```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*This is the Student Main class, where the project and main methods will
be executed. In This class, a Student manager object is
  instantiated to access the methods to be executed within the Student
Manager class. The student and studentRegister arrays stores
  information regarding the student when reading in values from the
textfile and adding/deleting new students.
*/


/**
 *
 * @author b00882569
 */

public class StudentMain {

    //Student manager object is created
    static  StudentManager sm = new StudentManager();
    //Student and studentRegister array is made to store the student
details for reading/writing
    String [] student = new String[20];
    Student [] studentRegister = new Student [20];

    public static void main(String[] args){
        //Start method is executed, starting the enrolment system
         sm.start();
    }
}
```

```
import java.util.Scanner;
import java.io.File;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
/*
This is the student manager class where the validation for each field is
stored and where each method is stored. The validation in this class is
used for each field
required to successfully register a student to the course. Other methods
in this class consist of reading in from a textfile and passing into a
constructor, writing
out to a textfile and passing through a constructor, deleting a student
then updating the constructor, printing off course details based on
gender percentage, total students,
and total PT and FT students, and a exit method which will end the
program, writing any changes to the textfile if any. There are other
methods used to aid efficiency
but the ones listed are the main ones which were requested to be
achieved.
 */

/**
 *
 * @author
 */


public class StudentManager extends StudentMain
{
    //A private String array for storing the Name, DOB and Gender
    private String [] studentInformation = new String [3];

    //A private string array to store each field into it (Course Name,
Percentage Male, etc)
    private String [] courseDetails = new String [6];

    //A private string array which takes all of the values in the
courseDetails array and stores it as one sentence in the course array
    //This is used when writing out to the textfile when the application
is finished
    private String [] course = new String [1];

    public Scanner scan = new Scanner(System.in);

    //Method used when the user wishes to register a student. This
validates to ensure the users name reaches the criteria
    public String addName()
    {

        String name = "";

        //Variable to check if the name entered is an already existing
student
        boolean used = true;

        //Boolean used to ensure the user has entered the correct details
otherwise they will have to keep trying til it passes the validation
        boolean Name = false;
```

```
        while(Name == false)
        {
            System.out.println("\nPlease enter the students name:");
            name = scan.nextLine();
            name = name.toUpperCase();


            while(used == true)
            {
                    int i = 0;

                   //A for loop to go round the student array
                  for(i = 0; i<student.length; i++)
            {

                //Checks that there is no null students as it will result
in an error
                if(studentRegister[i] != null)
            {

                //An if statemen to check if te name entered is an
already existing user by using the getName and comparing it to the value
entered by the user
               if(name.equals(studentRegister[i].getName()))


                   {
                       //IF the name already exists, the user will stay
in the while loop until they enter a different name
                    System.out.println("There is a student with that name
reistered already, please choose another name");
                    System.out.println("\nPlease enter the students
name:");
                    name = scan.nextLine();
                    name = name.toUpperCase();
                   }
               else
               {
                   //If the name is not existing the user will then go
through the validation to ensure their name passes the criteria
                    used = false;
               }
            }
            }
            }

            //If statement to ensure the student name entered is not
null, <3 in length or >15 in length
            if(name.isEmpty()||name.length()<3||name.length()>15){
                System.out.println("\nPlease ensure you have entered a
valid name between 3-15 letters\n");
                Name = false;
            }
            else
            {
                boolean studentValid = true;
```

```
                //For loop to go round each character in the string
variable (which stores the users input), and checks none of the
characters are numbers or spaces
                for(int i = 0; i<name.length(); i++)
                {
                    char letter = name.charAt(i);

if(Character.isDigit(letter)||Character.isSpace(letter)){
                        i = name.length();
                        System.out.println("\nPlease ensure you have
