import com.Ostermiller.util.\*;

import java.io.\*;

import java.util.\*;

import javax.swing.\*;

import java.awt.\*;

import javax.swing.JOptionPane;

import java.lang.Math.\*;

import javax.swing.GroupLayout;

import javax.swing.LayoutStyle;

import java.awt.event.\*;

public class MONOPOLY

{

protected int activePlayer;

public int desiredProperty;

public int player[][];

public int die1;

public int die2;

public int diceTotal;

public int street;

public int houses[];

public int hotels[];

public int freeParking;

public boolean same;

public String name1;

public String name2;

public int position;

public int position1;

public int position2;

public int balance1;

public int balance2;

public boolean onProperty;

public boolean turnC;

public boolean owned;

// instance variables

// file handling IN: Property Cost

private PROPERTIES PropertyCost[];

private String fileNameIN; // simply to store the file name

private FileReader fReader; // an object that can fetch data

private ExcelCSVParser csvReader; // an object that packages file data in csv format

private String rowItems[];

// file handling IN: Property Rent

private RENT PropertyRent[];

private String fileNameIN1; // simply to store the file name

private FileReader fReader1; // an object that can fetch data

private ExcelCSVParser csvReader1; // an object that packages file data in csv format

private String rowItems1[];

// file handling IN: Cards

private CARDS Cards[];

private String fileNameIN2; // simply to store the file name

private FileReader fReader2; // an object that can fetch data

private ExcelCSVParser csvReader2; // an object that packages file data in csv format

private String rowItems2[];

public MONOPOLY()

{

freeParking = 0; // setup procedure

activePlayer = 0; // which players turn it currently is

desiredProperty = 0; // the property they eithe rwant to buy/buy houses for/ buy hotel for

player = new int[2][5]; // 2D Array holding each players data

player[0][0] = 0;// position

player[1][0] = 0;// position

player[0][1] = 1500;// balance

player[1][1] = 1500;// balance

player[0][2] = 0;// train

player[1][2] = 0;// train

player[0][3] = 0;// jail

player[1][3] = 0;// jail

player[0][4] = 0;// utilities

player[1][4] = 0;// utilities

same = false; // for rolling a double

name1 = ""; // FOR PASSING INTO UI

name2 = "";// ^

street = 0;// ^

position = 0;// ^

die1 = 0;// ^

die2 = 0;// ^

diceTotal = 0;// ^

onProperty = false; // true if position is on a property

turnC = false; // if they roll double turn will remain active and enable roll dice button

owned = false; // true if current active player owns the property the active player is on

}

private void readData() throws IOException

{

PropertyCost = new PROPERTIES[40];

for(int i = 0; i < 40; i++)

{

PropertyCost[i] = new PROPERTIES();

}

// DECLARATION Property Cost

fileNameIN = "PropertyCost.csv"; // prepare file name

fReader = new FileReader(fileNameIN); // open the file

csvReader = new ExcelCSVParser(fReader); // pass management of IO to object that understands CSV

rowItems = new String[5]; // Set number of columns

PropertyRent = new RENT[40];

for (int i = 0; i < 40; i ++)

{

PropertyRent[i] = new RENT();

}

// DECLARATION Property Rent

fileNameIN1 = "PropertyRent.csv"; // prepare file name

fReader1 = new FileReader(fileNameIN1); // open the file

csvReader1 = new ExcelCSVParser(fReader1); // pass management of IO to object that understands CSV

rowItems1 = new String[7]; // Set number of columns

Cards = new CARDS[16];

for (int i = 0; i < 16; i ++)

{

Cards[i] = new CARDS();

}

// DECLARATION Property Rent

fileNameIN2 = "Cards.csv"; // prepare file name

fReader2 = new FileReader(fileNameIN2); // open the file

csvReader2 = new ExcelCSVParser(fReader2); // pass management of IO to object that understands CSV

rowItems2 = new String[2]; // Set number of columns

// read in Property Cost

rowItems = csvReader.getLine();

for(int i = 0; i < 40; i++)//Repeat for each order

{

rowItems = csvReader.getLine();

