CS102 – Algorithms and Programming II
Programming Assignment 4
Spring 2025

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention, where you replace the variables “Sec1”, “Surname” and “Name” with your actual section, surname and name:

  CS102_Sec1_Asgn4_Surname_Name.zip
- Upload the above zip file to Moodle by the deadline (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TAs and tutors during the lab. You have to make the preliminary submission before the lab day. After the TA checks your work in the lab, you will make your final submission. Even if your code does not change in between these two versions, you should make two submissions for each assignment.

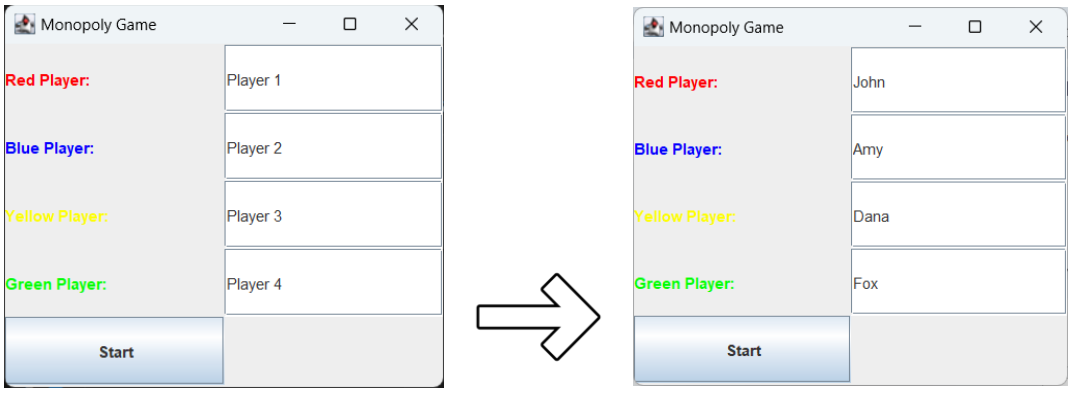
GRADING WARNING:

- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

Monopoly with GUI

For this assignment, you will implement the GUI version of the Simple Monopoly Game from the first lab assignment. You are allowed to utilize your code from the first lab for the game logic. Note that how GUIs operate is different from the console application logic, so you need to make changes in how your game loop operates. The rules of the game are the same, but you need to utilize GUI elements to create a more enjoyable experience.

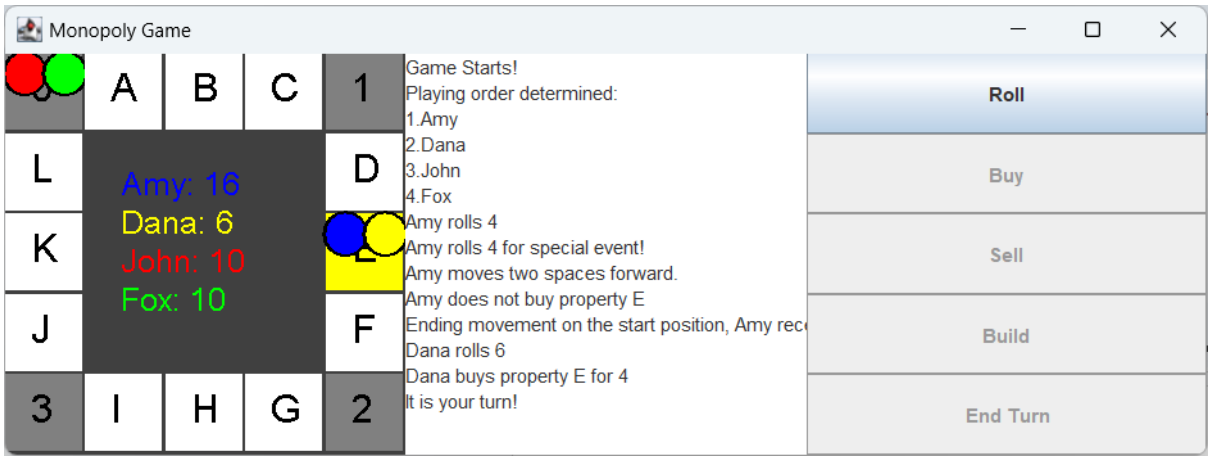
Your application should start with a welcome screen where we enter the player names. You should use GridLayout (*java.awt.GridLayout*) to properly display the player labels and text fields to enter the player names. Four players will compete in the game; the red player will be controlled by the user, and the rest of the players will be controlled by the computer. You may use some placeholder text (like Player 1, Player 2, etc.) in your text fields so that the user can fill in the names. You may use the setForeground(Color c) method of JLabel to color the text color of the labels. Try to achieve the following UI using appropriate elements.