entered a valid name with no numbers or spaces\n");
                        Name = false;
                        studentValid = false;
                    }

                    else if(Character.isLetter(letter))
                    {
                        studentValid = true;
                    }
                    else
                    {
                        i = name.length();
                        System.out.println("\nPlease ensure you have
entered a valid name with no numbers or spaces\n");
                        Name = false;
                        studentValid = false;
                    }
                }
                //If the name passes all validation, the studentValid
variable will be set to true and the user will resume onto the next field
required
                if(studentValid)
                    Name = true;
                }
            }

        //Retturns the name
        return name;
    }

    //Method used when the users wishes to register a student. This will
validate the DOB to ensure the user reaches the criteria needed
    public String addDate()
    {
        //Boolean value which is set once the DOB passes all validation
which takes them out of thew while loop, otherwise they will need to keep
entering the DOB til it passes validation
        boolean Date = true;
        String dob = " ";
        //Boolean used for the while loop to check that the DOB is not
over the length. ALL DOB has a Maximum length of 10, as its in the format
dd/mm/yyyy. Anything over does not meet the criteria
        boolean correctlength = false;


        while(Date)
        {
```

```
            //Takes the users input and stores it in the variable
            System.out.println("\nPlease Enter the DOB in the dd/mm/YYYY
format");
            dob = scan.nextLine();

            //While loop to make sure the user has not entered over the
character limit
            while(correctlength == false)
            {
                //If the length is over 10, it will be invalid and will
request the user to enter their name again
                 if(dob.length()>10)
                 {
                  System.out.println("\nThere is too many characters
entered. Please ensure the date is in the dd/mm/YYYY format");
                  dob = scan.nextLine();
                 }
                 else
                 {
                     correctlength = true;
                 }
            }



            try
            {
            //char array which will take each char of the string which
stores the DOB entered and ensures each char is correct as the 2nd char
HAS to be a forward slash, same as 5
            char[] dobArray = new char[dob.length()];

            //For loop to iterate through the DOB and check each
character matches the correct format
            for(int i = 0; i<dob.length(); i++){
            dobArray[i] = dob.charAt(i);
            }
                //Checks if it is a Digit as anything else is invalid
                if(Character.isDigit(dobArray[0])){
                    //As the highest date is 31st, any number over 4 is
invalid for the first char (eg a date starting with 4, 40)
                    if(Character.getNumericValue(dobArray[0]) <4){
                        //Checks the 2nd digit of the day is a number
                        if(Character.isDigit(dobArray[1])){
                            //Checks the 2nd character is a forward slash
                            if(dobArray[2] == '/'){
                                //Checks the first digit of the month is
a number
                                if(Character.isDigit(dobArray[3])){
                                    //As the highest month is 12, the
first digit cant be over 2

if(Character.getNumericValue(dobArray[3]) <2){
                                        //Checks the 2nd digit of the
month is a number

if(Character.isDigit(dobArray[4])){
```

```
                                                //As the highest number for
the 2nd value of month is 9, anything bigger would be invalid (if this
was not validated a month of 310 couldve been allowed)

if(Character.getNumericValue(dobArray[4]) < 10){
                                                    //Checks char is a
forward slash
                                                    if(dobArray[5] == '/'){
                                                        //Checks it is a
digit

if(Character.isDigit(dobArray[6])){
                                                                //Checks that the
year can't be over 3000 and below 0000 as it checks the first digit of
the year

if(Character.getNumericValue(dobArray[6]) >0 &&
Character.getNumericValue(dobArray[6])<3){
                                                                    //Checks if
it is a digit

if(Character.isDigit(dobArray[7])){
                                                                        //Checks
if it is a digit

if(Character.isDigit(dobArray[7])){

//Checks if it is a digit

if(Character.isDigit(dobArray[8])){

//Checks if it is a digit

if(Character.isDigit(dobArray[9])){

//If it passes all validation then the date is valid and the boolean is
set to false, exiting the while loop

Date = false;


