Samuel Brownley 31691379

ICT373 – Software Architectures

Assignment 1 - Question 2

Contents

[Title 2](#_Toc479952958)

[Requirements/Specification 2](#_Toc479952959)

[User Guide 2](#_Toc479952960)

[Structure/Design 3](#_Toc479952961)

[UML 3](#_Toc479952962)

[Class Description 4](#_Toc479952963)

[Limitations 5](#_Toc479952964)

[Testing](#_Toc479952965) 6

[Listings 11](#_Toc479952966)

# Title

Title: Dating Extreme

Author: Samuel Brownley - 31691379

Date: 14/04/2017

Files:

* Advertiser.java
* Assignment1.java
* Customer.java
* Customers.java
* PartnerSought.java
* ReplyMessage.java
* Repsonder.java

Purpose: An application that allows user to find a relation match based on certain criteria given at sign up.

# Requirements/Specification

The program allows users to enter their details as either an advertiser or responder and find matches based on those details.

Assumptions:

* The password for each user is the users name follow by a 1. Eg. If the users name is Bill, their password is Bill1

# User Guide

**How to use:**

Upon start up, users will be given a list of options to select from. These options will be number and the user simply has to enter a number from the list to perform the associated action. This will continue throughout the program for all option menus.

When the user is required to enter addition information (age, messages), they will be prompted to do so. If the input is considered invalid, they will be asked to reenter their input until it is valid.

To use any of the addition features, the user must create a profile by selecting 2 from the starting menu. Once they have entered their details, a profile is created and they will be able to login. As a responder, users will be able to view their matches (advertisers whose criteria matchs the responder) and send messages to those they would like to talk to. As an advertiser, you will be able to view the messages received from the responders.

In addition to these account specific functions, users will be able to view each of the responders and advertisers and their details.

# Structure/Design

## UML



## Class Description

**Advertiser.java**

Defines the Advertiser class. Advertiser is inherited from Customer, so it contains the name, password, age, income and gender in addition to an advert, partner details and messages.

**Assignment1.java**

Contains a Customers object that is manipulated through various input from the user.

**Customer.java**

Defines the Customer class. Customer contains a name, password, gender, age, income. These are the objects that the users of the program are manipulating when creating, deleting, and viewing the users of this software.

**Customers.java**

Defines the Customers class. Customers contain This class is used to store and manipulate a list of customers. These customers can be added to, removed and have elements viewed (messages and matches).

**PartnerSought.java**

Defines the PartnerSought class. PartnerSought contains the details that an Advertiser is able to search by. These details are gender, minimum age, maximum age, minimum income, maximum income.

**ReplyMessage.java**

Defines the ReplyMessage class. Contains the owner of the message (sender) and the message text. These are sent to Advertisers.

**Repsonder.java**

Defines the Responder class. Responders is inherited from Customer, so it contains the name, password, age, income and gender.

# Limitations

Finding matches: the way matches are found for each of the responders is different than how it has been laid out in the assignment sheet. Instead of showing the matches as soon as the user logs in, I have given an option in the menu once a responder has logged in to view their matchs. This also applies the messages of the responder. They are given an option to view their messages from the menu instead of lsiting them straight after log in.

Deleting a user: The method I have used to delete user is to find them by name. If there are multiple users of the same name, the first found in the ArrayList of customers is removed, even if it is not the intended customer to delete.

# Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Outcome** | **Actual Outcome** | **Proof** |
| Login – Bill   |  |  | | --- | --- | | **Input** | **Value** | | Name | Bill | | Password | Bill1 | | Bill is logged in and confirmation message is printed. | Bill is logged in and confirmation message is printed. |  |
| Create account – Advertiser   |  |  | | --- | --- | | **Input** | **Value** | | Name | Jerry | | Gender | m | | Age | 35 | | Income | 26500 | | Advert | I like stuff | | **Sought After Details** |  | | Gender | F | | Min age | 28 | | Max age | 40 | | Min income | 15000 | | Max income | 30000 | | Jerry is created as an advertiser | Jerry is created as an advertiser | New list of advertisers |
| Create account – Responder   |  |  | | --- | --- | | **Input** | **Value** | | Name | Jerry | | Gender | M | | Age | 35 | | Income | 26500 | | Jerry is created as an responder | Jerry is created as an responder | New list of responders |
| Delete account – name that exist – Bill (responder) | Deletes the user and prints confimation message | Deletes the user and prints confimation message | Remaining responders |
| Delete account – name that does not exist | Prints customer does not exist | Prints customer does not exist |  |
| List all advertisers | Prints the name, gender, age, income and advert details of each advertiser | Prints the name, gender, age, income and advert details of each advertiser |  |
| List all responders | Prints the name, gender, age, income and advert details of each responder | Prints the name, gender, age, income and advert details of each responder |  |
| Exit | Program exits | Program exits |  |
| Logout | User is logged out | User is logged out |  |
| Repsonder - check matches | List of matches are shown and input message is shown to message certain person | List of matches are shown and input message is shown to message certain person |  |
| Responder - Send a message to a match | List of matches are shown and input message is shown to message certain person | List of matches are shown and input message is shown to message certain person | Viewing message as Laura |
| Advertiser – Read message | Shows the message that is selected | Shows the message that is selected |  |
| Advertiser – Delete message | Deletes the chosen message and prints the remaining messages | Deletes the chosen message and prints the remaining messages |  |

# Listings

**Advertiser.java**

package assignment1;

import java.util.ArrayList;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class Advertiser extends Customer {

/\*\*

\* The advertisement the users wishes to be seen by other users

\*/

private String advert;

/\*\*

\* The details of the partners being looked for

\*/

private PartnerSought partnerSought;

/\*\*

\* A list of messages this user has received

\*/

private ArrayList<ReplyMessage> replyMessages;

/\*\*

\* Creates an advertiser

\*

\* Sets the values to the parameters and initialises the message array

\*

\* @param name username of the new advertiser

\* @param newG gender of the new advertiser

\* @param newAge age of the new advertiser

\* @param newIncome income of the new advertiser

\* @param newAd advert of the new advertiser

\* @param newDetails details of wanted partners

\*/

Advertiser(String name,char newG,int newAge,int newIncome,String newAd,PartnerSought newDetails)

{

super(name,newG,newAge,newIncome);

SetAdvert(newAd);

SetPart(newDetails);

replyMessages = new ArrayList<>();

}

/\*\*

\* Set the advert to newAd

\*

\* @param newAd

\*/

void SetAdvert(String newAd)

{

advert=newAd;

}

/\*\*

\* Gets the advert

\*

\* @return advert

\*/

String GetAdvert()

{

return advert;

}

/\*\*

\* Sets partnerSought to newPart

\*

\* @param newPart

\*/

void SetPart(PartnerSought newPart)

{

partnerSought = newPart;

}

/\*\*

\* Returns the partner details

\*

\* @return

\*/

PartnerSought GetPart()

{

return partnerSought;

}

/\*\*

\* Returns the list of messages

\*

\* @return

\*/

ArrayList<ReplyMessage> GetMsgs()

{

return replyMessages;

}

/\*\*

\* Add newMsg to the list of messages

\*

\* @param newMsg

\*/

void AddMsg(ReplyMessage newMsg)

{

replyMessages.add(newMsg);

}

/\*\*

\* Remove newMsg from the list of messages

\*

\* @param msg

\*/

void RemoveMsg(ReplyMessage msg)

{

replyMessages.remove(msg);

}

}

**Assignment1.java**

package assignment1;

import java.util.Scanner;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class Assignment1 {

static private Customers cust = new Customers();

static private Scanner scan = new Scanner(System.in).useDelimiter("\\n");

static private int input;

public static void main(String[] args)

{

PrintOptions();

input = GetValidInt();

System.out.println();

while(input!=0)

{

if (input==1)

{

if(cust.isLoggedIn())

{

String uname,pwrd;

int pwrdCnt=1;