When we click the start button, you should hide this frame to open a new one. To give functionality to a button, we use ActionListeners. Suppose we have a JButton that we want to print on the console when we click on it, you may use the following shorthand to attach a new ActionListener to this button with its actionPerformed method overridden:

```
JButton button = new JButton("My Button");
button.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      System.out.println("Hey!");
   }
});
```

You should also add this button to a frame or a panel to be able to use it. The frame that opens when we click the start button will be the main screen where we play the game. This screen should have the game map (that includes the cells, player positions, and player coins), the game information screen (a JTextArea that acts as a log of the actions players take), and the action buttons (roll, buy, sell, build, and end turn). Use the GridLayout to divide the frame accordingly.



The map screen requires you to draw 2D graphics on a panel. To this end, you may extend the JPanel class and override its *paintComponent* method. For example, the following class will paint a rectangle on the panel if you add it into a JFrame:

```
public class MyPanel extends JPanel {
   @Override
   protected void paintComponent(Graphics g) {
    g.setColor(Color.DARK_GRAY); // set drawing color
    g.fillRect(20, 20, 30, 30); // draw filled rectangle
   }
}
```

You should also set the desired size for the panel and use the pack method to make the JFrame show the panel in the desired size. Otherwise, since we do not have UI elements in it, the default panel size will not be visible. You may use the following code segment to make your panel visible in the frame:

```
JFrame myFrame = new JFrame("Main Frame")
MyPanel m = new MyPanel();
m.setPreferredSize(new Dimension(250, 250));
myFrame.add(m);
myFrame.pack();
myFrame.setVisible(true);
```

Graphics object includes methods such as fillOval, fillRect, drawString, and setFont, which you can utilize to achieve the desired look. The method setFont requires you to input the name of the font, its style, and size. For example, you may use the following code segment to have a font with size 12:

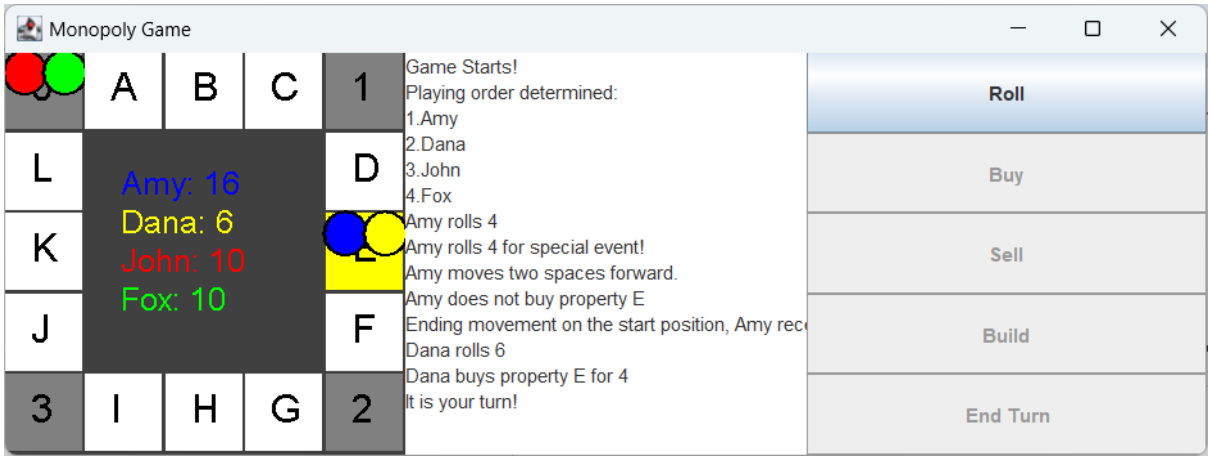
```
setFont(new Font("Arial", Font.PLAIN, 12));
```

