ISE307 IT Systems Analysis and Design Homework 3

Ahmet Furkan TEKE - 150120202

Point Class

```
package hw3;

public class Point {
    private int x_coordinate;
    private int y_coordinate;
    public Point(int x, int y){
        x_coordinate = x;
        y_coordinate = y;
    }
    public void setPoint(int x, int y){
        x_coordinate = x;
        y_coordinate = y;
    }
    public void setX(int x){
        x_coordinate = x;
    }
    public void setY(int y){
        y_coordinate = y;
    }
    public int getX(){
        return x_coordinate;
    }
    public int getY(){
        return y_coordinate;
    }
}
```

Robot Class

Direction keeps direction desired to go, N = North, W = West, S = South, E = East

```
package hw3;

public class Robot{
    private String direction;
    protected Point point;

    public Robot(){
        direction = "E";
        point = new Point(0, 0);
    }
    public void turnLeft(){
        switch (getDirection()) {
        case "E":  direction = "N";
            break;
        case "N":  direction = "W";
            break;
        case "W":  direction = "S";
            break;
        case "S":  direction = "E";
            break;
        }
    }
    public void turnRight(){
        switch (getDirection()) {
        case "E":  direction = "S";
            break;
        case "S":  direction = "W";
            break;
        case "W":  direction = "N";
            break;
        case "N":  direction = "E";
            break;
        }
    }
```

Move is on infinitive space, no checking

Returned string is going to use by label

```
public String move(){
    int x = point.getX();
    int y = point.getY();
    switch(getDirection()){
    case "E":  point.setX(x+1);
        break;
    case "S":  point.setY(y+1);
        break;
    case "W":  point.setX(x-1);
        break;
    case "N":  point.setY(y-1);
        break;
    }
    return "Robot moved to x: " + point.getX() + ", y: " + point.getY();
}
public Point getLocation(){
    return point;
}
public String getDirection(){
    return direction;
}
```

GridRobot Class

Uses super class constructor

Move checks the desired move is legal or not

Returns message to use by label

```
public class GridRobot extends Robot{
    private String name;
    private final int GRID_WIDTH = 20;
    private final int GRID_HEIGHT = 20;

    public GridRobot(String name){
        super();
        this.name = name;
    }

    public String move(){
        int x = point.getX();
        int y = point.getY();
        switch(getDirection()){
        case "E":
            if (x+1 < GRID_WIDTH)
                point.setX(x+1);
            else
                return "The Robot X can't move in the EAST direction since it is outside the grid";
            break;
        case "S":
            if (y+1 < GRID_HEIGHT)
                point.setY(y+1);
            else
                return "The Robot X can't move in the SOUTH direction since it is outside the grid";
            break;
        case "W":
            if (x-1 >= 0)
                point.setX(x-1);
            else
                return "The Robot X can't move in the WEST direction since it is outside the grid";
            break;
        case "N":
            if (y-1 >= 0)
                point.setY(y-1);
            else
                return "The Robot X can't move in the NORTH direction since it is outside the grid";
            break;
        }
        return "Robot moved to x: " + point.getX() + ", y: " + point.getY();
    }
```

