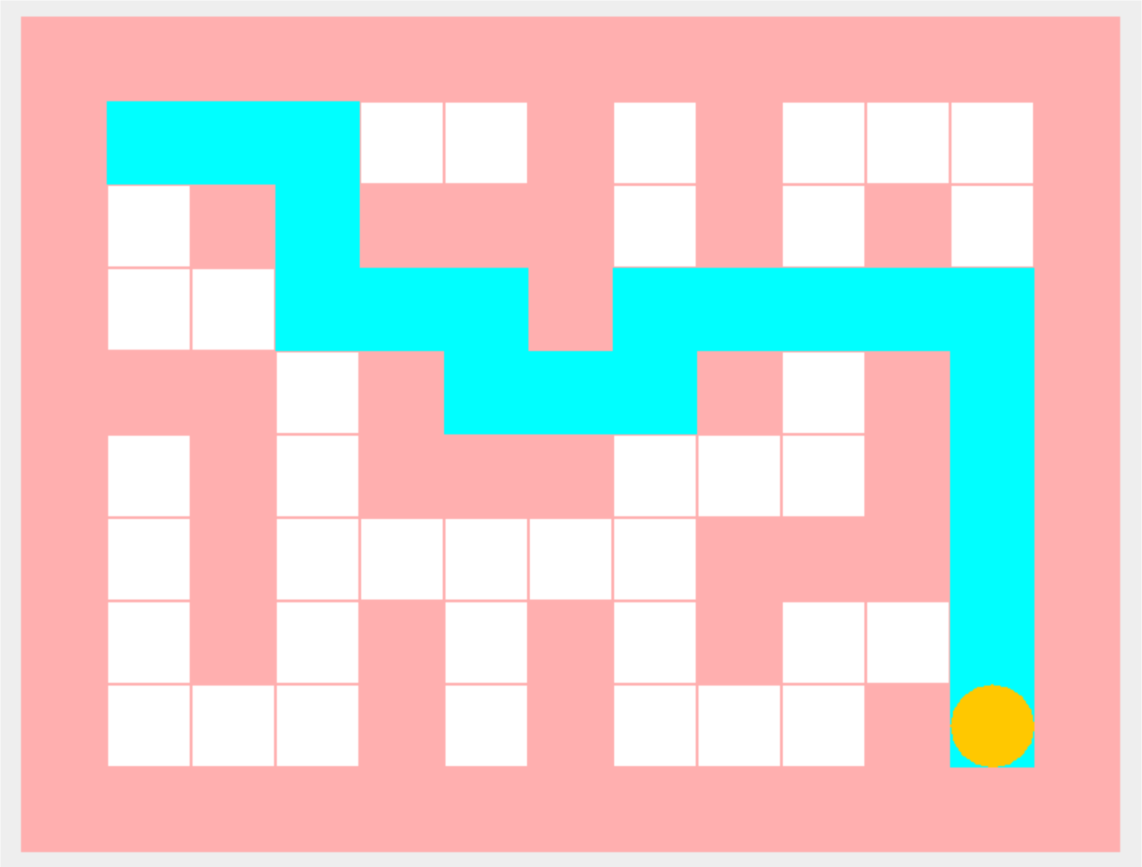
2. Finally, draw the path list



```
    // draw the path list
    for (int p = 0; p < path.size(); p += 2) {
        int pathX = path.get(p);
        int pathY = path.get(p + 1);
        g.setColor(Color.CYAN);
        g.fillRect( x: pathX * 30,  y: pathY * 30,  width: 30,  height: 30);
    }

    // draw the ball on path
    int pathX = path.get(pathIndex);
    int pathY = path.get(pathIndex + 1);
    g.setColor(Color.ORANGE);
    g.fillOval( x: pathX * 30,  y: pathY * 30,  width: 30,  height: 30);
}
```

Output





“By the name of Allah (God) Almighty, herewith I pledge and truly declare that I have solved quiz 2 by myself, did not do any cheating by any means, did not do any plagiarism, and did not accept anybody’s help by any means. I am going to accept all of the consequences by any means if it has proven that I have done any cheating and/or plagiarism.”

Surabaya, 23 November 2022

Mardhatillah Shevy Ananti	Kirana Alivia Enrico	Talitha Hayyinas Sahala
5025211070	5025211190	5025211263

Mardha 33.33% [view.java, analysis pdf], Kirana 33.33% [view.java, analysis pdf], Talitha 33.33% [DepthFirst.java, analysis pdf]