PropertyCost[i].readDetails(rowItems);//ead line in array

}//End Repeat

csvReader.close();//Close file

// read in Property Cost

rowItems1 = csvReader1.getLine();

for(int i = 0; i < 40; i++)//Repeat for each order

{

rowItems1 = csvReader1.getLine();

PropertyRent[i].readDetails1(rowItems1);//read line in array

}//End Repeat

csvReader.close();//Close file

// read in chance and community chest cards

rowItems2 = csvReader2.getLine();

for(int i = 0; i < 16; i++)//Repeat for each order

{

rowItems2 = csvReader2.getLine();

Cards[i].readDetails2(rowItems2);// read line in array

}// End Repeat

csvReader.close();// Close file

}

private void main()// used each time to refresh user interface

{

position1 = player[0][0]; // assign positions for images etc

position2 = player[1][0];//^

balance1 = player[0][1];//^

balance2 = player[1][1];//^

MyFrame frame = new MyFrame("Monopoly", activePlayer, die1, die2, name1, name2, position1, position2, balance1, balance2, onProperty, turnC, owned); // construct a MyFrame object

frame.setVisible( true ); // ask it to become visible

}

public void startGame() throws IOException

{

readData();// read in data from csv files

name1 = JOptionPane.showInputDialog(null, "Welcome to Monopoly! Please enter the first players name");// ask user for names

while(!(name1.matches("[a-zA-Z]+"))) // input validation

{

name1 = JOptionPane.showInputDialog(null, "ERROR! Please enter a valid name");

}

name2 = JOptionPane.showInputDialog(null, "Thanks. Now enter player 2's name.");//^

while(!(name2.matches("[a-zA-Z]+")))

{

name2 = JOptionPane.showInputDialog(null, "ERROR! Please enter a valid name");

}

turnC = true;// active turn

main();// display UI

}

private void runGame()

{

// display options

rollDice();// roll dice

checkPosition();// check where they have landed then display the appropiate buttons and positions and or message

if(same == true){// roll double function

same = false;

main();

}

}

private void rollDice()

{

diceTotal = 0;

die1 = 0;

die2 = 0;

die1 = (int)(6\*Math.random()+1);// roll 1d6

die2 = (int)(6\*Math.random()+1);// role 1d6

diceTotal = die1 + die2;

if (die1 == die2)// keep turn going and allow user to roll again

{

JOptionPane.showMessageDialog(null, "You rolled a double!");

same = true;

turnC = true;

} else // terminate turn

{

same = false;

turnC = false;

}

player[activePlayer][0] = player[activePlayer][0] + diceTotal;

if (player[activePlayer][0] > 39) // if user passes go

{

passGo();

}

}

private void passGo()

{

player[activePlayer][1] = player[activePlayer][1] + 200; // give £200

JOptionPane.showMessageDialog(null, "You passed go and have received £200!", "Passed Go", JOptionPane.INFORMATION\_MESSAGE);

player[activePlayer][0] = player[activePlayer][0] - 40; // reset position

}

private void endTurn()

{

// when end turn pressed

if (activePlayer == 0)

{

activePlayer = 1;

} else

{

activePlayer = 0;

}

turnC = true; // activate turn

main();

}

private void checkPosition()

{

position = player[activePlayer][0];

if (PropertyCost[position].getGroup() > 7)

{

onProperty = true;// to enable/disable the buy propery and buy houses/hotel button

if(PropertyCost[position].getOwner() == activePlayer)

{

owned = true;

}

main();

checkProperty();

}//check if on action tile

else if ((PropertyCost[position].getGroup() == 2) || (PropertyCost[position].getGroup() == 3))

{

onProperty = false;

takeCard();

// community chest and chance

}else if (PropertyCost[position].getGroup() == 6)

{

onProperty = true;

main();

scanUtilities();

// eg. electric company and water works

}else if (PropertyCost[position].getGroup() == 5)

{

onProperty = false;

//jail //4th index is jail presence

if (position == 10)

{

// just visiting

main();

