19

```
// Dating for Us Geeks
#include <iostream>
#include <string>
#include <sstream>
#include <cmath>

using namespace std;

class DatingProfile
{
private:
   string name;
   char gender;
   char searchGender;
   int romance;
   int finance;

   static bool validRomance(int romance);
   static bool validFinance(int finance);
   static bool validName(string name);

   double determinGenderFit(DatingProfile partner);
   double determineRomanceFit(DatingProfile partner);
   double determineFinanceFit(DatingProfile partner);

public:
   static const int MIN_ROMANCE = 1;
   static const int MAX_ROMANCE = 10;
   static const int MIN_FINANCE = 1;
   static const int MAX_FINANCE = 10;
   static const int MIN_NAME_LEN = 1;
   static const int MAX_NAME_LEN = 20;
   static const char DEFAULT_GENDER = 'F';
   static const char DEFAULT_SEARCHGENDER = 'M';
   static const int DEFAULT_ROMANCE = 5;
   static const int DEFAULT_FINANCE = 5;
   static const string DEFAULT_NAME;
   double fitValue(DatingProfile partner);

   DatingProfile();
   DatingProfile(string theName, char gend, char searchGend, int rom, int fin);

   void setDefaults();
   void setAll(string theName, char gend, char searchGend, int rom, int fin);

   char getGender() { return gender; }
   bool setGender(char gend);
   char getSearchGender() { return searchGender; }
   bool setSearchGender(char searchGend);
   int getRomance() { return romance; }
   bool setRomance(int rom);
   int getFinance() { return finance; }
   bool setFinance(int fin);
   string getName() { return name; }
   bool setName(string theName);
};
DatingProfile::DatingProfile()
{
   setDefaults();
}
void displayTwoProfiles(DatingProfile profile1, DatingProfile profile2)
{
   cout << "Fit between " << profile1.getName() << " and " << profile2.getName()
 << " \n" << profile1.fitValue(profile2) << "\n";
}
const string DatingProfile::DEFAULT_NAME = " Unknown ";
int main()
{
   DatingProfile
      applicate1("Damia Rose", 'F', 'M', 9, 8),
      applicate2("Kai Ocean", 'M', 'F', 7, 5),
      applicate3("Yuki Snow", 'F', 'M', 4, 9),
      applicate4("Auron Fantasy", 'M', 'F', 3, 4);

   displayTwoProfiles(applicate1, applicate1);
   displayTwoProfiles(applicate1, applicate2);
   displayTwoProfiles(applicate1, applicate3);
   displayTwoProfiles(applicate1, applicate4);
   displayTwoProfiles(applicate2, applicate1);
   displayTwoProfiles(applicate2, applicate2);
   displayTwoProfiles(applicate2, applicate3);
   displayTwoProfiles(applicate2, applicate4);
   displayTwoProfiles(applicate3, applicate1);
   displayTwoProfiles(applicate3, applicate2);
   displayTwoProfiles(applicate3, applicate3);
   displayTwoProfiles(applicate3, applicate4);
   displayTwoProfiles(applicate4, applicate1);
   displayTwoProfiles(applicate4, applicate2);
   displayTwoProfiles(applicate4, applicate3);
   displayTwoProfiles(applicate4, applicate4);

   applicate1.setDefaults();
   applicate2.setDefaults();
   applicate3.setDefaults();
   applicate4.setDefaults();

   displayTwoProfiles(applicate1, applicate1);
   displayTwoProfiles(applicate1, applicate2);
   displayTwoProfiles(applicate1, applicate3);
   displayTwoProfiles(applicate1, applicate4);
   displayTwoProfiles(applicate2, applicate1);
   displayTwoProfiles(applicate2, applicate2);
   displayTwoProfiles(applicate2, applicate3);
   displayTwoProfiles(applicate2, applicate4);
   displayTwoProfiles(applicate3, applicate1);
   displayTwoProfiles(applicate3, applicate2);
   displayTwoProfiles(applicate3, applicate3);
   displayTwoProfiles(applicate3, applicate4);
   displayTwoProfiles(applicate4, applicate1);
   displayTwoProfiles(applicate4, applicate2);
   displayTwoProfiles(applicate4, applicate3);