To string is overrided

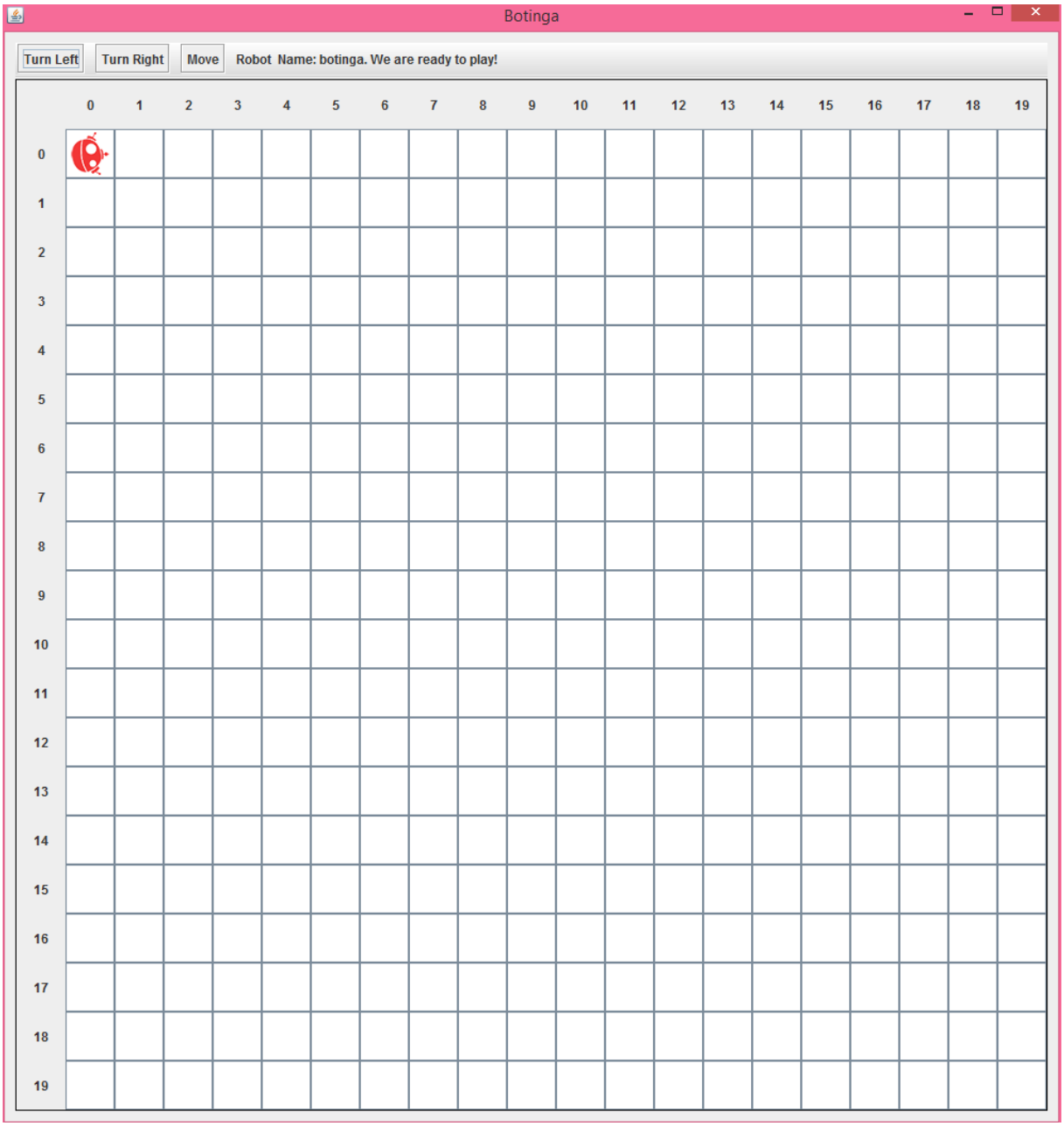
```
  public String toString(){
      return " Name: " + name;
  }
  public int getWidth(){
      return GRID_WIDTH;
  }
  public int getHeight(){
      return GRID_HEIGHT;

  }
```

Panel Class (Main & GUI)