}

else //GO TO jail

{

JOptionPane.showMessageDialog(null, "You have landed on the 'GO TO JAIL' tile so you will be moved to jail!","GO TO JAIL",JOptionPane.ERROR\_MESSAGE);

player[activePlayer][0] = 10;

jailProcedure();

}

//checkJail

}else if(PropertyCost[position].getGroup() == 4)

{

onProperty = false;

// super tax and income tax

if (position == 5)

{

JOptionPane.showMessageDialog(null, "You must pay £200 income tax!");

player[activePlayer][1] = player[activePlayer][1] - 200;

freeParking = freeParking + 200;

main();

checkBankrupt();

} else

{

JOptionPane.showMessageDialog(null, "You must pay £100 super tax!");

player[activePlayer][1] = player[activePlayer][1] - 100;

freeParking = freeParking + 100;

main();

checkBankrupt();

}

}else if (PropertyCost[position].getGroup() == 7)

{

onProperty = false;

//free parking

freeParking();

main();

}else if(PropertyCost[position].getGroup() == 1)

{

onProperty = false;

main();

}

main();

}

private void checkProperty()

{

desiredProperty = 0;

int currentHouses = 0;

int currentHotels = 0;

int position = 0;

int group = 0;

position = player[activePlayer][0];

if (PropertyCost[position].getGroup() == 17)//train station sequence

{

desiredProperty = position;

checkTrain();

}

else if(PropertyCost[position].getOwner() == (activePlayer)) // if owned by active player

{

street = 0;

street = PropertyCost[position].getGroup();

}

else if (PropertyCost[position].getOwner() == 2) // if unowned

{

if (PropertyCost[position].getPrice() > 1)

{

int a = 0;

desiredProperty = position;

a = JOptionPane.showConfirmDialog(null, "Do you want to buy this property?", "Option", JOptionPane.YES\_NO\_OPTION);

if (a == JOptionPane.YES\_OPTION)

{

buyProperty();// buy property assign to active player and deduct cost

owned = true;

}

onProperty = false;

}

}

else // if owned by other player

{

int receiver = 0;

if (activePlayer == 0)

{

receiver = 1;

}

onProperty = false;

int temp = 0;

currentHouses = PropertyCost[position].getHouses();

currentHotels = PropertyCost[position].getHotels();

// charge rent depending on the amount of houses or if a hotel is present

if (currentHotels == 0)

{

if (currentHouses == 1)

{

temp = PropertyRent[position].getOneHouse(); // rent cost

player[activePlayer][1] = player[activePlayer][1] - temp; // take away rent from balance

player[receiver][1] = player[receiver][1] + temp; // add rent to owner

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!"); // display that they have been charged for the property

checkBankrupt(); // check if they have enough money left

// same commentary for each one

}

else if (currentHouses == 2)

{

temp = PropertyRent[position].getTwoHouse();

player[activePlayer][1] = player[activePlayer][1] - temp;

player[receiver][1] = player[receiver][1] + temp;

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!");

checkBankrupt();

}

else if (currentHouses == 3)

{

temp = PropertyRent[position].getThreeHouse();

player[activePlayer][1] = player[activePlayer][1] - temp;

player[receiver][1] = player[receiver][1] + temp;

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!");

checkBankrupt();

}

else if (currentHouses == 4)

{

temp = PropertyRent[position].getFourHouse();

player[activePlayer][1] = player[activePlayer][1] - temp;

player[receiver][1] = player[receiver][1] + temp;

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!");

checkBankrupt();

}else

{

temp = PropertyRent[position].getRent();

player[activePlayer][1] = player[activePlayer][1] - temp;

player[receiver][1] = player[receiver][1] + temp;

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!");

checkBankrupt();

}

}else

{

temp = PropertyRent[position].getHotel();

player[activePlayer][1] = player[activePlayer][1] - temp;

player[receiver][1] = player[receiver][1] + temp;

JOptionPane.showMessageDialog(null, "You have been charged £" + temp + " rent for the property!");

checkBankrupt();

}

}

main();

}

private void buyProperty()