//Checks if the first digit of the day is 0, then the 2nd must be over 0
as they cant have the day as 00, but ensures its 01 - 09

//If it is invalid then the boolean is set to true, requiring the user to
enter the while loop again

if(Character.getNumericValue(dobArray[0]) == 0 &&
Character.getNumericValue(dobArray[1]) <1){

Date = true;

}

//Checks if the first digit of the montn is 0, then the 2nd must be over
0 as they cant have the month as 00, but ensures its 01 - 09
```

```
//If it is invalid then the boolean is set to true, requiring the user to
enter the while loop again

if(Character.getNumericValue(dobArray[3]) == 0 &&
Character.getNumericValue(dobArray[4])<1){

Date = true;

}
                                                                                    }
                                                                                }
                                                                            }
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }

                            }
                            catch(Exception ex)
                            {
                                System.out.println("Error Occurred. Date entered
incorrectly");
                            }

                        }
                            return dob;
    }

    //MEthod used to check what study mode the user was, Part Time or
Full Time
    public String addMode()
    {

        String mode = "";

        boolean Mode = false;
        while(Mode == false)
        {
            System.out.println("Please enter the students study mode
(FT/PT):");
            mode = scan.nextLine();
            //Comverts user entry to Upper case as its easier to validate
            mode = mode.toUpperCase();

            //If the users enters FT or PT, then it is accepted as a
valid input, anything else is invalid
            if(mode.equals("FT")){
                Mode = true;
            }
```

```
            else if (mode.equals("PT")){
                Mode = true;
            }

            //If the entry isnt PT or FT they need to retry
            else{
                System.out.println("Please only enter 'FT' or 'PT'\n");
                Mode = false;
            }
        }


        return mode;
    }
    public char addGender(){

        //Boolean used to check if the value entered is valid or not,
used for a while loop incase the entry is invalid
        boolean check = false;

        //String to store the gender then convert to a char as it saves
additional code to validate
        String genderS;
        char gender = ' ';

        //While the entry is invalid or the user is yet to entry the
value for the first time
        while(check == false)
        {
            System.out.println("\nPlease Enter a Gender (M/F): ");
            genderS = scan.nextLine();

            //Check the Gender is not null by checking its more than 0
characters long
            if(genderS.length()>0)
            {
            //Store the string as a char
            gender = genderS.charAt(0);
            gender = Character.toLowerCase(gender);
            }
            //Stores gender as a String then it is converted to a char
and then to lower case

            //If the char is over the length or it is not m or f, then it
is invalid
            if(genderS.length() >1)
            {
                check = false;
            }
            else{
                if(gender == 'm' || gender == 'f'){
                check = true;
                }
            }
        }

        return gender;
    }
```

```
    public String addYear(){

        //String to store the year value
        String year = "";

        //While loop to constantly iterate through until the user meets
the criteria for the year
        boolean Year = false;
        while(Year == false)
        {
            //Stores Year
            System.out.println("Please enter what year the student is in
(1,2,3 or 4):");
            year = scan.nextLine();


            //If the Year entered is 1, 2, 3, 4 then it is valid

if(year.equals("1")||year.equals("2")||year.equals("3")||year.equals("4")
){
                Year = true;
            }

            //If not it is invalid, and needs to be entered again
            else{
                System.out.println("Please only enter numbers 1-4\n");
                Year = false;
            }
        }
        return year;
    }
    public int addModules(){

        //Int to store the modules. As calculations are required
regarding the modules, int is an appropriate data type
        int modules = 0;

        //While loop to ensure the user meets the criteria for the module
        boolean Module = false;
        while(Module == false)
        {
            //Stores module
            System.out.println("Please enter the number of modules that
the student is taking:");
            String module = scan.nextLine();
            boolean number = false;

            //Checks that the module is a digit
            for(int i = 0; i< module.length(); i++){
                char letter = module.charAt(i);


                if(Character.isDigit(letter)){
                    number = true;
                }
            }
```

```
            //The number boolean is used to check when its valid or not.
