**Reviewing Assignment**

Lab Assignment 8

|  |  |
| --- | --- |
| Started: | Nov 16, 2014 11:34 PM |
| Finished: | Nov 20, 2014 1:33 PM - late |

 1 of 1

**Lab Assignment 8     Total Grade:  (of possible 20 points)**

**Score: 8   (of possible 20 points)**

**Assignment 8 - Computer Dating**

Select one option from below.  All (both) options are worth the same number of points.  The more advanced option(s) are provided for students who find the basic one too easy and want more of a challenge.   Make sure you have read and understood

* both ***modules A*** and ***B*** this week, and
* ***module 2R - Lab Homework Requirements***

before submitting this assignment. Hand in only one program, please.

OPTION A (Basic) Date Profiles

**Class DateProfile**

Create a class called **DateProfile** that has the following ***private instance members***:

* **gender** - a **char**, the gender of the applicant ('M' or 'F').
* **searchGender** - a **char**, the gender of desired partner ('M' or 'F'). This is not the gender of the applicant, but of the applicant's requested partner.
* **romance** - an **int** from 1 to 10, indicating the importance of romance to the applicant.
* **finance** - an **int** from 1 to 10, indicating the importance of finance to the applicant.
* **name** - a **String** indicating the full name of the applicant.

Each object in the **DateProfile** class represents an applicant's profile.  If the object is ('M', 'F', 7, 4, "Hugh Hefner") then the applicant's name is "Hugh Hefner", he's looking for a date who is Female, with romance being somewhat important (7) and finance being less important (4).

Create***public static constants*** for:

* All range limits (like **MIN\_ROMANCE**, **MIN\_NAME\_LEN**, etc.).  These are the limits that the mutators test for.
* All default values (like **DEFAULT\_GEND**, **DEFAULT\_SEARCH\_GEND**, **DEFAULT\_NAME**).  These are used if the default constructor is called or if a parameter-taking constructor is called but a bad value (out-of-range) is detected.  However, they are not used directly in the constructors, as we will find that there are helper methods that do the dirty work for the constructors -- see below.

You should supply all of the following ***member functions*** of class **DateProfile** (at a minimum):

* ***Accessors*** and ***Mutators*** for each field (instance member). For example: **char getGender()** and **boolean setGender(char gdr)**.  In addition to individual mutators, create a public **void setAll( ... )** that takes all five parameters and acts like a mutator, except that it has a void return type, i.e., in this one, you do not have to report bad parameters back to the user, even though you would still filter them out.  Also, create a public **void setDefaults()** that sets all five members to their default values.
* ***Constructors*** that take no parameters (default) and all 5 parameters.  To avoid code duplication make sure that the parameter-taking constructor utilizes the above mutators rather than do anything direct assignments or testing.
* **double fitValue(DateProfile partner)**, which returns a number from 0.0 (very bad fit) to 1.0 (perfect fit).  The *public instance method*compares the calling object (**this**) to the object passed as a parameter.  This method should call three ***private*** methods**determineGenderFit( ... ), determineRomanceFit( ... )**and**determineFinanceFit( ... )**, that will be used to return intermediate results for each of the three factors.  It should compute the final fit value that it returns by returning a 0 if gender is not compatible (regaredless of the other values) and return the ***average*** of finance and romance values if gender is compatible.
* **double determineGenderFit(DateProfile partner)** This *private instance method*returns either a 0 or 1 depending on the gender compatibility of the calling object and the passed parameter object.  You have to compare gender compatibility completely: i.e., there must be *mutual* consent on this one!  It is used by the public**fitValue()**, not directly by the client**main()**.
* **double determineRomanceFit(DateProfile partner)** This *private instance method*returns a number from (but not including) 0.0 to (and including) 1.0 depending on the **romance** compatibility of the calling object and the passed parameter object. The **romance** numbers should be highest (1.0) if the two values are equal (both 3, both 5, both 7) and lowest (perhaps a small non-zero value like .1)  if their difference is 9. It is used by the public**fitValue()**, not directly by the client**main()**.
* **double determineFinanceFit(DateProfile partner)** This *private instance method*returns a number  from (but not including) 0.0 to (and including) 1.0 depending on the **finance** compatibility of the calling object and the passed parameter object. The **finance** numbers should be highest (1.0) if the two values are equal (both 3, both 5, both 7) and lowest (perhaps a small non-zero value like .1)  if their difference is 9. It is used by the public**fitValue()**, not directly by the client**main()**.

**DateProfile** should be a class *distinct* from (and not contained within) your main class (which we call **Foothill**).  However, you can and should defined it as a non-public class so it can reside in the same file, **Foothill.java**.