System.out.print("Enter your username: ");

uname=GetValidStr();

if (cust.CustExist(uname))

{

System.out.print("Enter your password: ");

pwrd=GetValidStr();

while(pwrdCnt<3 && !cust.Login(uname, pwrd))

{

pwrdCnt++;

System.out.println("Incorrect password. Try again.");

System.out.print("Enter your password: ");

pwrd=GetValidStr();

}

if(cust.Login(uname, pwrd))

{

System.out.println(uname + " has been logged in.");

}

else

{

System.out.println("Password attempt limit has been reached.");

}

}

else

{

System.out.println("That user does not exist.");

}

}

else

{

cust.Logout();

}

}

else if(input==2)

{

System.out.println("1. Advertiser");

System.out.println("2. Responder");

System.out.println("0. Go back");

int input=GetValidInt();

if(input!=0)

{

String name;

char gender;

int age;

int income;

System.out.print("\nEnter name(username): ");

name=GetValidStr();

System.out.print("Enter gender(M/F): ");

gender=GetValidStr().charAt(0);

System.out.print("Enter age: ");

age=GetValidInt();

System.out.print("Enter income: ");

income=GetValidInt();

if(input==1)

{

String advert;

char lookingGen;

int MinAge;

int MaxAge;

int MinIncome;

int MaxIncome;

System.out.print("Enter a short introduction to yourself: ");

advert=GetValidStr();

System.out.println("The following questions are about the type of people you are looking for.");

System.out.print("Enter gender(M/F): ");

lookingGen=GetValidStr().charAt(0);

System.out.print("Enter minimum age: ");

MinAge=GetValidInt();

System.out.print("Enter maximum age: ");

MaxAge=GetValidInt();

System.out.print("Enter minimum income: ");

MinIncome=GetValidInt();

System.out.print("Enter maximum income: ");

MaxIncome=GetValidInt();

cust.add(name,gender,age,income,advert,lookingGen,MinAge,MaxAge,MinIncome,MaxIncome);

}

else if(input==2)

{

cust.add(name,gender,age,income);

}

System.out.println(name + " has been created.");

}

}

else if(input==3)

{

String name;

boolean removed;

System.out.print("Enter a username to delete: ");

name=GetValidStr();

removed=cust.remove(name);

if(!removed)

{

System.out.println("Customer does not exist.");

}

else

{

System.out.println(name + " has been removed.");

}

}

else if(input==4)

{

System.out.println("Name Gender Age Income Advert");

PrintAdvertisers();

}

else if(input==5)

{

System.out.println("Name Gender Age Income");

PrintResponders();

}

else if(input==6)

{

if(cust.isResponder())

{

int matchCount;

matchCount=cust.FindMatches();

if(matchCount>0)

{

String message;

PrintMatchesOpt();

System.out.print("Enter a number for who you would like the message: ");

input=GetValidInt();

while(input!=0)

{

System.out.println("Enter a message you would like to send to " + cust.GetMatches().get(input-1).GetName() + ": ");

message=GetValidStr();

cust.SendMessage(cust.GetMatches().get(input-1), message);

PrintMatchesOpt();

System.out.print("Enter a number for who you would like the message: ");

input=GetValidInt();

}

}

else

{

System.out.println("No matches.");

}

}

else if(cust.isAdvertiser())

{

PrintMessOpt();

input=GetValidInt();

while(input!=0)

{

if(input==1)

{

int msgToRead;

System.out.println("\nYou have " + cust.GetMsgCount()+ " messages.");

PrintMessNames();

System.out.print("Enter number of message to read: ");

msgToRead=GetValidInt();

while(msgToRead!=0)

{

if(msgToRead<=cust.GetMsgCount())

{

System.out.println("From: " + cust.GetMsgs().get(msgToRead-1).GetOwner().GetName());

System.out.println("Message: " + cust.GetMsgs().get(msgToRead-1).GetMsg());

PrintMessNames();

System.out.print("Enter number of message to read: ");

msgToRead=GetValidInt();

}

else

{

System.out.println("Invalid message number.\n");