You may include a draw method in the class that represents the map cells to draw each cell accordingly; this would require you to pass the Graphics parameter to the classes that will use it to draw. The Graphics object is created by the UI classes; to be able to see the graphics on the panel, we need to use that specific Graphics object. If we create a Graphics object manually, it will not be attached to the panel, so drawing on it will not show the graphics on the screen.

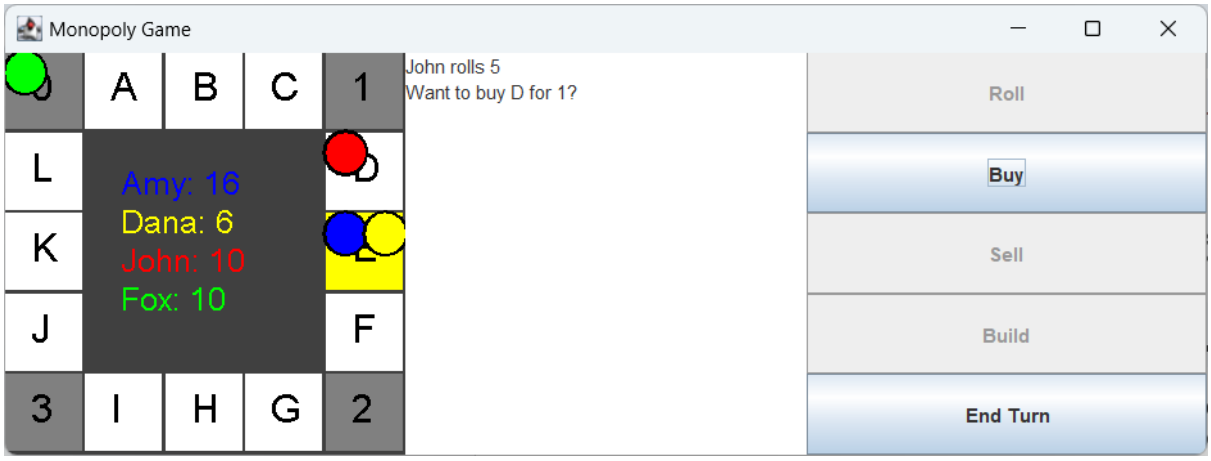
The middle part of the frame will include an information screen that will include the log of the previous actions. You may use a JTextArea here, which supports multiple lines. Normally, JTextArea is used to input text, to prevent the user from editing the text of the JTextArea, you may use its `setEditable(false)` method. This class includes `append(String s)` method to add new text to the JTextArea. You need to add the end line character (`"\n"`) manually to have your text in different lines. You can use `setText(String s)` method to replace its content. Clear the log each time the human player's turn ends so it does not get too crowded.

The buttons will be enabled or disabled based on which actions the human player can currently take. While computer players operate, all the buttons will be disabled. When the human player's turn starts, enable the roll button. The player cannot end the turn before rolling. After rolling, move the player accordingly (to update the view of your map, you may need to call the `repaint()` method of the panel so that the view refreshes). If we end up on a property without an owner and if we have enough coins, the buy button will be enabled. The sell button is also enabled if the player has at least one property to sell. Similarly, the sell button is also enabled if we have at least one property with a place to build a house. After rolling, the end turn button is enabled nevertheless. Clicking on the end turn button ends the user's turn, so the computer players do their actions.