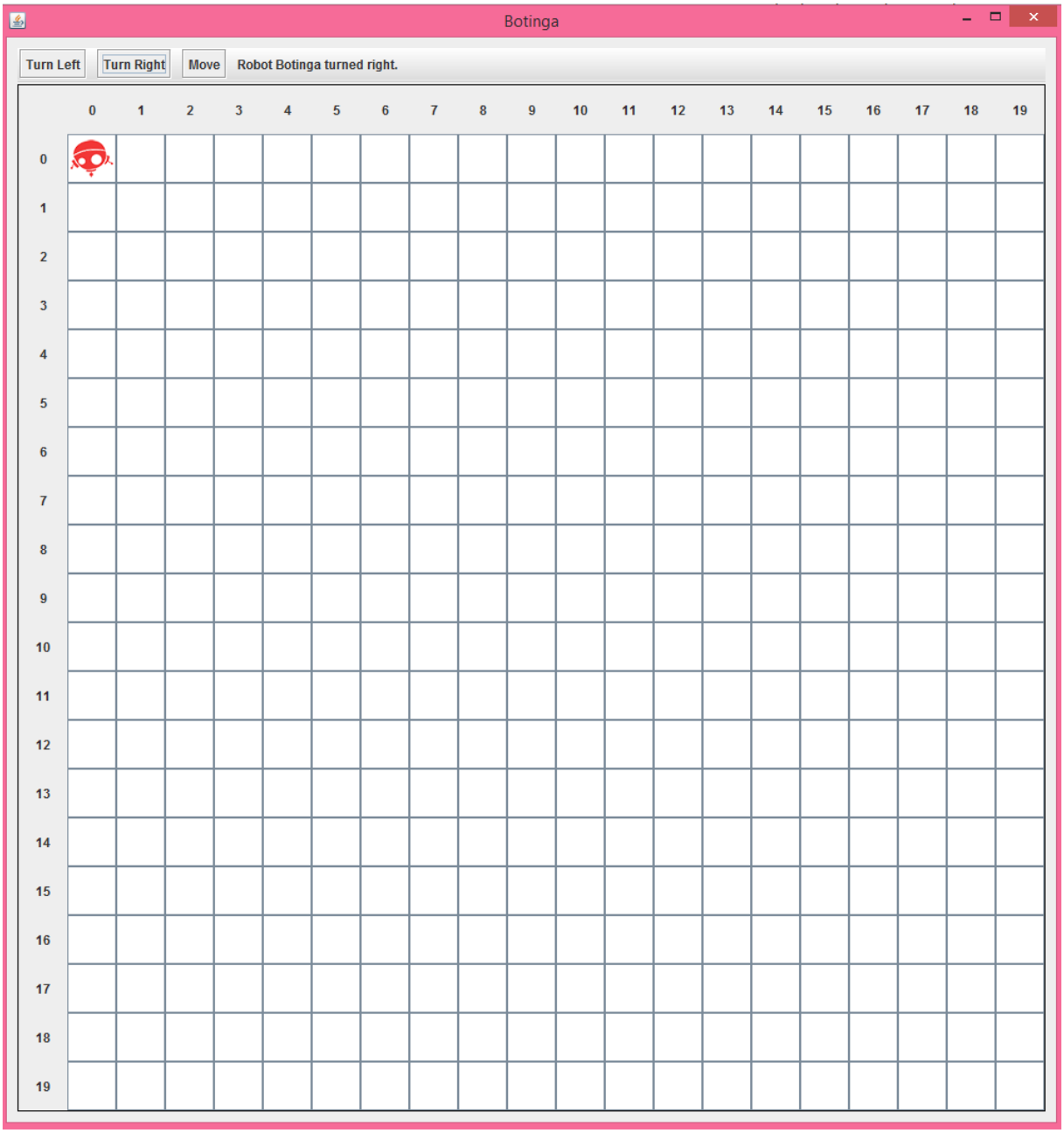
I used images to represent directions

EAST NORTH SOUTH WEST