{

int activeGroup = 0;

activeGroup = PropertyCost[player[0][0]].getGroup();

if (player[activePlayer][1] > PropertyCost[desiredProperty].getPrice()) // if they can afford

{

player[activePlayer][1] = player[activePlayer][1] - PropertyCost[desiredProperty].getPrice();

buyTile();

} else // if they dont have enough

{

JOptionPane.showMessageDialog(null, "You can't afford this property!");

}

}

private void checkTrain()

{

if (PropertyCost[desiredProperty].getOwner() == 2) // if unowned

{

int a = 0;

a = JOptionPane.showConfirmDialog(null, "Would you like you to buy this train station?", "Option", JOptionPane.YES\_NO\_OPTION);

if (a == JOptionPane.YES\_OPTION)

{

if (player[activePlayer][1] > 199)

{

player[activePlayer][2] = player[activePlayer][2] + 1;// increment for how many stations they own

player[activePlayer][1] = player[activePlayer][1] - 200;

buyTile(); // change owner to the active player in the array

}else

{

JOptionPane.showMessageDialog(null, "You do not have enough money to buy this station!");

}

}

} else if (PropertyCost[desiredProperty].getOwner() == activePlayer) // if owned by the active player

{

// do nothing

} else // charge rent

{

int owner = 0;

if (activePlayer == 0)

{

owner = 1;

}

int owned = player[owner][2];

player[activePlayer][1] = player[activePlayer][1] - (owned \* 50);// charge rent to active player

player[owner][1] = player[owner][1] + (owned \* 50);// give rent to owner

JOptionPane.showMessageDialog(null, "You have been charged £" + (owned \* 50) + " rent for the station!");

checkBankrupt();

}

}

private void buyHouses()

{

PropertyCost[desiredProperty].addHouse();// increment one house in array

}

private void buyHotels()

{

PropertyCost[desiredProperty].addHotel();// increment one hotel in array

}

private void buyTile()

{

if (activePlayer == 0)

{

PropertyCost[desiredProperty].buyProperty1(); // change owner to player 1

}else

{

PropertyCost[desiredProperty].buyProperty2(); // change owner to player 2

}

}

private void checkStreet()

{

int a = 0;

desiredProperty = position;

street = PropertyCost[position].getGroup();

String[] name = new String[3];

int[] place = new int[3];

int counter = 0;

boolean active = true;

int i = 0;

int j = 0;

do

{

if (PropertyCost[i].getOwner() == activePlayer && street == PropertyCost[i].getGroup())

{

counter++;

place[j] = i;

name[j] = PropertyCost[i].getCostName();

j++;

}

if (counter == 2)

{

if (street == 8 || street == 16)

{

active = false;

}

} else if (counter == 3)

{

active = false;

} else if (i == 39)

{

active = false;

}

i++;

}

while(active);

String input = (String)JOptionPane.showInputDialog(null, "What property would you like to put the houses/hotel on?", "Property choice", JOptionPane.QUESTION\_MESSAGE, null, name, name[0]);

int tempe = 0;

if(input == name[0])

{

// do nothing

}else if(input == name[1])

{

tempe = 1;

}else

{

tempe = 2;

}

if (PropertyCost[place[tempe]].getHotels() == 1)// if maximum is on this property

{

JOptionPane.showConfirmDialog(null, "You already have one hotel, that is the maximum on this property!");

}

else // place house on property

{

desiredProperty = place[tempe];

if (counter == 2)// if two are owned

{

if (street == 8 || street == 16)// check if brown or dark blue streets

{

if (PropertyCost[position].getHouses() == 4)

{

buyHotels();

}else

{

buyHouses();

}

} else if (counter == 3) // if three owned add house

{

if (PropertyCost[position].getHouses() == 4)

{

buyHotels();

}else if (PropertyCost[position].getHotels() == 0)

{

buyHouses();

}

}

}else // if whole street isn't owned they can't buy a house

{

JOptionPane.showMessageDialog(null, "You don't own the entire street. So you cannot buy houses or hotels.");

}

}

main(); // refresh ui

}

private void takeCard()

{

int random = 0;

int position = player[activePlayer][0];

if(PropertyCost[position].getGroup() == 2)