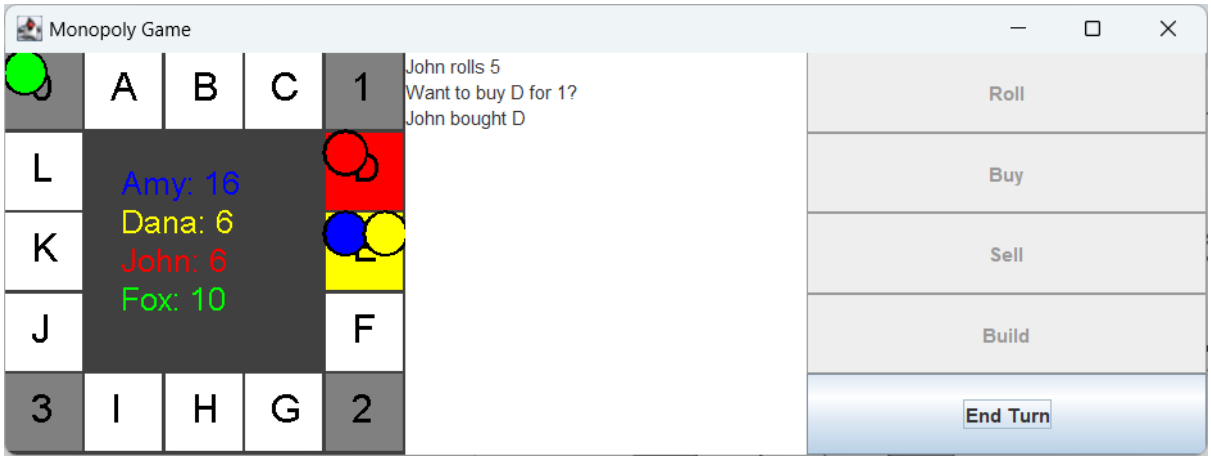
Clicking on the buy button buys the current property, decreasing its price from the player's coins. You will also color the cells based on the owner of the property. When we start the game, the playing order is determined. If the human player does not start first, the computer players will do their actions before the user takes any action. For example, after entering the names and starting the game, we may get the following screen:



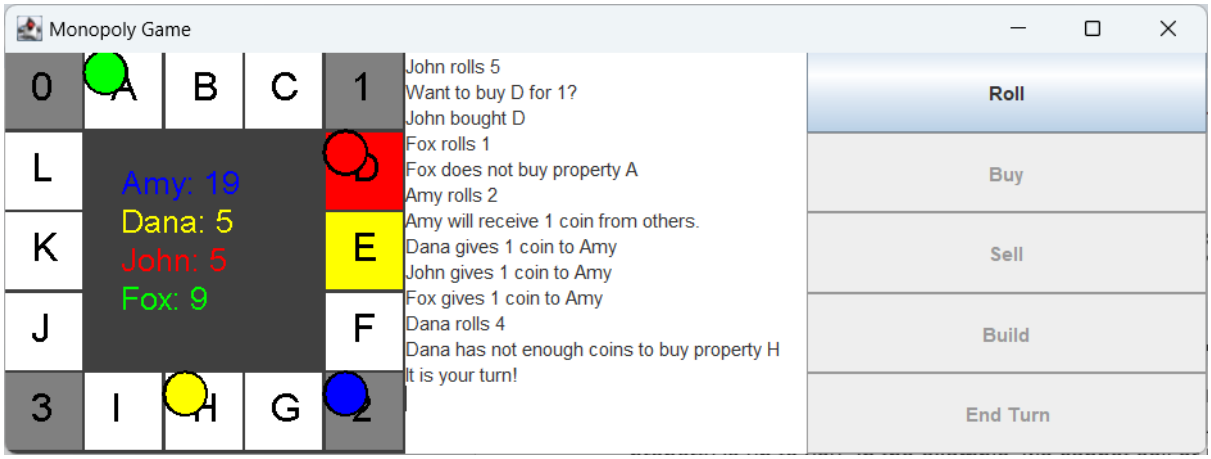
Then, we can click on the roll button to roll the die. This clears the log so that we can fit all the messages on the information screen. Since we have enough coins, we can buy property D.