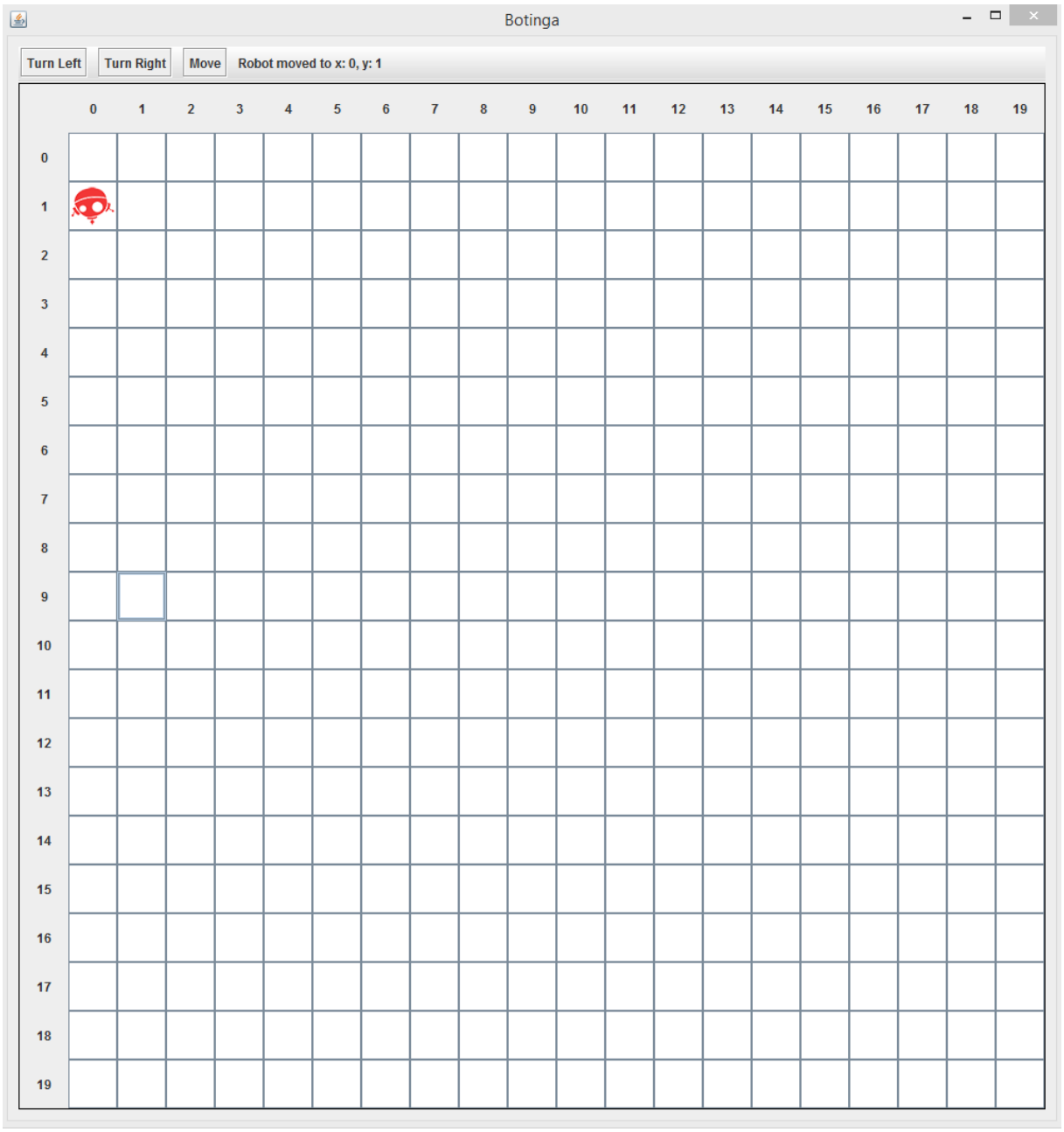
INITIAL PAGE



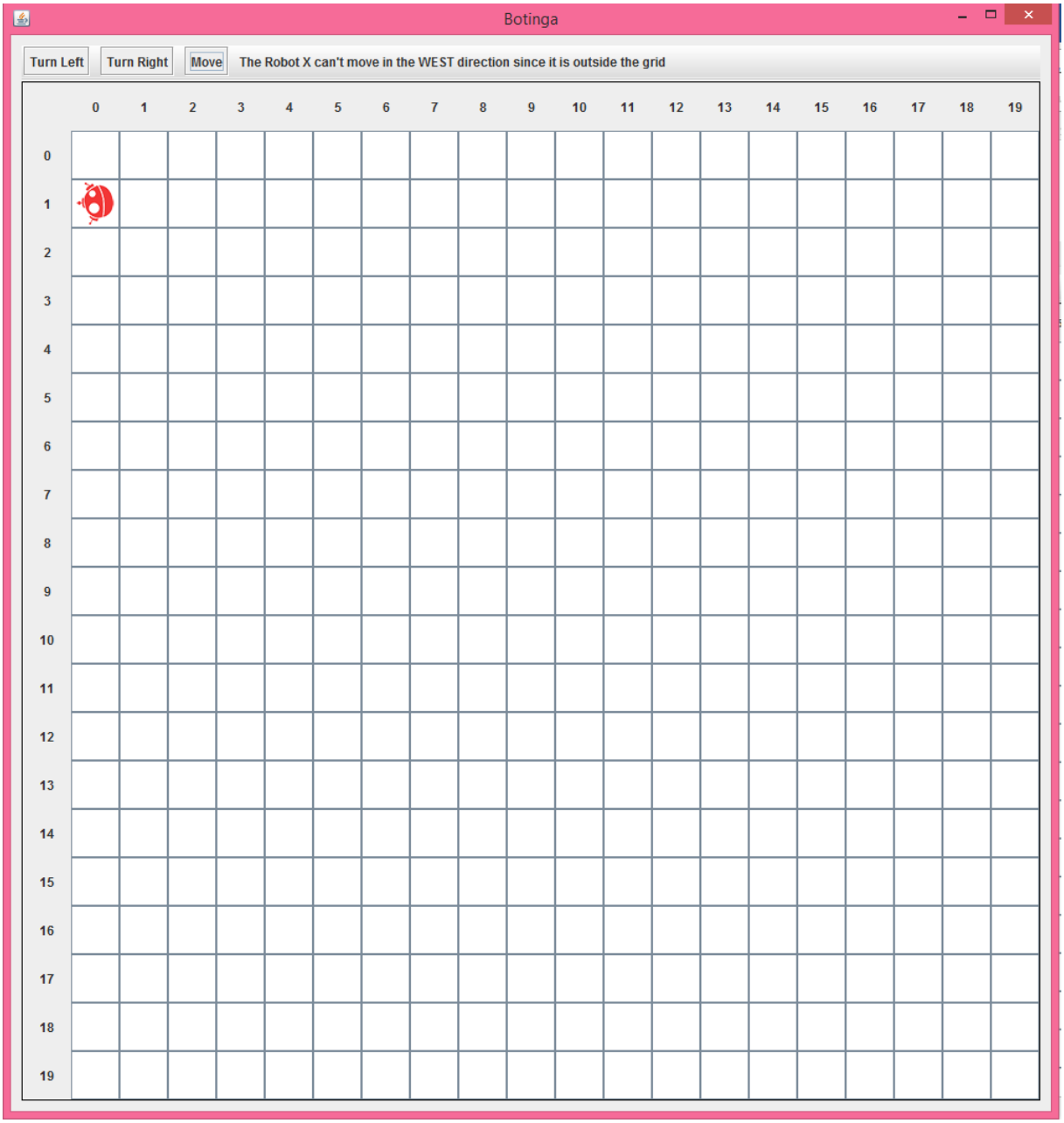
Turn Right


Botinga
Turn Left
Turn Right
Move
Robot Botinga turned right.
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Move (Legal Move)