When it is false, the Module input is incorrect, requiring the user to
enter it again
            if(number){
                modules = Integer.parseInt(module);
            }

            //Checks the Module is between 1-6
            if(modules >= 1 && modules <= 6){
                Module = true;
            }
            //If it isnt, they need to go through the input again
            else{
                System.out.println("Please only enter numbers 1-6\n");
                Module = false;
            }

        }
        return modules;
    }

    //Method to write students information
    public void writeStudentInformation()
    {
        //Data types to store the students information into using the
methods for each value
        String name = " ";
        String dob = " ";
        char gender;
        String studyMode = " ";
        String year = " ";
        int modules;

        //Creating a new studentInformation array. so when the method is
executed, any previous data stored in the array will be reset
        studentInformation = new String[7];

        //Check to see if the maximum number of Students has already been
reached
        int numStudents = 0;

        //Check each line in the student array which stores students
information
        for(String checkStudent : student)
        {
            try
            {
                //This checks if there is a student within the line
                if(checkStudent != null)
                {
                    //IF there is a student entry in the line, add one to
the total counter
                    numStudents = numStudents +1;
                }
            }
            catch(Exception ex)
            {
```

```
            }
        }

        //This checks if the file has exceeded the maximum number of
students
        if(numStudents == 20)
        {
            System.out.println("Sorry, the enrolment has already reached
the maximum amount of students. To register please delete a student
before continuing.");

        }
        else
        {
          //local variables which will execute each method for each field
of information require, and save the outcome
          name = addName();
          name = name.toUpperCase();

        //Execute the method to request the user to insert their DOB,
validaitng it
          dob = addDate();
          dob = dob.toUpperCase();

        //Execute the method to request the user to insert their Gender,
validaitng it
          gender = addGender();

        //Execute the method to request the user to enter their study
mode
          studyMode = addMode();

        //Execute the method to request the user to enter what year they
are currently studying in
          year = addYear();

        //Execute the method to request the user to enter how many
modules they are currently partaking in
          modules = addModules();

      //The student information array stores each value of a student such
as name and DOB in it
      //Once all values are stored, another array called student,
combines all values in the studentInformation array into one value and
stores it, which is used when writing to a textfileS
        studentInformation[0] = name;
        studentInformation[1] = dob;
        //As studentInformation is a string array, the gender needs to be
converted to string
        studentInformation[2] = Character.toString(gender);
        studentInformation[3] = studyMode;
        studentInformation[4] = year;
        //As studentInformation is a string array, the modules needs to
be converted to an int
        studentInformation[5] = Integer.toString(modules);
```

```
        //Creating a new object for the new student, and assigning the
information entered through the constructor
        studentRegister[numStudents] = new Student(
studentInformation[0],studentInformation[1],gender,
studentInformation[3], studentInformation[4],modules);

        //double to store the tuition fee
        double finalFee =  0;

        //Checks if the student is a Full time student, using gets from
the constructor
         if(studentRegister[numStudents].getMode().equals("FT"))
         {
            //Passes the values through the class and uses the
computingFee method to calculate it
            Student fullTime = new
FT(studentInformation[0],studentInformation[1],gender,
studentInformation[3], studentInformation[4],modules);
            finalFee =  fullTime.computingFee();
            //Sets the value in the class
            studentRegister[numStudents].setFees(finalFee);


         }
         else
         {
             //Checks if the student is a Part time student using gets
from the constructor
             if(studentRegister[numStudents].getMode().equals("PT"))
             {
                //Passes the values through the class and uses the
computingFee method to calculate it
                Student partTime = new
PT(studentInformation[0],studentInformation[1],gender,
studentInformation[3], studentInformation[4],modules);
                finalFee =  partTime.computingFee();
                studentRegister[numStudents].setFees(finalFee);
             }
         }

         //Prints out fee
        System.out.println("Tuition Fee: " +
studentRegister[numStudents].getFees());

        //Stores the fee in the array, as the fee needs to be printed out
when the application is exited
        studentInformation[6] = Double.toString(finalFee);

        //Appends each field for the new student into one string, and
stores the string in the student array, as the student array is used for
writing the data out to the text file
        student[numStudents] = studentInformation[0] + "," +
studentInformation[1] + "," + studentInformation[2] + "," +