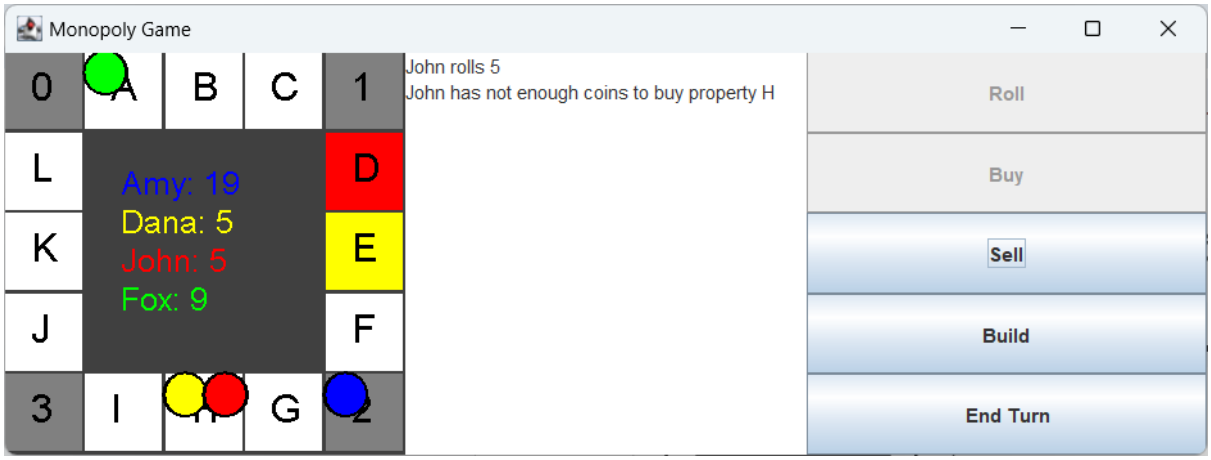
After clicking on the buy button, the color of the cell changes, our coins decrease, and the buy button is disabled. Whether you allow selling and building on the first turn we buy a property is up to you. In the example, we cannot sell or build if it is the only house we bought on this turn, so the sell and build buttons are not enabled. Then we need to end turn.



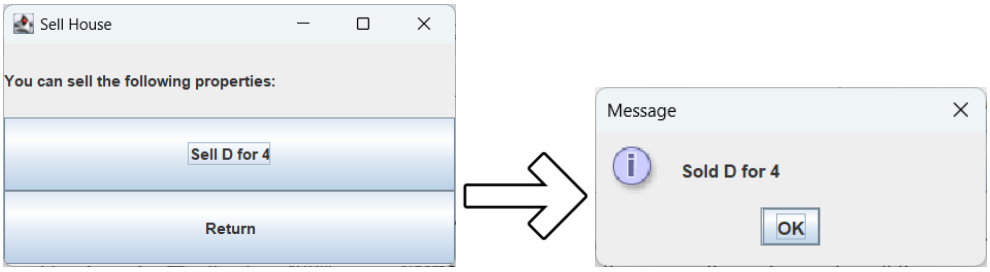
After ending our turn, other players take their actions, and now it is our turn again.



We roll, but we do not have enough coins to buy a new place, so the buy button is not enabled. Since we bought a property earlier, we can now sell or build. Clicking on sell or build opens a new frame that lists the available properties to sell or build on.