Illegal Move



```java
package hw3;

import java.awt.*;

public class Panel {
    private final JPanel gui = new JPanel(new BorderLayout(2, 2));
    static GridRobot robot;
    private JButton[][] boardSquares; // each square will be button
    private JPanel board; // board to put buttons
    private JLabel message = new JLabel("Robot " + robot.toString() + ". We are ready to play!"); // status message

    Panel(int width, int height) {
        boardSquares = new JButton[width][height];
        initializeGui(width, height);
    }

  public final void initializeGui(int width, int height) {
      gui.setBorder(new EmptyBorder(10, 10, 10, 10));
      JToolBar tools = new JToolBar();
      tools.setFloatable(false);
      gui.add(tools, BorderLayout.PAGE_START);

      // Turn left button
      JButton btn = new JButton("Turn Left");
      btn.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
              robot.turnLeft();
              Insets buttonMargin = new Insets(0,0,0,0);
              int x = robot.getLocation().getX();
              int y = robot.getLocation().getY();
              boardSquares[x][y].setMargin(buttonMargin);
              ImageIcon icon = new ImageIcon("img/"+robot.getDirection()+".png");
              boardSquares[x][y].setIcon(icon);
              boardSquares[x][y].setBackground(Color.WHITE);
              message.setText("Robot Botinga turned left.");
          }
      });
      tools.add(btn);
      tools.addSeparator();

     // Turn right button
     btn = new JButton("Turn Right");
     tools.add(btn);
     btn.addActionListener(new ActionListener(){
         public void actionPerformed(ActionEvent e){
             robot.turnRight();
             Insets buttonMargin = new Insets(0,0,0,0);
             int x = robot.getLocation().getX();
             int y = robot.getLocation().getY();
             boardSquares[x][y].setMargin(buttonMargin);
             ImageIcon icon = new ImageIcon("img/"+robot.getDirection()+".png");
             boardSquares[x][y].setIcon(icon);
             boardSquares[x][y].setBackground(Color.WHITE);
             message.setText("Robot Botinga turned right.");
         }
     });
     tools.addSeparator();

     // Move button
     btn = new JButton("Move");
     tools.add(btn);
     btn.addActionListener(new ActionListener(){
         public void actionPerformed(ActionEvent e){
             Insets buttonMargin = new Insets(0,0,0,0);
             // Old place
             int x = robot.getLocation().getX();
             int y = robot.getLocation().getY();
             ImageIcon icon = new ImageIcon(new BufferedImage(40, 40, BufferedImage.TYPE_INT_ARGB));
             boardSquares[x][y].setIcon(icon);
             boardSquares[x][y].setBackground(Color.WHITE);
             // New place
             String answer = robot.move();
             message.setText(answer);
             x = robot.getLocation().getX();
             y = robot.getLocation().getY();
             boardSquares[x][y].setMargin(buttonMargin);
             icon = new ImageIcon("img/"+robot.getDirection()+".png");
             boardSquares[x][y].setIcon(icon);
             boardSquares[x][y].setBackground(Color.WHITE);
         }
```

```
  tools.addSeparator();
  tools.add(message);
  board = new JPanel(new GridLayout(0, width+1));
  board.setBorder(new LineBorder(Color.BLACK));
  gui.add(board);

  Insets buttonMargin = new Insets(0,0,0,0);
  for (int i = 0; i < boardSquares.length; i++) {
      for (int j = 0; j < boardSquares[i].length; j++) {
          JButton b = new JButton();
          b.setMargin(buttonMargin);
          // our  pieces are 40x40 px in size, so we'll
          ImageIcon icon = new ImageIcon(new BufferedImage(40, 40, BufferedImage.TYPE_INT_ARGB));
          b.setIcon(icon);
          b.setBackground(Color.WHITE);
          boardSquares[j][i] = b;
      }
  }
  //put robot to board
  JButton b = new JButton();
  b.setMargin(buttonMargin);
  ImageIcon icon = new ImageIcon("img/"+robot.getDirection()+".png");
  b.setIcon(icon);
  b.setBackground(Color.WHITE);
  boardSquares[robot.getLocation().getX()][robot.getLocation().getY()] = b;

board.add(new JLabel(""));
// fill the top row
for (int i = 0; i < width; i++) {
    board.add(new JLabel("" + i, SwingConstants.CENTER));
}
// fill the  non-pawn piece row
for (int i = 0; i < height; i++) {
    for (int j = 0; j < height; j++) {
        switch (j) {
            case 0:
                board.add(new JLabel("" + i, SwingConstants.CENTER));
            default:
                board.add(boardSquares[j][i]);
        }
    }
}
```

```
public final JComponent getGui() {
    return gui;
}

public static void main(String[] args) {
    robot = new GridRobot("botinga");
    Runnable r = new Runnable() {
        public void run() {
            int width = robot.getWidth();
            int height = robot.getHeight();

            Panel cb = new Panel(width, height);

            JFrame f = new JFrame("Botinga");
            f.add(cb.getGui());
            f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            f.setLocationByPlatform(true);

            f.pack();
            f.setVisible(true);

        }
    };
    SwingUtilities.invokeLater(r);
}
```