**Project 1**

**<LeepFrog Mathematics Game>**

**CSC-5/46024**

**Name: Megan Varner**

**Date: 07/18/14**

**Introduction**

Title: LeepFrog Mathmatics

My game is LeepFrog Mathematics that I grew up playing on a tablet and absolutely loved. The game comes in an arcade like structure and the first game within the game is called “guess the number I’m thinking of,” which is a game of chance. It is used on long car rides and to entertain young children. Someone [known as the chief in the game] will think of a number and not tell anyone. The player will then try and guess the number the chief is thinking of. If done fair, the chief will usually tell a bystander what the number is, to make sure he or she does not change his/her mind mid-game. But as the code is written the computer will not change its guess therefore it is still following the rules. The point of the game is to try and get the least amount of guesses and beat your personal best record. On your own you make a score card and record the amount of guesses it took you each time and then you can just keep playing and recording until you’re tired. You can return to the same score card days later and play again and record the number of guesses each time till you guessed the correct number as well. This game is meant to pass time, there is no strategy and you try and beat your own best record of least amount of guesses till you got the correct number. The next game within LeepFrog Mathematics is an addition game. You try and do the random addition equation and when I played, we use to get treats if we won from our parents. Like the addition game, the subtraction game, is the same concept only with a random subtraction equation and again when we won our parents gave us treats.

**Summary**

Project size: 200+ lines

The number of variables: about 30

The number of method: 17

My goal was to create an all-age program, that anyone could play and I could use to entertain my young nieces. Therefore, I used the new concepts that we have learned including loops, if/else statements, menus, ect to create a program that is number oriented that they could play and get comfortable with different types of number positive, negative, large, small, ect . The program itself is a work in progress because ultimately I want to make the program with five or six multiple games within it to choose to play. As of right now the ‘arcade’ consist of three games and that’s the guess the number I’m thinking of, add with me, and subtract with me.

It took about two weeks to develop the outline and the sub-games.

I met a few problems in the process.

I referenced our text book, past homework assignments, and the professor’s online examples to solve them.

**Description**

The main point of this program is how it randomly chooses numbers and can be used as a learning tool.

**Pseudo Code**

*Initialize*

*Prompt to press 1, 2, or 3 in game menu or anything else to exit menu*

*If number 1 is pressed*

*Prompt to guess the random number*

*If number is random number congratulate player*

*Else if number is larger display it’s larger and prompt to guess again*

*Else if number is smaller display it’s smaller and prompt to guess again*

*Play again press P/p or else bool is false and exit sub- game to menu*

*If number 2 is pressed*

*Generate random numbers to be added*

*Prompt to put in answer for random addition equation*

*Output correct answer*

*If answer is correct congratulate player*

*Else answer is wrong and prompt player to play again*

*Play again press P/p or else bool is false and exit sub-game to menu*

*If number 3 is pressed*

*Generate random numbers to be subtracted*

*Prompt to put in answer for random subtraction equation*

*Output correct answer*

*If answer is correct congratulate player*

*Else answer is wrong and prompt player to play again*

*Play again press P/p or else bool is false and exit sub-game to menu*

*If anything else pressed*

*Exit menu*

**FlowChart**

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| Integer | PlayerEggNum[] [] | The number of eggs in holes | Class gameBoardpublic  -void actionPerformed(ActionEvent e) |
|  | Socre[] | Each player’s score array | GetScore() |
|  | ChickenNum[] | The number of chickens in  Mancala | GetScore() |
|  | RemaindEggs[] | The number of eggs remaining in holes after finised | isFinished() |
|  | thisTurn | Have the player number Whose turn now is | Class gameBoardpublic  -void actionPerformed(ActionEvent e)  changeTurn() |
|  | number | The number of holes | MoveEgg(), opponent(), actionPerformed(ActionEvent e) |
|  | eggNum | The number eggs when distributing | MoveEgg(), actionPerformed(ActionEvent e) |
|  | player | Player zone eggs are distributed | MoveEgg(), actionPerformed(ActionEvent e) |
|  | getScore | The score | actionPerformed(ActionEvent e) |
| JButton | playerBt[][] | The JButton array that contains each hole button | actionPerformed(ActionEvent e) |
|  | start | The start button | Class Mancala – ActionPerformed(ActionEvent e) |
|  | giveUp | The give up button | Class Mancala –  actionPerformed(ActionEvent e) |
| JPanel | TitlePanel | The panel that contains titleLabel | Class GameBoard – GameBoard() |
|  | gamePanel | The panel that contains player buttons, labels… | Class GameBoard – GameBoard() |
|  | scorePanel | The panel that contains scores | Class GameBoard – GameBoard() |
|  | buttonPanel | The panel that contains start and giveup buttons | Class Mancala – init() |
| JLabel | PlayerLb[][] | The JLabel for number of eggs in each hole | actionPerformed(ActionEvent e) |
|  | Chicken[] | The Jlabel for number of chickens of each player | GetScore() |
| JTextField | ScoreText[] | The textField for each player’s score | GetScore() |
| Color | Color[] | The array of button colors | SetColors() |

|  |  |  |  |
| --- | --- | --- | --- |
| Icon | Egg[] | The Icon array for button images | SetImageFiles() |
|  | No[] | The Icon array for number of eggs | SetImageFiles() |
| Timer | timer | The object of Timer class | MoveEgg() |
| Boolean | flag | Check the button is the first button clicked | actionPerformed(ActionEvent e) |
| GameBoa rd | mainPanel | The object of GameBoard class | Class Mancala – init() |
| BorderLa yout | border | The layout for GameBoard | Class GameBoard – GameBoard() |
| GirdBagL  ayout | gbLayout | The layout for gamePanel | Class GameBoard – GameBoard() |