**The Client Driver**

In addition to**main()** supply a static method of class **Foothill** that compares and displays two **DateProfile** objects:

* **static void displayTwoProfiles( DateProfile profile1, DateProfile profile2 )**  This method will print the names of the two objects and show the fit value.  It's output for a single call will look like this:

Fit between Steve Nobody and Helen Anybody:

0.7222222222222222

From your **main()** inside **Foothill**, you will  instantiate a total of four **DateProfile** objects, **applicant1, ... applicant4**manually from literal values in your program, i.e., do not involve the user with run-time input.  Then for each of the four applicants, display the ***fits*** with the others - including themselves.  Do this by calling**displayTwoProfiles()** for all possible combinations of the four applicants producing 16 comparison figures.  (You will be comparing each applicant to him/herself in each of these four groups.  This will serve to check whether the result is correct - it must be either a 1 or a 0 depending on the **searchGender** they requested, but never a number between (can you see why?)).  Here is part of a sample output:

Fit between Joe Somebody and Joe Somebody:

0.0

Fit between Joe Somebody and Steve Nobody:

0.0

Fit between Joe Somebody and Jane Peabody:

0.8888888888888888

Fit between Joe Somebody and Helen Anybody:

0.8333333333333333

Fit between Steve Nobody and Joe Somebody:

0.0

Fit between Steve Nobody and Steve Nobody:

0.0

Fit between Steve Nobody and Jane Peabody:

0.6666666666666667

...

Make sure all mutators, constructors and other methods that affect private data adequately test for illegal values and, if possible, return a **bool** that reports the results of this test.

OPTION B (Intermediate) Beautify and Increase Precision

Change the romance and finance to double type and adjust the precision of the determine fit methods so that even small differences (.01) in values result in different fit values.  Also, make the output look beautiful by using number formatting techniques.

.

**Answer**

* text/plain[foothillAssignment8.txt](https://myetudes.org/access/mneme/content/private/mneme/cff3240c-b51c-41f6-80dc-4db4530bdd05/submissions/15226079/238b7c7d-35dd-4863-806d-fc969a13633e/foothillAssignment8.txt)

[[https://myetudes.org/ambrosia_library/icons/collapse.gif](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/15226079/list) Model Answer](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/15226079/list)

/\* CS 1A Lab 8

 \* Instructor Solution

 \*/

public class Foothill

{

   public static void main(String[] args)

   {

      DateProfile app1 = new DateProfile('m', 'f', 5, 5, "Joe Somebody"),

      app2 = new DateProfile('m', 'f', 1, 10, "Steve Nobody"),

      app3 = new DateProfile('f', 'f', 4, 7, "Jane Peabody"),

      app4 = new DateProfile('f', 'm', 10, 1, "Helen Anybody");

      // compare everyone to app1

      displayTwoProfiles(app1, app1);

      displayTwoProfiles(app1, app2);

      displayTwoProfiles(app1, app3);

      displayTwoProfiles(app1, app4);

      // compare everyone to app2

      displayTwoProfiles(app2, app1);

      displayTwoProfiles(app2, app2);

      displayTwoProfiles(app2, app3);

      displayTwoProfiles(app2, app4);

      // compare everyone to app3

      displayTwoProfiles(app3, app1);

      displayTwoProfiles(app3, app2);

      displayTwoProfiles(app3, app3);

      displayTwoProfiles(app3, app4);

      // compare everyone to app4

      displayTwoProfiles(app4, app1);

      displayTwoProfiles(app4, app2);

      displayTwoProfiles(app4, app3);

      displayTwoProfiles(app4, app4);

      // prove a mutator

      if ( app4.setGender('Q') )

         System.out.println("Q accepted as gender");

      else

         System.out.println("Q rejected as gender");

   }

   static void displayTwoProfiles( DateProfile profile1, DateProfile profile2 )

   {

      System.out.println("Fit between " + profile1.getName() + " and "

         + profile2.getName() + ":");

      System.out.println( "   " + profile1.fitValue(profile2) );

   }

}

// class DateProfile ---------------------------------------------------

class DateProfile

{

   // private data

   private char gender;

   private char searchGender;

   private int romance;

   private int finance;

   private String name;

   //static constants;

   public static final int MIN\_VAL = 1;

   public static final int MAX\_VAL = 10;

   public static final int MIN\_NAME\_LENGTH = 3;

   public static final int MAX\_NAME\_LENGTH = 100;