{

random = (int)(9\*Math.random());

if (random == 0)

{

player[activePlayer][0] = 0;

JOptionPane.showMessageDialog(null,Cards[random].getCardName());

player[activePlayer][1] = player[activePlayer][1] + 200; // give £200

JOptionPane.showMessageDialog(null, "You passed go and have received £200!", "Passed Go", JOptionPane.INFORMATION\_MESSAGE);

} else

{

JOptionPane.showMessageDialog(null,Cards[random].getCardName(), "Chance" ,JOptionPane.INFORMATION\_MESSAGE);

player[activePlayer][1] = player[activePlayer][1] + Cards[random].getCost();

freeParking = freeParking + Cards[random].getCost();

}

}else

{

random = (int)(Math.random()\*8)+8;

if (random == 8)

{

player[activePlayer][0] = 0;

JOptionPane.showMessageDialog(null,Cards[random].getCardName());

player[activePlayer][1] = player[activePlayer][1] + 200; // give £200

JOptionPane.showMessageDialog(null, "You passed go and have received £200!", "Passed Go", JOptionPane.INFORMATION\_MESSAGE);

} else

{

JOptionPane.showMessageDialog(null,Cards[random].getCardName(), "Community Chest" ,JOptionPane.INFORMATION\_MESSAGE);

player[activePlayer][1] = player[activePlayer][1] + Cards[random].getCost();

freeParking = freeParking + Cards[random].getCost();

}

}

}

private void scanUtilities()

{

// check if both are owned

// x10 if bth owned and x4 owned

int position = player[activePlayer][0];

if (PropertyCost[position].getOwner() == 2)

{

int a = 0;

desiredProperty = position;

a = JOptionPane.showConfirmDialog(null, "Would you like you to buy this utility?", "Option", JOptionPane.YES\_NO\_OPTION);

if (a == JOptionPane.YES\_OPTION)

{

if (player[activePlayer][1] > 149)

{

player[activePlayer][4] = player[activePlayer][4] + 1;

player[activePlayer][1] = player[activePlayer][1] - 150;

buyTile();

}

}

} else if (PropertyCost[position].getOwner() == activePlayer)

{

// do nothing

} else

{

int other = 0;

if(activePlayer == 0)

{

other = 1;

}

int owned = player[other][4];

if (owned == 2)

{

owned = 10; //setting multipler to 10

}

else

{

owned = 4; //setting multipler to 4

}

JOptionPane.showMessageDialog(null, "You have been charged £" + (owned \* diceTotal) + " rent for the utility!");

player[activePlayer][1] = player[activePlayer][1] - (owned \* diceTotal);

player[other][1] = player[other][1] + (owned \* diceTotal);

checkBankrupt();

main();

}

}

private void jailProcedure()

{

int match = 0;

if (player[activePlayer][1] < 50)

{

JOptionPane.showMessageDialog(null, "You cannot afford the fine for jail! You must attempt to roll 3 doubles!");

boolean roll = true;

while (roll)

{

die1 = (int)(6\*Math.random()+1);

die2 = (int)(6\*Math.random()+1);

JOptionPane.showMessageDialog(null, "You have rolled " + die1 + " and " + die2 + ".");

if (die1 == die2)

{

JOptionPane.showMessageDialog(null, "That was a double!");

match++;

if (match == 3)

{

JOptionPane.showMessageDialog(null, "You have rolled 3 doubles and escaped jail!");

player[activePlayer][4] = 0;

}

}else

{

JOptionPane.showMessageDialog(null, "These are not doubles your turn is over.");

roll = false;

}

}

main();

}else

{

JOptionPane.showMessageDialog(null, "You must pay a £50 bail.");

player[activePlayer][0] = 10;

player[activePlayer][1] = player[activePlayer][1] - 50;

player[activePlayer][4] = 0;

main();

}

}

private void freeParking()

{

JOptionPane.showMessageDialog(null, "You have landed on free parking, you will gain £" + freeParking + "!");

player[activePlayer][1] = player[activePlayer][1] + freeParking;

freeParking = 0;