   displayTwoProfiles(applicate4, applicate4);
}
DatingProfile::DatingProfile(string theName, char gend, char searchGend, int rom
, int fin)
{
   if (!setGender(gend))
      setGender('M');
   else
      gend = DEFAULT_GENDER;
   if (!setSearchGender(searchGend))
      setSearchGender('F');
   else
      searchGend = DEFAULT_SEARCHGENDER;
   if (!setRomance(rom))
      rom = DEFAULT_ROMANCE;
   if (!setFinance(fin))
      fin = DEFAULT_FINANCE;
   if (!setName(theName))
      theName = DEFAULT_NAME;
}
bool DatingProfile::setGender(char gend)
{
   if (gend != 'F' && gend != 'M')
      return false;
   gender = gend;
   return true;
}
bool DatingProfile::setSearchGender(char searchGend)
{
   if (searchGend != 'M' && searchGend != 'F')
      return false;
   searchGender = searchGend;
   return true;
}
bool DatingProfile::setRomance(int rom)
{
   if (!validRomance(rom))
      return false;
   romance = rom;
   return true;
}
bool DatingProfile::setFinance(int fin)
{
   if (!validFinance(fin))
      return false;
   finance = fin;
   return true;
}
bool DatingProfile::setName(string theName)
{
   if (!validName(theName))
      return false;
   name = theName;
   return true;
}
bool DatingProfile::validRomance(int rom)
{
   if (rom >= MIN_ROMANCE && rom <= MAX_ROMANCE)
      return true;
   return false;
}
bool DatingProfile::validFinance(int fin)
{
   if (fin >= MIN_FINANCE && fin <= MAX_FINANCE)
      return true;
   return false;
}
bool DatingProfile::validName(string theName)
{
   if (theName.length() >= MIN_NAME_LEN && theName.length() <= MAX_NAME_LEN)
      return true;
   return false;
}
void DatingProfile::setDefaults()
{
   gender = DEFAULT_GENDER;
   searchGender = DEFAULT_SEARCHGENDER;
   romance = DEFAULT_ROMANCE;
   finance = DEFAULT_FINANCE;
   name = DEFAULT_NAME;
}
double DatingProfile::determinGenderFit(DatingProfile partner)
{
   if (getGender() != partner.getGender())
      return 1.0;
   else
      return 0.0;
}
double DatingProfile::determineRomanceFit(DatingProfile partner)
{
   double results;
   if (getRomance() == partner.getRomance())
      results = 1.0;
   else
   {
      double difference;
      difference = abs(partner.getRomance() - getRomance());
      results = (10 - difference) / 10.0;
   }
   return results;
}
double DatingProfile::determineFinanceFit(DatingProfile partner)
{
   double results;
   if (getFinance() == partner.getFinance())
      results = 1.0;
   else
   {
      double difference;
      difference = abs(partner.getFinance() - getFinance());
      results = (10 - difference) / 10.0;
   }
   return results;
}
double DatingProfile::fitValue(DatingProfile partner)
{
   double fitVal = determinGenderFit(partner)*
      determineRomanceFit(partner)* determineFinanceFit(partner);
   return fitVal;
}

/*---------------------------- Posted Run -------------------------------------
-

Fit between Damia Rose and Damia Rose
0
Fit between Damia Rose and Kai Ocean
0.56
Fit between Damia Rose and Yuki Snow
0
Fit between Damia Rose and Auron Fantasy
0.24
Fit between Kai Ocean and Damia Rose
0.56
Fit between Kai Ocean and Kai Ocean
0
Fit between Kai Ocean and Yuki Snow
```

-1

```
0.42
Fit between Kai Ocean and Auron Fantasy
0
Fit between Yuki Snow and Damia Rose
0
Fit between Yuki Snow and Kai Ocean
0.42
Fit between Yuki Snow and Yuki Snow
0
Fit between Yuki Snow and Auron Fantasy
0.45
Fit between Auron Fantasy and Damia Rose
0.24
Fit between Auron Fantasy and Kai Ocean
0
Fit between Auron Fantasy and Yuki Snow
0.45
Fit between Auron Fantasy and Auron Fantasy
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Fit between  Unknown  and  Unknown
0
Press any key to continue . . .