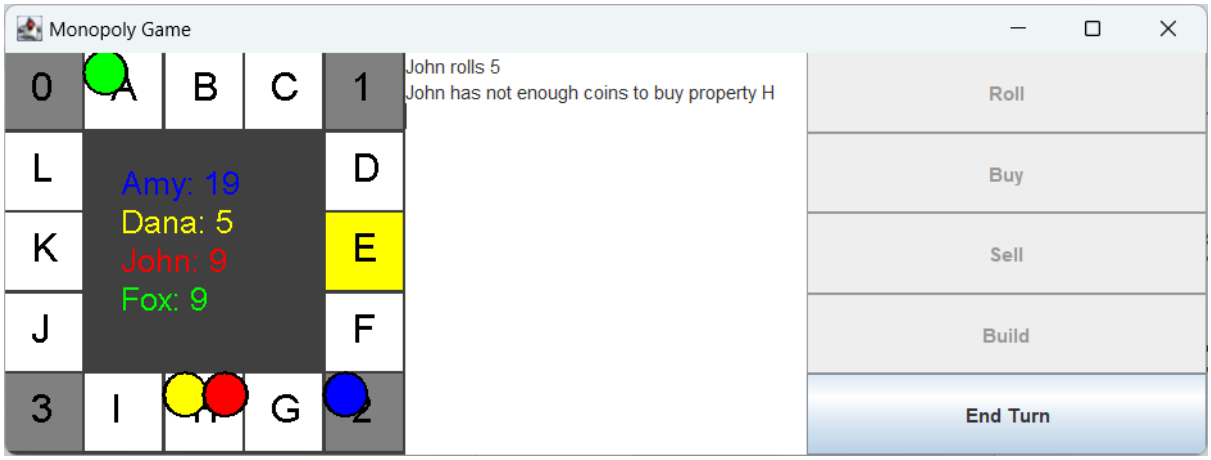
Suppose we click on the sell button, then we get the following frame. The return button simply closes this frame so that we continue the game. If we choose to sell the property D, we click the corresponding button. This opens a pop-up message indicating that we sold the property.



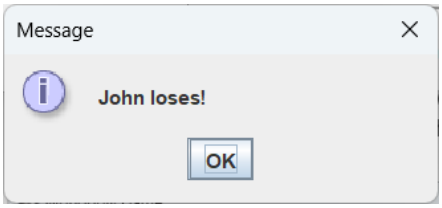
To this end, you may use the following code to show a one-time pop-up message:

JOptionPane.showMessageDialog(null, "My Message!");

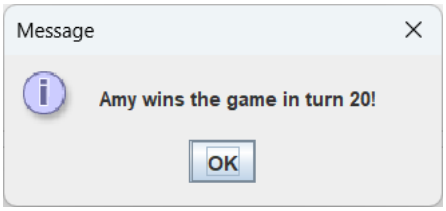
After clicking on the OK button, we will return to the main game view. Since we sold our only property, the build button is disabled as well as the sell button. In one turn, we can only sell one property and we can only build one house on a property we own.