**Java Constructs**

|  |  |  |
| --- | --- | --- |
| Chapter | New syntax and Keywords | Location |
| 2 | public class (class definition) | Public class Mancala  Public class ButtonHandler  Public class GameBoard |
| JoptionPane.showMessageDialog  JOptionPane.INFORMATION\_MESSAGE | JOptionPane.showMessageDialog(null, scroll, “How to play, JOptionPane.INFORMATION\_MESSAGE); |
| Import statement  Javax.swing package | Iimport javax.swing.\*; |
| If structure | if (e.getSource() == playerBt[i][j]) {} |
| Equality operators and relational operators  (==, !=, >, <, >=, <=) | for (int i=0; i<2; i++) |
| Arithmetic operators (+, -, \*, /) | score[0] += (remainedEggs[0]\*25); |
| Int primitive type | Int score[] |
| String | String output |
| Void keyword | Void init() |
| 3 | Extends JApplet | Public class Project extends JApplet |
| Appletviewer, HTML tag | Mancala.html |
| Init() | Public void init() |
| Java.awt package | Import java.awt.\*; |
| Java.awt.event package | Import java.awt.event.\*; |
| 4 | If/else selection structure | If (e.getSource() == start) {} Else {} |
| Assignment operator (+=) | score[thisTurn] += 25; |
| Increment operator (++) | chickenNum[thisTurn]++; |
| While repetition structure | while ((text = r.readLine()) != null) |
| 5 | For repetition structure | For (int i=0; i<2; i++) {} |
| JTextArea | output = new JTextArea(15, 35); |

|  |  |  |
| --- | --- | --- |
|  | Break | if (e.getSource() == playerBt[i][j]) {  player = i; number = j; break; |
| Logical operators (true, false) | Flag = true; |
| 6 | Container c = getContentPane(); | Container c = getContentPane(); |
| Add | c.add(mainPanel, BorderLayout.CENTER); |
| setText | whoseTurn.setText("Player 1's Turn!"); |
| setEditable | scoreText[i].setEditable(false); |
| Return | Return true; |
| ActionListener | Public class ButtonHandler implements  ActionListener |
| actionPerformed(ActionEvent e) | Public void actionPerformed(ActionEvent e) |
| JButton | PlayerBt[][] = new Jbutton[I][j];  giveUp = new JButton(“Give up”); |
| JLabel | PlayerLb[][] = new Jlabel[I][j]; PlayerLabel1 = new JLabel(“Player 1”); |
| JTextField | scoreText[] = new JTextField[2]; |
| setLayout | gamePanel.setLayout(gbLayout); |
| addActionListener(this) | playerBt[i][j].addActionListener(handler); |
| Method | Public Boolean isFinished() Public void setImageFiles() |
| 7 | Array | PlayerBt[][] = new JButton[2][6]; Score[] = new int[2]; |
| Font.BOLD, Font.ITALIC, Font.PLAIN | Font f = new Font("Arial", Font.PLAIN, 14); |
| setFont | output.setFont(f); |
| 8 | Instance of a class | mainPanel = new GameBoard(); |
| Dot(.) operator | MainPanel.initialize(); |
| private | Private Container c; |
| 9 | Show method | applicationWindow.show(); |
| WindowAdapter | new WindowAdapter() |
| WindowClosing method | public void windowClosing(WindowEvent e) |
| WindowEvent class | public void windowClosing(WindowEvent e) |
| WindowListener interface | applicationWindow.addWindowListener |
| 10 | Append method | buf.append(text + "\n"); |
| StringBuffer | StringBuffer buf = new StringBuffer(); |
| 11 | setBackground | start.setBackground(color1); |
| Color | Color color[] = new color[15]; |
| 12 | BorderLayout | private BorderLayout border; |
| Jpanel class | private JPanel buttonPanel |
| Icon | private Icon egg[] = new Icon[16]; |
| getSource | if (e.getSource() == start) |
| ImageIcon | egg[i] = new ImageIcon("images/egg" + i + ".gif"); |
| setEnabled | start.setEnabled(false); |

|  |  |  |
| --- | --- | --- |
|  | SetIcon | title.setIcon(titleIcon); |
| 13 | GridBagLayout | private GridBagLayout gbLayout; |
| GridBagConstraints.NONE | gbConstraints.fill = GridBagConstraints.NONE; |
| Gridheight, gridwidth, gridx, gridy | gbConstraints.gridx = column; gbConstraints.gridy = row; gbConstraints.gridwidth = width; gbConstraints.gridheight = height; |
| JMenu class | JMenuBar bar = new JMenuBar(); |
| JMenuBar class | JMenuBar bar = new JMenuBar(); |
| JMenuItem class | JMenuItem newItem = new  JMenuItem("New"); |
| setConstraints | gbLayout.setConstraints(com, gbConstraints); |
| setJMenuBar | setJMenuBar(bar); |
| 14 | Catch (Exception e) | catch (IOException e2) |
| IOException | catch (IOException e2) |
| Try block | try {RandomAccessFile r = new  RandomAccessFile(name, "r");  …. } |
| 16 | File class | File name = new File("help.txt"); |
| .gif file | egg[i] = new ImageIcon("images/egg" + i + ".gif"); |
| Start method | timer.start(); |
| Stop method | Timer.stop(); |
| Timer class | Timer timer = new Timer(1000, this); |
| 17 | RandomAccessFile | RandomAccessFile r = new  RandomAccessFile(name, "r"); |
| readLine method | while ((text = r.readLine()) != null) |

**Reference**

1. textbook

2. https://github.com/mkvarner/mv2504740/tree/master/Hmwk

3. https://github.com/Riverside-City-College-Computer-

Science/Summer14\_CSC5\_46024/tree/master/ml1150258/Examples

**Program**

CODE