**Dating simulator – User’s and Developer’s Guide**

Stephen Wen, Alvin Janes, MJ Chen

**User’s Guide**

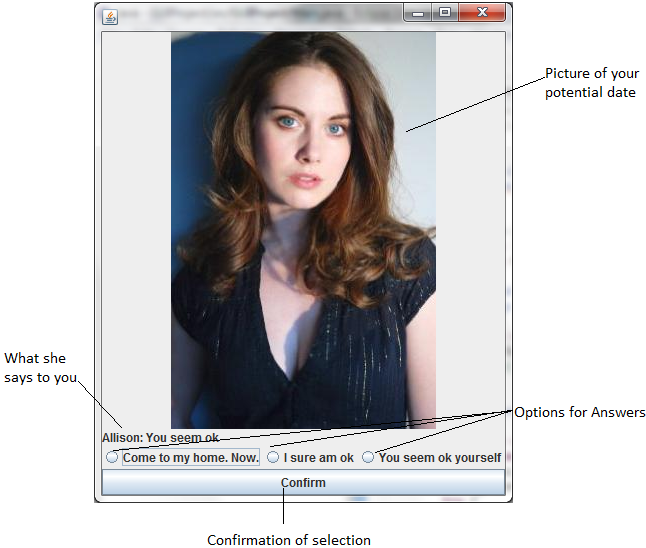
Background

A dating simulation is a fun way to pretend to talk to girls. This game intends to give you a simple idea of what one can say to a girl and how she may react. The objective is to get as many phone numbers from girls as you can. Pick the answers that please the girl the most and she may give you her number. If she is unsatisfied, she will answer differently.

Getting Started

This game has been coded with Eclipse. The program will run on Eclipse, this program can be downloaded online for free. In the .zip file, several files exist, the driver class is the one named Main and this is the class that contains the main method which runs the program. Compiling and running the Main class will start the game. The game is controlled completely with mouse clicks. An answer option must be selected by clicking the circle next to the option you wish. After that, the user must confirm their option by clicking the button labeled “Confirm” at the bottom of the window. Selecting answers that please the girl you are talking to will lead to continued conversation or receiving her number.

Features of the Program



Limitations and Bugs of the Program

This program currently has a very limited number of women with a very limited question pool. Currently there are only 3 women in the game, and each only has 3 point levels of questions, each with two questions. Women do not really have personalities. The problems listed so far could be solved by adding to the input text file, as well as generalizing the parsing of input. The employment of a writer who could write effective storylines could also be helpful in this regard. There is also no way of choosing which woman or type of woman you wish to talk to. This was intended to be a part of the game, in the Map class, though due to a lack of group cooperation, this part of the project was incomplete and omitted from the final product.

**Developer’s Guide**

Overview

There are 7 classes that make up this program and 2 classes that are not used. There is the Main class, this is a driver class for the rest of the program. There is the Location class, this contains Person objects as its main purpose. There is the Person class, this describes a person to talk to, it contains the questions the person will ask as well as an image to show the person. There is the question object, it contains the possible answer objects as well as the text that will be said by the person. There is the Answer object, this contains a text for what the answer option is as well as a point value for selecting this option. Finally, there is the EndDialog class, this class is the popup that appears at the end of an encounter with a woman.

Sections of each Feature of the Program

**Main Class**

The Main class of this program contains a JFrame object on which all of the game will be played. It also contains the scanner to take in file input and parses the file initially. This is a driver class and as such creates a location object to pick a person to be added to the JFrame, but what it does stops there. It is broken into methods that were originally intended to test aspects of the program, but the testLocation method was adapted to be the complete program.

**Location Class**

The location class is a container for person objects. The main constructor for this object takes in the input from the scanner in Main and parses it further to be used in each Person object. It contains a Person array that holds all the Person objects at this location. It also has the pickPerson method to randomly select one of the Person objects to be used for the game. There are some methods and data fields that were included to be used in the Map class which was not completed. These include the coordinates and icon data fields, and the getLoc method. The coordinates and getLoc were meant to be used to have a place on the map. The icon was to be drawn on the map.

**Person Class**

The person class is meant to represent any girl that you can choose to interact with in the game. It contains a two dimensional array of questions. The first index of this array represents a point value of the question. This point value is used to determine the happiness of the questions and furthermore the ending. The second index of this array is used to have some variability in the speech of the character. It contains an add method to add that takes in a question object to add each of the components of a question to the JPanel that represents the person. Current point value is stored as well as a maximum and a minimum value. These, along with the maximum number of questions, are used to determine which questions to ask and when to end the interaction. The Image object Face is used to display a face that can be associated with the person. A JLabel and extra JPanel are used to show the picture. The constructor for this takes in the string as was parsed for use in the Location class. It also takes in a reference to the location that this person belongs to, this allows backtracking up the ‘has-a’ relationships. The String is used to determine the values on which questions to choose as well as when to stop the relationship. The String also contains the strings needed to create the Question objects. There is an end method that is called when the date ends, this will create a EndDialog object which shows the ending dialog box. The reset method exists for after the person has ended the date, so this person can be used again if such is the case. The actionPerformed method is used when an ActionEvent is fired by the confirm button from a question. This updates the number of points, and selects a new question to be displayed. It also checks if the interaction should end and ends the interaction appropriately if this is the case. The loadImage method is used to define all images used in this class.

**Question Class**

The Question class contains three Answer objects, a confirm JButton, and a String to represent the words of the woman. The constructor for Question takes in a string as it was parsed by Person class, it further separates the String to be used to define the Answer objects. It also takes in a person object so we can trace back up the stack. It contains several getter methods so the objects can be accessed. It also has a method to add an ActionListener to the confirm button.

**Answer Class**

The Answer class contains a String and an int value. The String is to represent the possible response and the int value represents the number of points awarded to the player for selecting this answer. There is a getter method for the points so they can be used in other classes.

**Map Class**

This class was not completed and is not used in the current version of the program, but it still exists in the code. This was intended to be used to give a player a chance to select which location they wish to go to. It implements keyListener because it was intended to use keys to move the player around the map. The getLoc method was coded to select the closest location on the map.

**Player Class**

This class was not completed and not included in the version of the program either. The purpose of this class was to store the score of the user and create high scores as well as move the player object around the map. The moveUp, moveDown, moveLeft, and moveRight methods were coded to move the player around the map by translating the position Point of the Player object.

Suggestions for Improvement

A few major areas of improvement for this game include increasing the speech options for each person and completing the Player and Map classes. Increasing the amount of options each person has would make the game much more interesting. Differentiating the people as well, giving each a different personality would also be a good thing to do. It would add flavor to the game. This would interact well with the Map class. With the Map class you could select a Location to meet women who have a personality based on where you found them. Having women that interacted with you and held much longer questions would be much more interesting. Completing the Map class would allow players to have much more choice in which women they talk to. The Map class working completely would require the Player class to work to some extent. High scores would be another way to improve this program. Counting the number of phone numbers earned by the player would be a way to determine the prowess of the player. Writing a file to store high scores would allow for scores to be kept and returned later.