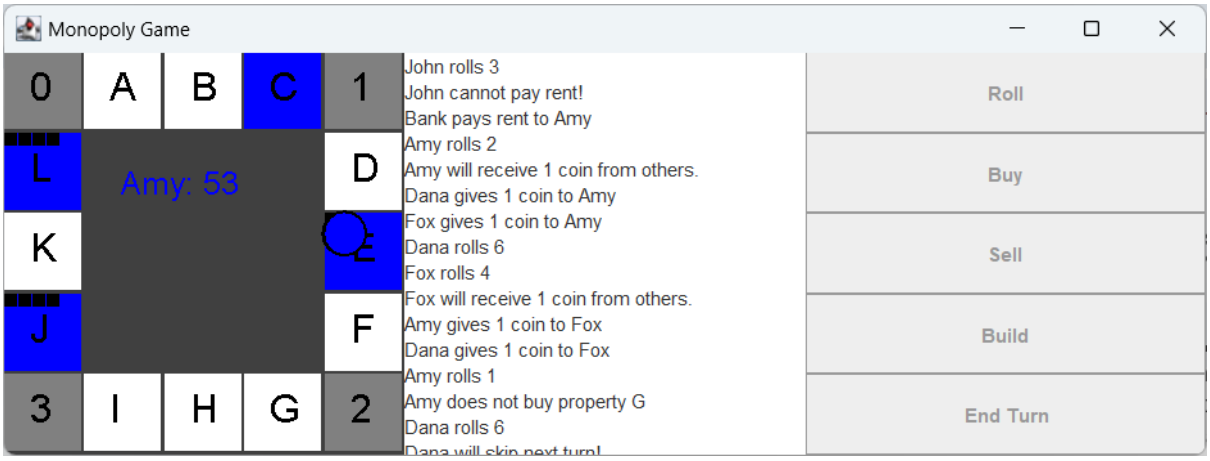
Suppose we play the game for some turns, and one of the players loses by having coins less than zero. Then, you should display a pop-up message indicating who lost.



If the human player loses, the rest of the game plays automatically. The winner is shown at the end with a different pop-up message. Also, display the turn number when the game ends.

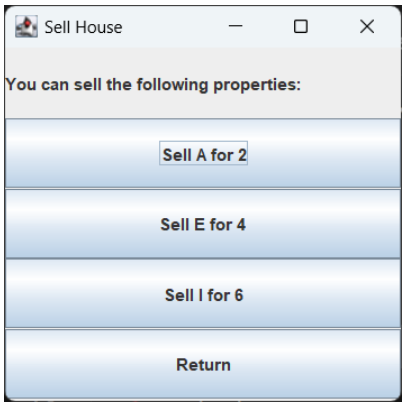


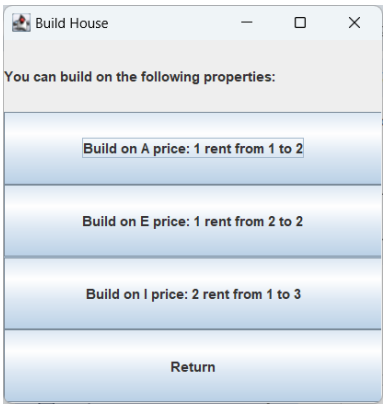
Clicking on the OK button on the pop-up message for the winning player, we see the final view of the game. Since the players that lose also lose their properties, the final view should only have the properties of the winner. If the game ends with a tie, the final view may contain multiple players and properties owned by those players.



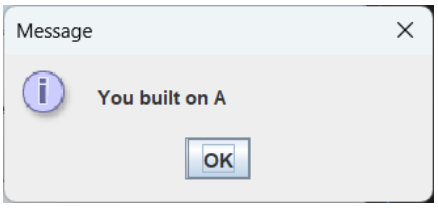
We could exit the game when it finishes; you may also return to the starting screen at the end. You should indicate the number of houses on the properties owned by players. In the given examples, this is indicated with the little black squares at the top left corner of the related cells. You may also use numbers for this purpose.

When selling or building, the number of buttons you show on the sell or build frames will change based on the number of available properties.





Similar to the pop-up window when we sell a house, you should show a message if the player chooses to build on a property.



You may have a different look for your game as long as the desired interaction is achieved. For example, your map cells could use different shapes, you may write the property letters at different positions, your action buttons (buy, sell, build, etc.) could use a different layout. For this assignment, console interaction is not allowed, we should play the game using the GUI only.

Preliminary Submission: You will submit an early version of your solution before the final submission. This version should at least include the following:

- The starting screen where we enter player names should be complete.
- You should be able to draw the map with the players on it.
- The players should be able to roll and move on the map.

You will have time to complete your solution after you submit your preliminary solution. You can consult the TAs and tutors during the lab. Do not forget to make your final submission at the end. Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

Not completing the preliminary submission on time results in 50% reduction of this assignment's final grade.