PrintMessNames();

System.out.print("Enter number of message to read: ");

msgToRead=GetValidInt();

}

}

}

else if(input==2)

{

int msgToDelete;

System.out.println("\nYou have " + cust.GetMsgCount() + " messages.");

PrintMessNames();

System.out.print("Enter number of message to delete: ");

msgToDelete=GetValidInt();

while(msgToDelete!=0)

{

if(msgToDelete<=cust.GetMsgCount())

{

cust.RemoveMsg(msgToDelete-1);

PrintMessNames();

System.out.print("Enter number of message to delete: ");

msgToDelete=GetValidInt();

}

else

{

System.out.println("Invalid message number.\n");

PrintMessNames();

System.out.print("Enter number of message to delete: ");

msgToDelete=GetValidInt();

}

}

}

PrintMessOpt();

input=GetValidInt();

}

}

}

System.out.println();

PrintOptions();

input = GetValidInt();

System.out.println();

}

}

//Prints the options the program starts with and alters based on the type of user logged in

static void PrintOptions()

{

if (cust.isLoggedIn())

{

System.out.println("1. Login");

}

else

{

System.out.println("1. Logout");

}

System.out.println("2. Create account");

System.out.println("3. Delete account");

System.out.println("4. List all advertisers");

System.out.println("5. List all responders");

if (cust.isResponder())

System.out.println("6. Check matches");

else if(cust.isAdvertiser())

System.out.println("6. Check messages");

System.out.println("0. Exit");

}

//Print Create Account options

static void PrintCAOpt()

{

System.out.println("1. Advertiser");

System.out.println("2. Responder");

System.out.println("0. Go back");

}

//Prints match options

static void PrintMatchesOpt()

{

for (int j=0;j<cust.FindMatches();j++)

{

System.out.println((j+1) + ". " + cust.GetMatches().get(j).GetName());

}

System.out.println("0. Back");

}

//Prints message options

static void PrintMessOpt()

{

System.out.println("1. Read messages");

System.out.println("2. Delete messages");

System.out.println("0. Back");

}

//Print the options when selecting to read/delete messages

static void PrintMessNames()

{

for (int i=0;i<cust.GetMsgCount();i++)

{

ReplyMessage GetMsg=cust.GetMsgs().get(i);

System.out.println((i+1) + ". From: " + GetMsg.GetOwner().GetName());

}

System.out.println("0. Back");

}

//Prints all the advertisers in Customers

static void PrintAdvertisers()

{

for(int i=0;i<cust.GetAdvertiser().size();i++)

{

System.out.println(String.format("%-8s%-8s%-4s%-10s%s", cust.GetAdvertiser().get(i).GetName(),cust.GetAdvertiser().get(i).GetGender(),cust.GetAdvertiser().get(i).GetAge(),cust.GetAdvertiser().get(i).GetIncome(),cust.GetAdvertiser().get(i).GetAdvert()));

}

}

//Prints all the responders in Customers

static void PrintResponders()

{

for(int i=0;i<cust.GetResponder().size();i++)

{

System.out.println(String.format("%-8s%-8s%-4s%-10s", cust.GetResponder().get(i).GetName(),cust.GetResponder().get(i).GetGender(),cust.GetResponder().get(i).GetAge(),cust.GetResponder().get(i).GetIncome()));

}

}

//Gets a valid integer

static int GetValidInt()

{

int a = 0;

boolean valid=false;

do

{

if(scan.hasNextInt())

{

a=scan.nextInt();

valid=true;

}

else

{

scan.nextLine();

System.out.print("Enter a valid number: ");

}

}

while(!valid);

return a;

}

//Gets a valid string

static String GetValidStr()

{

String a = "";

boolean valid=false;

do

{

if(scan.hasNext())

{

a=scan.next();

valid=true;

}

else

{

scan.nextLine();

System.out.print("Enter a valid string: ");

}

}

while(!valid);

return a;

}

}

**Customer.java**

package assignment1;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class Customer {

/\*\*

\* Username of the customer

\*/

private String name;