}

private void checkBankrupt()

{

if (player[activePlayer][1] < 0)

{

ImageIcon trophy = new ImageIcon("Z:\\Documents\\Computing\\Project\\trophy.png");

if (activePlayer == 0)

{

JOptionPane.showMessageDialog(null,name1 + " has lost the game!", "Unlucky!", JOptionPane.ERROR\_MESSAGE);

JOptionPane.showMessageDialog(null,name2 + " has won the game!", "Congratulations!", JOptionPane.INFORMATION\_MESSAGE, trophy);

}else

{

JOptionPane.showMessageDialog(null,name2 + " has lost the game!", "Unlucky!", JOptionPane.ERROR\_MESSAGE);

JOptionPane.showMessageDialog(null,name1 + " has won the game!", "Congratulations!", JOptionPane.INFORMATION\_MESSAGE, trophy);

}

}

}

class MyFrame extends JFrame{

JPanel panel;

JLabel label;

//GUI

JButton buyHouse;

JButton endTurn;

JButton rollDice;

JButton buyProperty;

public JLabel turnTitle;

JLabel label1;

JLabel label3;

JLabel label4;

JLabel rollText;

JLabel label5;

// extending the jframe class

MyFrame(String title, int activePlayer, int die1, int die2, String name1, String name2, int position1, int position2 ,int balance1, int balance2, boolean onProperty, boolean turnC, boolean owned) {

super(title);

// setSize( 150, 100 );

setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

setLayout( new FlowLayout() ); // set the layout manager

// label = new JLabel("Hello Swing!"); // construct a JLabel

// add( label ); // add the label to the JFrame

buyHouse = new JButton();

endTurn = new JButton();

rollDice = new JButton();

buyProperty = new JButton();

turnTitle = new JLabel();

label1 = new JLabel();

label3 = new JLabel();

label4 = new JLabel();

rollText = new JLabel();

label5 = new JLabel();

buyProperty.setEnabled(false);

buyProperty.addActionListener(taskPerformer);

endTurn.addActionListener(taskEndTurn);

rollDice.addActionListener(taskRoll);

buyHouse.addActionListener(taskHouse);

buyHouse.setEnabled(false);

if (owned == true)

{

buyHouse.setEnabled(true);

}

if (onProperty == false)

{

buyProperty.setEnabled(false);

}else

{

buyProperty.setEnabled(false);

}

if (turnC == false)

{

rollDice.setEnabled(false);

}

// ======== this ========

setBackground(new Color(191, 219, 174));

Container contentPane = getContentPane();

// ---- button1 ----

buyHouse.setText("Buy House/Hotel");

// ---- button2 ----

endTurn.setText("End Turn");

// ---- button3 ----

rollDice.setText("Roll Dice");

// ---- button4 ----

buyProperty.setText("Buy Property");

// ---- turnTitle ----

if (activePlayer == 0)

{

turnTitle.setText("It is player " + name1 + "'s turn! Balance: £" + balance1);

}else

{

turnTitle.setText("It is player " + name2 + "'s turn! Balance: £" + balance2);

}

turnTitle.setHorizontalAlignment(SwingConstants.CENTER);

turnTitle.setFont(turnTitle.getFont().deriveFont(turnTitle.getFont().getSize() + 10f));

// ---- label1 ----

label1.setIcon(new ImageIcon("Z:\\Documents\\Computing\\Project\\Monopoly Pictures\\" + position1 + ".png"));

// ---- label3 ----

label3.setIcon(new ImageIcon("Z:\\Documents\\Computing\\Project\\Dice\\" + die1 + ".png"));

// ---- label4 ----

label4.setIcon(new ImageIcon("Z:\\Documents\\Computing\\Project\\Dice\\" + die2 + ".png"));

// ---- textField2 ----

rollText.setText("You rolled a " + (die1+die2) + ".");

rollText.setHorizontalAlignment(SwingConstants.CENTER);

// ---- label5 ----

label5.setIcon(new ImageIcon("Z:\\Documents\\Computing\\Project\\Monopoly Pictures\\" + position2 +".png"));