   // defaults

   public static final int DEFAULT\_FINANCE = 1;

   public static final int DEFAULT\_ROMANCE = 1;

   public static final char DEFAULT\_GENDER  = 'F';

   public static final char DEFAULT\_SEARCH\_GENDER  = 'M';

   public static final String DEFAULT\_NAME = "(undefined)";

   // constructors

   DateProfile()

   {

      setDefaults();

   }

   DateProfile(char gen, char srchGen, int rom, int fin, String nm)

   {

      setDefaults();  // in case of error in next call

      setAll(gen, srchGen, rom, fin, nm);

   }

   // mutators -- do not revert to defaults if error

   void setAll(char gen, char srchGen, int rom, int fin, String nm)

   {

      setGender(gen);

      setSearchGender(srchGen);

      setRomance(rom);

      setFinance(fin);

      setName(nm);

   }

   void setDefaults()

   {

      gender = DEFAULT\_GENDER;

      searchGender = DEFAULT\_SEARCH\_GENDER;

      finance = DEFAULT\_FINANCE;

      romance = DEFAULT\_ROMANCE;

      name = DEFAULT\_NAME;

   }

   boolean setGender(char gen)

   {

      if ( !validGender(gen) )

         return false;

      // always store gender as upper case

      gender = Character.toUpperCase(gen);

      return true;

   }

   boolean setSearchGender(char gen)

   {

      if ( !validGender(gen) )

         return false;

      searchGender = Character.toUpperCase(gen);

      return true;

   }

   boolean setRomance(int val)

   {

      if ( val < MIN\_VAL || val > MAX\_VAL )

         return false;

      romance = val;

      return true;

   }

   boolean setFinance(int val)

   {

      if ( val < MIN\_VAL || val > MAX\_VAL )

         return false;

      finance = val;

      return true;

   }

   boolean setName(String nm)

   {

      if (nm.length() < MIN\_NAME\_LENGTH || nm.length() > MAX\_NAME\_LENGTH )

         return false;

      name = nm; // this could have been name = new Name(nm);

      return true;

   }

   // accessors (ok to use brief style in these

   char getGender() { return gender; }

   char getSearchGender() { return searchGender; }

   int getRomance() { return romance; }

   int getFinance() { return finance; }

   String getName() { return name; }

   static private boolean validGender(char gen)

   {

      char newGender = Character.toLowerCase(gen);

      if ( newGender != 'm' && newGender != 'f' )

         return false;

      else

         return true;

   }

   // helpers

   private double determineFinanceFit(DateProfile partner)

   {

      int myVal, partnersVal, diff;

      double fit;

      myVal = finance;

      partnersVal = partner.finance;

      diff = Math.abs(myVal - partnersVal);

      fit = MAX\_VAL  - diff; // 10 is largest and 1 is the smallest

      fit = (fit - 1) \* (9.9/9);  // 9.9 is largest and 0 is the smallest

      fit += .1;     // 10 is largest and .1 is the smallest

      fit = fit / (double)MAX\_VAL; // now goes from 0.01 to 1.0

      return fit;

   }

   private double determineRomanceFit(DateProfile partner)

   {

      int myVal, partnersVal, diff;

      double fit;

      myVal = romance;

      partnersVal = partner.romance;

      diff = Math.abs(myVal - partnersVal);

      fit = MAX\_VAL  - diff; // 10 is largest and 1 is the smallest

      fit = (fit - 1) \* (9.9/9);  // 9.9 is largest and 0 is the smallest

      fit += .1;     // 10 is largest and .1 is the smallest

      fit = fit / (double)MAX\_VAL; // now goes from 0.01 to 1.0

      return fit;

   }

   private double determineGenderFit(DateProfile partner)

   {

      if (searchGender != partner.gender)

         return 0.0;

      if (partner.searchGender != gender)

         return 0.0;

      return 1.0;

   }

   // computational

   public double fitValue(DateProfile partner)

   {

      double gendResult, finResult, romResult, returnResult;

      // compute individual results for easy debugging and readability

      gendResult =  determineGenderFit(partner);

      romResult = determineRomanceFit(partner);

      finResult = determineFinanceFit(partner);

      // form the return value for easy debugging

      returnResult = gendResult \* (romResult + finResult)/2. ;

      return returnResult;

   }

}

/\* ------------------ Output of Above ------------------------------

Fit between Joe Somebody and Joe Somebody:

   0.0

Fit between Joe Somebody and Steve Nobody:

   0.0

Fit between Joe Somebody and Jane Peabody:

   0.0

Fit between Joe Somebody and Helen Anybody:

   0.505

Fit between Steve Nobody and Joe Somebody:

   0.0

Fit between Steve Nobody and Steve Nobody:

   0.0

Fit between Steve Nobody and Jane Peabody:

   0.0

Fit between Steve Nobody and Helen Anybody:

   0.01

Fit between Jane Peabody and Joe Somebody:

   0.0

Fit between Jane Peabody and Steve Nobody:

   0.0

Fit between Jane Peabody and Jane Peabody:

   1.0

Fit between Jane Peabody and Helen Anybody:

   0.0

Fit between Helen Anybody and Joe Somebody:

   0.505

Fit between Helen Anybody and Steve Nobody:

   0.01

Fit between Helen Anybody and Jane Peabody:

   0.0

Fit between Helen Anybody and Helen Anybody:

   0.0

Q rejected as gender

------------------------------------------------------------------- \*/

**Comments**

CONSTRUCTORS  
\* The default constructor assigns default values correctly.  
\* The 5 parameter constructor calls the mutators as it should.  
  
DATA FILTERING / MUTATORS  
\* You are filtering data so that no bad values can be set.  
\* Good job using constants in place of literal values.  
- Your mutator functions should not be outputting an error message. How the error is handled should be up to the calling method.   
- Validators should be labeled static since they are not instance methods.

DETERMINING FIT  
\* Nice job coming up with a formula that correctly calculates the romance and finance fits.  
\* You are successfully passing a DateProfile object. This was a tricky part, so kudos on that.  
\* You have checked for mutual compatability in determining the gender fit.  
- fitValue() does not use the expression which was given in the spec. See my solution for the correct calculation.   
- Your determineGenderFit() did not correctly compare the two objects.  For instance, two heterosexual females will match each other in the second case.

MAIN / DISPLAY / ETC  
\* main() is exactly what the spec requires.  
\* You are displaying your results from the helper method, displayTwoProfiles(), which is just what we want.  
\* You seem to have a good grasp on writing and using your own classes.

Good work, Dmitri. There are some issues, though. Be sure to carefully read the comments and examine the samle solution.

 1 of 1

public class Foothill

{

public static void main(String[] args)

{

DateProfile applicant1, applicant2, applicant3, applicant4;

applicant1 = new DateProfile('M', 'M', 8, 2, "Tom Cruise" );

applicant2 = new DateProfile('M', 'F', 2, 11, "Bill Gates");

applicant3 = new DateProfile('F','F', 3, 4, "Fox");

applicant4 = new DateProfile('F', 'M', 5, 7, "Angelina Jolie");

DateProfile[] applicantArray = {applicant1, applicant2,

applicant3, applicant4};

for (int k = 0; k < applicantArray.length; k++)

{

System.out.println(applicantArray[k].getName() + " fit score:");

for (int i = 0; i < applicantArray.length; i++)

displayTwoProfiles(applicantArray[k], applicantArray[i]);

System.out.println("");

}

}

static void displayTwoProfiles(DateProfile profile1, DateProfile profile2)

{

System.out.println("Fit between " + profile1.getName() + " and "

+ profile2.getName() + ": ");

System.out.println(profile1.fitValue(profile2));

}

}

class DateProfile

{

private char gender;

private char searchGender;

private int romance;

private int finance;

private String name;

public static final int MIN\_ROMANCE = 1;

public static final int MAX\_ROMANCE = 10;

public static final int MIN\_FINANCE = 1;

public static final int MAX\_FINANCE = 10;

public static final int MIN\_NAME\_LEN = 4;

public static final int MAX\_NAME\_LEN = 32;

public static final char MALE\_GENDER = 'M';

public static final char FEMALE\_GENDER = 'F';

public static final char DEFAULT\_GENDER = 'M';

public static final char DEFAULT\_SEARCH\_GENDER = 'F';

public static final int DEFAULT\_ROMANCE = 10;

public static final int DEFAULT\_FINANCE = 1;

public static final String DEFAULT\_NAME = "Undefined Name";

public DateProfile(char gender, char searchGender, int romance,

int finance, String name)

{

setAll(gender, searchGender, romance, finance, name);

}

public void setDefaults()

{

gender = DEFAULT\_GENDER;

searchGender = DEFAULT\_SEARCH\_GENDER;

romance = DEFAULT\_ROMANCE;

finance = DEFAULT\_FINANCE;

name = DEFAULT\_NAME;

}

public void setAll(char gender, char searchGender, int romance,

int finance, String name)

{

if (!setName(name))

this.name = DEFAULT\_NAME;

if (!setGender(gender))

this.gender = DEFAULT\_GENDER;

if (!setSearchGender(searchGender))

this.searchGender = DEFAULT\_SEARCH\_GENDER;

if (!setRomance(romance))

this.romance = DEFAULT\_ROMANCE;

if (!setFinance(finance))

this.finance = DEFAULT\_FINANCE;