/\*\*

\* Password of the customer

\*/

private String password;

/\*\*

\* Gender of the customer

\*/

private char gender;

/\*\*

\* Age of the customer

\*/

private int age;

/\*\*

\* Income of the customer

\*/

private int income;

/\*\*

\* Creates a new customer

\*

\* Sets the values of the customer to the parameters

\*

\* @param newName username of the new customer

\* @param newG gender of the new customer

\* @param newAge age of the new customer

\* @param newIncome income of the new customer

\*/

Customer(String newName, char newG,int newAge,int newIncome)

{

SetName(newName);

SetPwrd(newName+"1");

SetGender(newG);

SetAge(newAge);

SetIncome(newIncome);

}

/\*\*

\* Gets the username

\*

\* @return username of customer

\*/

String GetName()

{

return name;

}

/\*\*

\* Gets the password

\*

\* @return password of the customer

\*/

String GetPwrd()

{

return password;

}

/\*\*

\* Gets the gender

\*

\* @return gender of the customer

\*/

char GetGender()

{

return gender;

}

/\*\*

\* Gets the age

\*

\* @return age of the customer

\*/

int GetAge()

{

return age;

}

/\*\*

\* Gets the income

\*

\* @return income of the customer

\*/

int GetIncome()

{

return income;

}

/\*\*

\* Sets the name to the param

\*

\* @param newName new username

\*/

void SetName(String newName)

{

name=newName;

}

/\*\*

\* Sets the password to the param

\*

\* @param pwrd new password

\*/

void SetPwrd(String pwrd)

{

password=pwrd;

}

/\*\*

\* Sets the gender to the param

\*

\* @param newG new gender

\*/

void SetGender(char newG)

{

char gen = Character.toUpperCase(newG);

if(gen=='M' || gen=='F')

{

gender=gen;

}

else

{

System.out.println("You have entered an invalid gender. You have been assigned the default gender of M");

gender='M';

}

}

/\*\*

\* Sets the age to param

\*

\* @param newAge new age

\*/

void SetAge(int newAge)

{

if (newAge < 18)

{

newAge = 18;

System.out.println("You have entered an invalid age. Age has been set to 18");

}

age=newAge;

}

/\*\*

\* Sets the income to param

\*

\* @param newIncome new income

\*/

void SetIncome(int newIncome)

{

income=newIncome;

}

}

**Customers.java**

package assignment1;

import java.util.ArrayList;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class Customers

{

/\*\*

\* List of all customers

\*/

private ArrayList<Customer> cust = new ArrayList<>();

/\*\*

\* List of matches for the logged in customer (if responder)

\*/

private ArrayList<Customer> matches = new ArrayList<>();

/\*\*

\* Temporary customer used to determine of logged in

\*/

private Customer temp;

/\*\*

\* Customer that is logged in

\*/

private Customer loggedIn;

/\*\*

\* Create a Customers

\*

\* Set the temp to an empty Customer and then loggedIn to temp.

\* Creates 4 Advertisers and 4 Responders

\*/

Customers()

{

temp = new Customer("temp",'M',18,0);

loggedIn = temp;

cust.add(new Advertiser("Laura",'F',28,35000,"Walks on the beach",new PartnerSought('M',25,35,25000,50000)));

cust.add(new Advertiser("Larry",'M',42,35000,"Doing stuff",new PartnerSought('M',25,35,25000,50000)));

cust.add(new Advertiser("Harry",'M',26,35000,"Playing games",new PartnerSought('F',25,35,25000,50000)));

cust.add(new Advertiser("Sarah",'F',32,35000,"Not doing much",new PartnerSought('M',25,35,25000,50000)));

cust.add(new Responder("Bill",'M',22,35000));

cust.add(new Responder("Jim",'M',31,35000));

cust.add(new Responder("Ashley",'F',33,35000));

cust.add(new Responder("Lisa",'F',19,35000));

}

/\*\*

\* Add a responder to the customer list

\*

\* @param name name of responder

\* @param gender gender of responder

\* @param age age of responder

\* @param income income of responder

\*/

void add(String name,char gender,int age,int income)