GroupLayout contentPaneLayout = new GroupLayout(contentPane);

contentPane.setLayout(contentPaneLayout);

contentPaneLayout.setHorizontalGroup(

contentPaneLayout.createParallelGroup()

.addGroup(contentPaneLayout.createSequentialGroup()

.addGap(50, 50, 50)

.addGroup(contentPaneLayout.createParallelGroup()

.addGroup(contentPaneLayout.createSequentialGroup()

.addGap(10, 10, 10)

.addComponent(turnTitle, GroupLayout.PREFERRED\_SIZE, 536, GroupLayout.PREFERRED\_SIZE))

.addGroup(contentPaneLayout.createSequentialGroup()

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)

.addComponent(label1)

.addComponent(buyProperty, GroupLayout.PREFERRED\_SIZE, 150, GroupLayout.PREFERRED\_SIZE)

.addComponent(buyHouse, GroupLayout.PREFERRED\_SIZE, 150, GroupLayout.PREFERRED\_SIZE))

.addGroup(contentPaneLayout.createParallelGroup()

.addGroup(contentPaneLayout.createSequentialGroup()

.addGap(29, 29, 29)

.addComponent(label3)

.addGap(51, 51, 51)

.addComponent(label4))

.addGroup(contentPaneLayout.createSequentialGroup()

.addGap(68, 68, 68)

.addComponent(rollText, GroupLayout.PREFERRED\_SIZE, 100, GroupLayout.PREFERRED\_SIZE)))

.addGap(34, 34, 34)

.addGroup(contentPaneLayout.createParallelGroup()

.addComponent(rollDice, GroupLayout.Alignment.TRAILING, GroupLayout.PREFERRED\_SIZE, 150, GroupLayout.PREFERRED\_SIZE)

.addComponent(endTurn, GroupLayout.Alignment.TRAILING, GroupLayout.PREFERRED\_SIZE, 150, GroupLayout.PREFERRED\_SIZE)

.addComponent(label5))))

.addContainerGap(38, Short.MAX\_VALUE))

);

contentPaneLayout.setVerticalGroup(

contentPaneLayout.createParallelGroup()

.addGroup(GroupLayout.Alignment.TRAILING, contentPaneLayout.createSequentialGroup()

.addContainerGap()

.addComponent(turnTitle, GroupLayout.PREFERRED\_SIZE, 40, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 71, Short.MAX\_VALUE)

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.LEADING, false)

.addGroup(GroupLayout.Alignment.TRAILING, contentPaneLayout.createSequentialGroup()

.addComponent(rollText, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)

.addComponent(label3)

.addComponent(label4))

.addGap(68, 68, 68))

.addGroup(GroupLayout.Alignment.TRAILING, contentPaneLayout.createSequentialGroup()

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)

.addComponent(label1)

.addComponent(label5))

.addGap(18, 18, 18)))

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(buyHouse)

.addComponent(rollDice))

.addGap(41, 41, 41)

.addGroup(contentPaneLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(buyProperty)

.addComponent(endTurn))

.addContainerGap())

);

pack();

setLocationRelativeTo(getOwner());

// JFormDesigner - End of component initialization //GEN-END:initComponents

}

public ActionListener taskPerformer = (new ActionListener()

{

public void actionPerformed(ActionEvent e) {

runBuyProperty();

}

});

public void runBuyProperty()

{

checkProperty();

}

public ActionListener taskEndTurn = (new ActionListener()

{

public void actionPerformed(ActionEvent e) {

runEndTurn();

setVisible(false); //you can't see me!

dispose();

}

});

public void runEndTurn()

{

setVisible(false);

endTurn();

}

public ActionListener taskRoll = (new ActionListener()

{

public void actionPerformed(ActionEvent e) {

runRollDice();

setVisible(false); //you can't see me!

dispose();

}

});

public void runRollDice()

{

setVisible(false);

runGame();

}

public ActionListener taskHouse = (new ActionListener()

{

public void actionPerformed(ActionEvent e) {

runBuyHouse();

}

});

public void runBuyHouse()

{

checkStreet();

}

}

}