--------------------------------------------------------------------------------
-*/
```

```
// Lab 08 - Instructor Solution:
// Original - Prof. Loceff, Updates, Edits, Annotations: &
//
//Notes:
//- Correct access qualifiers (private/public)
//- Correct use of getters/setters
//- Correct use of global consts
//- Use of symbolic consts rather than literals (magics)
//- No output in interior methods
//-
//- Faithfulness to spec

#include <iostream>
#include <string>
using namespace std;

// DateProfile Class Prototype
class DateProfile {
private:
    char gender;
    char searchGender;
    int romance;
    int finance;
    string name;

public:
    // int static consts can be initialized in-line like so:
    static const int MIN_VAL = 1;
    static const int MAX_VAL = 10;
    static const int MAX_NAME_LENGTH = 100;
    static const int MIN_NAME_LENGTH = 3;

    // defaults
    static const int DEFAULT_FINANCE = 1;
    static const int DEFAULT_ROMANCE = 1;
    static const char DEFAULT_GENDER = 'F';
    static const char DEFAULT_SEARCH_GENDER = 'M';
    static const string DEFAULT_NAME;

    // constructors
    DateProfile();
    DateProfile(char theGender, char theSearchGender,
                int theRomance, int theFinance, string theName);

    // mutators
    bool setGender(char theGender);
    bool setSearchGender(char theSearchGender);
    bool setRomance(int theRomance);
    bool setFinance(int theFinance);
    bool setName(string theName);
    void setDefaults();
    void setAll(char theGender, char theSearchGender, int theRomance,
                int theFinance, string theName);
    // accessors
    char getGender() { return gender; }
    char getSearchGender() { return searchGender; }
    int getRomance() { return romance; }
    int getFinance() { return finance; }
    string getName() { return name; }

    // computational
    double fitValue(DateProfile partner);

private:
    // helpers
    bool validGender(char theGender);
    double determineGenderFit(DateProfile partner);
    double determineRomanceFit(DateProfile partner);
    double determineFinanceFit(DateProfile partner);
};

// static initializations
string const DateProfile::DEFAULT_NAME = "(undefined)";

// global method prototypes
void displayTwoProfiles(DateProfile profile1, DateProfile profile2);

// Main Program (Client)
int main() {
    DateProfile app1('m', 'f', 2, 8, "Joe Somebody"),
    app2('m', 'f', 5, 5, "Steve Nobody"),
    app3('f', 'm', 1, 7, "Jane Peabody"),
    app4('f', 'm', 4, 9, "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'))
        cout << "Q accepted as gender" << endl;
    else
        cout << "Q rejected as gender" << endl;

    return 0;
}

void displayTwoProfiles(DateProfile profile1, DateProfile profile2) {
    cout << "Fit between "
         <<  profile1.getName() <<  " and "
         <<  profile2.getName() + ":\n";
    cout << "   " << profile1.fitValue(profile2) << endl;
}

// DateProfile Method Definitions
DateProfile::DateProfile() {
    setDefaults();
}

DateProfile::DateProfile(char theGender, char theSearchGender,
                         int theRomance, int theFinance, string theName) {
    setDefaults();  // in case of error in next call
    setAll(theGender, theSearchGender, theRomance, theFinance, theName);
}

// mutators -- do not revert to defaults if error
void DateProfile::setAll(char theGender, char theSearchGender,
                         int theRomance, int theFinance, string theName) {
    setGender(theGender);