{

cust.add(new Responder(name,gender,age,income));

}

/\*\*

\* Add an advertiser to the customer list

\*

\* @param name name of advertiser

\* @param gender gender of advertiser

\* @param age age of advertiser

\* @param income income of advertiser

\* @param advert advert of advertiser

\* @param lookingGen lookingGen of advertiser

\* @param MinAge MinAge of advertiser

\* @param MaxAge MaxAge of advertiser

\* @param MinIncome MinIncome of advertiser

\* @param MaxIncome MaxIncome of advertiser

\*/

void add(String name,char gender,int age,int income,String advert,char lookingGen,int MinAge,int MaxAge,int MinIncome,int MaxIncome)

{

cust.add(new Advertiser(name,gender,age,income,advert,new PartnerSought(lookingGen,MinAge,MaxAge,MinIncome,MaxIncome)));

}

/\*\*

\* Remove user with the username name

\*

\* @param name name of user to remove

\* @return true is customer was deleted and false if it didn't exist

\*/

boolean remove(String name)

{

for(int i=0;i<cust.size();i++)

{

if(cust.get(i).GetName().equals(name))

{

cust.remove(i);

return true;

}

}

return false;

}

/\*\*

\* Gets the customer at index i

\*

\* @param i index to get

\* @return customer at index i

\*/

Customer get(int i)

{

return cust.get(i);

}

/\*\*

\* Gets the customer with username name

\*

\* @param name name of user to get

\* @return customer with the username name

\*/

Customer get(String name)

{

Customer tmp = null;

for(int i=0;i<cust.size();i++)

{

if(cust.get(i).GetName().equals(name))

{

tmp = cust.get(i);

}

}

return tmp;

}

/\*\*

\* Logs in the appropriate user

\*

\* @param uname Username to log in

\* @param pword Password of user

\* @return true if user is logged in, false if user doesn't exist or password doesn't match

\*/

boolean Login(String uname, String pword)

{

for(int i=0;i<cust.size();i++)

{

if(cust.get(i).GetName().equals(uname))

{

if(cust.get(i).GetPwrd().equals(pword))

{

loggedIn=cust.get(i);

return true;

}

else

{

return false;

}

}

}

return false;

}

/\*\*

\* Resets logged in and matches array

\*/

void Logout()

{

System.out.println(loggedIn.GetName() + " has been logged out");

loggedIn=temp;

matches = new ArrayList<>();

}

/\*\*

\* Gets whether a user is logged in

\*

\* @return true if a user is logged in

\*/

boolean isLoggedIn()

{

return loggedIn.equals(temp);

}

/\*\*

\* Returns the loggedIn customer

\*

\* @return loggedIn

\*/

Customer CurrCust()

{

return loggedIn;

}

/\*\*

\* Checks if logged in user is an Advertiser

\*

\* @return true if advertiser, false otherwise

\*/

boolean isAdvertiser()

{

return loggedIn instanceof Advertiser;

}

/\*\*

\* Checks if logged in user is an Responder

\*

\* @return true if responder, false otherwise

\*/

boolean isResponder()

{

return loggedIn instanceof Responder;

}

/\*\*

\* Sends the rm to Customer c

\*

\* @param c Customer the message is going to

\* @param rm Message being sent

\*/

void SendMessage(Customer c, String rm)

{

Advertiser ad = (Advertiser)c;

for(int i=0;i<cust.size();i++)

{

if(cust.get(i) instanceof Advertiser)

{

if(cust.get(i).equals(ad))

{

((Advertiser)cust.get(i)).AddMsg(new ReplyMessage((Responder)loggedIn,rm));

System.out.println("Message sent.\n");

break;

}

}

}

}

/\*\*

\* Check if certain customer exists in customer array

\*

\* @param name username of user

\* @return true if customer exists, false otherwise

\*/

boolean CustExist(String name)

{

for(int i=0;i<cust.size();i++)

