Note: The class Profile has information about a user.   
 The class ProfileManager that follows has an undirected graph of Profile

The main creates an object of type ProfileManger

The GUI code is not shown below but an example of sample results of GUI interface is shown.

This file should give you an idea – it does not mean you must implement as described below.

public class Profile

{

private BufferedImage picture;

private String name;

private String status;

private ArrayList<Profile> friendProfiles;

/\*\* Constructor for an instance of a profile. \*/

public Profile()

{

} // end default constructor

/\*\* Returns the picture associated with the profile.

@return  The profile's picture. \*/

public BufferedImage getProfilePicture()

{

} // end getProfilePicture

/\*\* Sets the picture associated with the profile to the given picture.

@param newPicture  The new picture associated with the profile. \*/

public void setProfilePicture(BufferedImage newPicture)

{

} // end setProfilePicture

/\*\* Sets the name associated with the profile to the given name.

       @param firstName  The first name for the profile.

       @param lastName   The last name for the profile. \*/

   public void setName(String firstName, String lastName)

   {

   } // end setName

   /\*\* Returns the name associated with the profile.

       @return  The profile's name. \*/

   public String getName()

   {

   } // end getName

/\*\* Sets the current status of the profile to the given string.

@param stat  The new status for the profile. \*/

public void setStatus(String stat)

{

} // end setStatus

/\*\* Returns the status associated with the profile.

@return  The profile's status.\*/

public String getStatus()

{

} // end getStatus

/\*\* Returns a list of all the person's friendProfiles on the social network.

@return  The list of friendProfiles. \*/

public ArrayList<Profile> getFriends()

{

} // end getFriends

/\*\* Adds a friend to the profile's list of friendProfiles.

@param p  The profile to be added to the list. \*/

public void addFriend(Profile p)

{

} // end addFriend

/\*\* Removes a friend from the profile's list of friendProfiles.

@param p  The profile to be removed from the list.

       @return  True if the profile was removed. \*/

public boolean removeFriend(Profile p)

{

} // end removeFriend

public String toString()

{

} // end toString

/\*\* Displays the profile's information and friendProfiles. \*/

public void display()

{

} // end display

} // end Profile

import java.awt.image.\*;

import GraphPackage.\*;

import ADTPackage.\*;

/\*\*

An implementation of a profile manager on a simple social network.

@author Jesse Grabowski

@author Joseph Erickson

@version 5.0 \*/

public class ProfileManager

{

private UndirectedGraph<Profile> allProfiles;

/\*\* Constructor for an instance of a profile manager. \*/

public ProfileManager()

{

allProfiles = new UndirectedGraph<>();

} // end default constructor

/\*\* Adds a profile to the social network.

@param p  The profile to be added to the network. \*/

public void addProfile(Profile p)

{

allProfiles.addVertex(p);

} // end addProfile

/\*\* Creates a friendship between two users on the social network.

@param a  The first profile in the friendship.

@param b  The second profile in the friendship. \*/

public void createFriendship(Profile a, Profile b)

{

} // end createFriendship

/\*\* Displays each profile's information and friends. \*/

   public void display(Profile startPoint)

   {

} // end ProfileManager

public class Driver

{

public static void main(String[] args)

{

ProfileManager m = new ProfileManager();

      System.out.println("Creating profiles and the network.");

      Profile john = new Profile();

john.setName("John", "Doe");

john.setStatus("Married");

      m.addProfile(john);

} // end main

} // end Drive