    setSearchGender(theSearchGender);
    setRomance(theRomance);
    setFinance(theFinance);
    setName(theName);
}

void DateProfile::setDefaults() {
    gender = DEFAULT_GENDER;
    searchGender = DEFAULT_SEARCH_GENDER;
    finance = DEFAULT_FINANCE;
    romance = DEFAULT_ROMANCE;
    name = DEFAULT_NAME;
}

bool DateProfile::setGender(char theGender) {
    if (!validGender(theGender))
        return false;
    gender = toupper(theGender);
    return true;
```

```
}

bool DateProfile::setSearchGender(char theGender) {
    if (!validGender(theGender))
        return false;
    searchGender = toupper(theGender);
    return true;
}

bool DateProfile::setRomance(int theRomance) {
    if (theRomance < MIN_VAL || theRomance > MAX_VAL)
        return false;
    romance = theRomance;
    return true;
}

bool DateProfile::setFinance(int theFinance) {
    if (theFinance < MIN_VAL || theFinance > MAX_VAL)
        return false;
    finance = theFinance;
    return true;
}

bool DateProfile::setName(string theName) {
    if (theName.length() < MIN_NAME_LENGTH ||
        theName.length() > MAX_NAME_LENGTH)
        return false;
    name = theName;
    return true;
}

bool DateProfile::validGender(char theGender) {
    char newGender = tolower(theGender);

    return (newGender != 'm' && newGender != 'f');
}

double DateProfile::determineFinanceFit(DateProfile partner) {
    int diff = abs(finance - partner.finance);

    double fit = MAX_VAL - 1 - diff; // 9 is largest and 0 is the smallest
    fit = fit / (double)(MAX_VAL - 1);  // now goes from 0.0 to 1.0

    return fit;
}

double DateProfile::determineRomanceFit(DateProfile partner) {
    int diff = abs(romance - partner.romance);

    double fit = MAX_VAL - 1 - diff; // 9 is largest and 0 is the smallest
    fit = fit / (double)(MAX_VAL - 1);  // now goes from 0.0 to 1.0

    return fit;
```

```
}

double DateProfile::determineGenderFit(DateProfile partner) {
    if (searchGender != partner.gender || partner.searchGender != gender)
        return 0.0;
    return 1.0;
}

double DateProfile::fitValue(DateProfile partner) {
    // compute individual results for easy debugging and readability
    double genderFit = determineGenderFit(partner);
    double romanceFit = determineRomanceFit(partner);
    double financeFit = determineFinanceFit(partner);

    // form the return value for easy debugging
    double totalFit = genderFit * (romanceFit + financeFit) / 2.;

    return totalFit;
}

/* ------------------ Output of Above ------------------------------
 Fit between Joe Somebody and Joe Somebody:
 0
 Fit between Joe Somebody and Steve Nobody:
 0
 Fit between Joe Somebody and Jane Peabody:
 0.888889
 Fit between Joe Somebody and Helen Anybody:
 0.833333
 Fit between Steve Nobody and Joe Somebody:
 0
 Fit between Steve Nobody and Steve Nobody:
 0
 Fit between Steve Nobody and Jane Peabody:
 0.666667
 Fit between Steve Nobody and Helen Anybody:
 0.722222
 Fit between Jane Peabody and Joe Somebody:
 0.888889
 Fit between Jane Peabody and Steve Nobody:
 0.666667
 Fit between Jane Peabody and Jane Peabody:
 0
 Fit between Jane Peabody and Helen Anybody:
 0
 Fit between Helen Anybody and Joe Somebody:
 0.833333
 Fit between Helen Anybody and Steve Nobody:
 0.722222
 Fit between Helen Anybody and Jane Peabody:
 0
 Fit between Helen Anybody and Helen Anybody:
 0
```

```
Q rejected as gender
Press any key to continue . . .
------------------------------------------------------------------- */
```