{

if(cust.get(i).GetName().equals(name))

{

return true;

}

}

return false;

}

/\*\*

\* Finds the number of matches for a user and populates the matches array

\*

\* @return count of matches

\*/

int FindMatches()

{

int mCount=0;

for (Customer cust1 : cust)

{

if(cust1 instanceof Advertiser)

{

if(CompatiblePair((Responder)loggedIn,(Advertiser)cust1))

{

matches.add(cust1);

mCount++;

}

}

}

return mCount;

}

/\*\*

\* Gets matches

\*

\* @return matches

\*/

ArrayList<Customer> GetMatches()

{

return matches;

}

/\*\*

\* Gets the number of messages a user has

\*

\* @return number of messages of current user

\*/

int GetMsgCount()

{

return ((Advertiser)loggedIn).GetMsgs().size();

}

/\*\*

\* Gets the messages a user has

\*

\* @return list of messages

\*/

ArrayList<ReplyMessage> GetMsgs()

{

return ((Advertiser)loggedIn).GetMsgs();

}

/\*\*

\* Removes a message at index mi

\*

\* @param mi index of message

\*/

void RemoveMsg(int mi)

{

((Advertiser)loggedIn).RemoveMsg(((Advertiser)loggedIn).GetMsgs().get(mi));

}

/\*\*

\* Gets list of all advertisers

\*

\* @return list of advertisers

\*/

ArrayList<Advertiser> GetAdvertiser()

{

ArrayList<Advertiser> ads = new ArrayList<>();

for(int i=0;i<cust.size();i++)

{

if(cust.get(i) instanceof Advertiser)

{

ads.add((Advertiser) cust.get(i));

//System.out.println(String.format("%-8s%-8s%-4s%-10s", cust.get(i).GetName(),cust.get(i).GetGender(),cust.get(i).GetAge(),cust.get(i).GetIncome()));

}

}

return ads;

}

/\*\*

\* Gets list of all responders

\*

\* @return list responder

\*/

ArrayList<Responder> GetResponder()

{

ArrayList<Responder> res = new ArrayList<>();

for(int i=0;i<cust.size();i++)

{

if(cust.get(i) instanceof Responder)

{

res.add((Responder) cust.get(i));

//System.out.println(String.format("%-8s%-8s%-4s%-10s", cust.get(i).GetName(),cust.get(i).GetGender(),cust.get(i).GetAge(),cust.get(i).GetIncome()));

}

}

return res;

}

/\*\*

\* Compares a responders against the search criteria of an advertiser

\*

\* @param a responder to be compared

\* @param b advertiser to be compared

\* @return false if any criteria is not met, true if all are met

\*/

private boolean CompatiblePair(Responder a, Advertiser b)

{

PartnerSought adPS = b.GetPart();

if(a.GetGender()!=adPS.GetGender())

{

return false;

}

if(a.GetAge()<adPS.GetMinAge() || a.GetAge()>adPS.GetMaxAge())

{

return false;

}

if(a.GetIncome()<adPS.GetMinIncome() || a.GetIncome()>adPS.GetMaxIncome())

{

return false;

}

return true;

}

}

**PartnerSought.java**

package assignment1;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class PartnerSought {

/\*\*

\* Gender being looked for

\*/

private char gender;

/\*\*

\* Minimum age being looked for

\*/

private int MinAge;

/\*\*

\* Maximum age being looked for

\*/

private int MaxAge;

/\*\*

\* Minimum income being looked for

\*/

private int MinIncome;

/\*\*

\* Maximum income being looked for

\*/

private int MaxIncome;

/\*\*

\* Creates a PartnerSought

\*

\* Creates a PartnerSought with the params

\*

\* @param newGen

\* @param minA

\* @param maxA

\* @param minI

\* @param maxI

\*/

PartnerSought(char newGen,int minA,int maxA,int minI,int maxI)

{

SetGender(newGen);

SetMinAge(minA);

SetMaxAge(maxA);

SetMinIncome(minI);

SetMaxIncome(maxI);