}

public boolean setGender(char gdr)

{

if (!validGender(gdr))

{

System.out.println("Gender of " + this.name

+ " is neither male or female");

System.out.println("");

return false;

}

this.gender = gdr;

return true;

}

public boolean setSearchGender(char searchGender)

{

if (!validSearchGender(searchGender))

{

System.out.println("Search gender for " + this.name

+ " is neither male or female");

System.out.println("");

return false;

}

this.searchGender = searchGender;

return true;

}

public boolean setRomance(int romance)

{

if (!validRomance(romance))

{

System.out.println("Romance of " + this.name

+ " does not fall within a range of 1-10");

System.out.println("");

return false;

}

this.romance = romance;

return true;

}

public boolean setFinance(int finance)

{

if (!validFinance(finance))

{

System.out.println("Finance of " + this.name

+ " does not fall within a range of 1-10");

System.out.println("");

return false;

}

this.finance = finance;

return true;

}

public boolean setName(String name)

{

if (!validName(name))

{

System.out.println("Name \"" + name +"\""

+ " is either too long or too short");

System.out.println("");

return false;

}

this.name = name;

return true;

}

private boolean validGender(char gender)

{

if (gender == MALE\_GENDER || gender == FEMALE\_GENDER)

return true;

return false;

}

private boolean validSearchGender(char searchGender)

{

return validGender(searchGender);

}

private boolean validRomance(int romance)

{

if (romance > MAX\_ROMANCE || romance < MIN\_ROMANCE)

return false;

return true;

}

private boolean validFinance(int finance)

{

if (finance > MAX\_FINANCE || finance < MIN\_FINANCE)

return false;

return true;

}

private boolean validName(String name)

{

if (name.length() < MIN\_NAME\_LEN || name.length() > MAX\_NAME\_LEN)

return false;

return true;

}

public char getGender()

{

return this.gender;

}

public char getSearchGender()

{

return this.searchGender;

}

public int getRomance()

{

return this.romance;

}

public int getFinance()

{

return this.finance;

}

public String getName()

{

return this.name;

}

double fitValue(DateProfile partner)

{

return determineGenderFit(partner) \* determineFinanceFit(partner)

\* determineRomanceFit(partner);

}

private double determineRomanceFit(DateProfile partner)

{

double aplRomanceA;

double aplRomanceB;

aplRomanceA = romance - partner.romance;

aplRomanceB = 1 - (Math.abs(aplRomanceA))/10;

return aplRomanceB;

}

private double determineGenderFit(DateProfile partner)

{

double doesItFit = 0;

if (partner.gender != gender)

{

if (searchGender != partner.searchGender)

doesItFit = 1.0;

else

doesItFit = 0.0;

}

else if (partner.gender == gender)

{

if (searchGender == partner.searchGender)

doesItFit = 1.0;

else

doesItFit = 0.0;

}

return doesItFit;

}

private double determineFinanceFit(DateProfile partner)

{

double aplFinanceA;

double aplFinanceB;

aplFinanceA = finance - partner.finance;

aplFinanceB = 1 - (Math.abs(aplFinanceA))/10;

return aplFinanceB;

}

}

/\*----------paste of run from console window------------

Finance of Bill Gates does not fall within a range of 1-10

Name "Fox" is either too long or too short

Tom Cruise fit score:

Fit between Tom Cruise and Tom Cruise:

1.0

Fit between Tom Cruise and Bill Gates:

0.0

Fit between Tom Cruise and Undefined Name:

0.4

Fit between Tom Cruise and Angelina Jolie:

0.0

Bill Gates fit score:

Fit between Bill Gates and Tom Cruise:

0.0

Fit between Bill Gates and Bill Gates:

1.0

Fit between Bill Gates and Undefined Name:

0.0

Fit between Bill Gates and Angelina Jolie:

0.27999999999999997

Undefined Name fit score:

Fit between Undefined Name and Tom Cruise:

0.4

Fit between Undefined Name and Bill Gates:

0.0

Fit between Undefined Name and Undefined Name:

1.0

Fit between Undefined Name and Angelina Jolie:

0.0

Angelina Jolie fit score:

Fit between Angelina Jolie and Tom Cruise:

0.0

Fit between Angelina Jolie and Bill Gates:

0.27999999999999997

Fit between Angelina Jolie and Undefined Name:

0.0

Fit between Angelina Jolie and Angelina Jolie:

1.0

----------------------------------------------------- \*/