}

/\*\*

\* Gets the gender

\*

\* @return gender

\*/

char GetGender()

{

return gender;

}

/\*\*

\* Get the minimum ages

\*

\* @return MinAge

\*/

int GetMinAge()

{

return MinAge;

}

/\*\*

\* Gets the maximum age

\*

\* @return MaxAge

\*/

int GetMaxAge()

{

return MaxAge;

}

/\*\*

\* Gets the minimum income

\*

\* @return MinIncome

\*/

int GetMinIncome()

{

return MinIncome;

}

/\*\*

\* Gets the maximum income

\*

\* @return MaxIncome

\*/

int GetMaxIncome()

{

return MaxIncome;

}

/\*\*

\* Set gender to newG. If newG is not M or F, default is set

\*

\* @param newG

\*/

void SetGender(char newG)

{

char gen = Character.toUpperCase(newG);

if(gen=='M' || gen=='F')

{

gender=gen;

}

else

{

System.out.println("You have entered an invalid gender. You have been assigned the default gender of M");

gender='M';

}

}

/\*\*

\* Sets minimum age to age. If age is below 18, minimum age is set to 18

\*

\* @param age

\*/

void SetMinAge(int age)

{

if (age>18)

{

MinAge=age;

}

else

{

System.out.println("The age values entered were not valid. Minimum age has been set to 18.");

MinAge=18;

}

}

/\*\*

\* Set maximum age to age. If maximum age is less than minimum age, maximum age is set to minimum age plus 1

\*

\* @param age

\*/

void SetMaxAge(int age)

{

if(age>MinAge)

{

MaxAge=age;

}

else

{

System.out.println("The age values entered were not valid. Maximum age has been set to minimum age plus 1.");

MaxAge=MinAge+1;

}

}

/\*\*

\* Sets minimum income to income. If income is below 0, minimum income is set to 0

\*

\* @param income

\*/

void SetMinIncome(int income)

{

if(income<0)

{

MinIncome=0;

}

else

{

System.out.println("The income values entered were not valid. Minimum income has been set to 0.");

MinIncome=income;

}

}

/\*\*

\* Set maximum income to income. If maximum income is less than minimum income, maximum income is set to minimum income plus 1

\*

\* @param income

\*/

void SetMaxIncome(int income)

{

if(income>MinIncome)

{

MaxIncome=income;

}

else

{

System.out.println("The income values entered were not valid. Maximum income has been set to minimum income plus 1.");

MaxIncome=MinIncome+1;

}

}

}

**ReplyMessage.java**

package assignment1;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class ReplyMessage {

/\*\*

\* Customer who sent the message

\*/

private Responder owner;

/\*\*

\* Text of the message

\*/

private String msgText;

/\*\*

\* Creates a ReplyMessage

\*

\* Creates a new ReplyMessage using the given params

\*

\* @param newO

\* @param newMsg

\*/

ReplyMessage(Responder newO,String newMsg)

{

SetOwner(newO);

SetMsg(newMsg);

}

/\*\*

\* Sets owner to newO

\*

\* @param newO

\*/

void SetOwner(Responder newO)

{

owner = newO;

}

/\*\*

\* Gets owner

\*

\* @return owner

\*/

Responder GetOwner()

{

return owner;

}

/\*\*

\* Sets msgText to newMsg

\*

\* @param newMsg

\*/

void SetMsg(String newMsg)

{

msgText = newMsg;

}

/\*\*

\* Get msgText

\*

\* @return msgText

\*/

String GetMsg()

{

return msgText;

}

}

**Responder.java**

package assignment1;

/\*\*

\* @author Samuel Brownley 31691379

\* @version 1.0

\*/

public class Responder extends Customer{

/\*\*

\* Creates a new responder

\*

\* Calls the parent constructor with the params

\*

\* @param name username of the new responder

\* @param newG gender of the new responder

\* @param newAge age of the new responder

\* @param newIncome income of the new responder

\*/

Responder(String name,char newG,int newAge,int newIncome)

{

super(name,newG,newAge,newIncome);

}

}