studentInformation[3] + "," + studentInformation[4] + "," +
studentInformation[5] + ","  + studentInformation[6];

        }
    }
```

```
    //Method to delete a students information regarding their name
    public void deleteStudentInformation()
    {
        //String to store the user they wish to delete
        String deleteUser;

        //int to store the index if the name is found
        int iValue=0;

        //Boolean to check if the student they are wanting to delete
exists or not
        //If it is false then the student doesn't exist
        boolean found = false;

        System.out.println("Please Enter the Students name you wish to
delete");
        deleteUser = scan.nextLine();

        //This converts the String to all Upper case as all NAME values
stored in the textfile as uppercase
        deleteUser = deleteUser.toUpperCase();




        //For loop to go through all of the students
            for(int i = 0; i<student.length; i++)
            {

                //Checks that there is no null students as it will result
in an error
                if(studentRegister[i] != null)
                 {

                //An if statement to check if the name entered is an
already existing user by using the getName and comparing it to the value
entered by the user
                     if(deleteUser.equals(studentRegister[i].getName()))

                      {
                          //If the user is found, the index of that user
in the array is stored and the boolean value is true, meaning the user
passes is found
                           iValue = i;
                          found = true;
                      }

                  }
            }

            //If the name entered does not match any existing records, it
means it was not found
            if(found == false)
              {
                System.out.println("The name you are looking for does not
exist");
              }
```

```
        //A String array to store the data of the other users who were
not requested to be deleted. The -1 is employed to prevent the deleted
user to get added in as if its deleted the size will be 1 less
        String otherUsers [] = new String[student.length-1];

        //for loop to iterate through the student array
        for(int ii = 0, x = 0; ii<student.length; ii++)
        {
            //If the index is not the same index which the user to be
deleted is stored in, store them values in the otherUsers array from the
student array, the student array will then become the otherUsers array
            if(ii != iValue)
            {
                //Add the values of the users NOT to be deleted in the
otherUsers array. The student array will then replicate the values in the
otherUsers array
                otherUsers[x++] = student[ii];
            }
        }

        //Create a new student array to update the students as one may
have been deleted, and set the max number of students to 20
        student = new String [20];

        //For loop to go through the student array and adds all the users
which was not to be deleted into the student array, updating it
        for(int i = 0; i < student.length; i++)
        {
            try{
                //Populate the new student array with the values of the
otherUsers array (which has the values of the students not deleted
                student[i] = otherUsers[i];

            //As the users are getting deleted, I am reusing the same
code used for setting the values (when read in), so the deleted user no
longer remains in the system
            //This is cause a students index may change once being
deleted. If the index of the user to be deleted was 5, the the user in
index 6 is now index 5
            String name = student[i].split(",")[0];
            String dob = student[i].split(",")[1];
            String genderS = student[i].split(",")[2];
            char gender = genderS.charAt(0);
            String mode = student[i].split(",")[3];
            String year = student[i].split(",")[4];
            String modulesS = student[i].split(",")[5];
            int modules = Integer.parseInt(modulesS);

            //Sets the new values
            studentRegister[i] = new Student(name, dob, gender, mode,
year, modules);

                }
            catch (Exception ex)
                {

                }
```

```
        }
    }

     //Method to search for the users details
     public void searchStudentDetails()
     {
         //String variable to store the user wishes to search for
         String user;

         //Asks the user for the students who's details they wish to
access
         System.out.println("Please Enter the Student you's details you
wish to see: ");
         user = scan.nextLine();
         user = user.toUpperCase();

         //Boolean which changes if the name is an existing user
         boolean found = false;
         //For loop to iterate through each detail in the student array
and output the values of that student
         for(int i = 0; i <student.length; i++)
         {
             if(studentRegister[i] != null)
             {
                 //If the name entered matches any existing records, all
values to be outputted
                 if(user.equals(studentRegister[i].getName()))
                     {
                         //Prints the values for the student they wish to
see
                          found = true;
                          System.out.println(">||\nName: " +
studentRegister[i].getName());
                          System.out.println("||\nDate of Birth: " +
studentRegister[i].getDoB());
                          System.out.println("||\nGender: " +
studentRegister[i].getGender());
                          System.out.println("||\nYear of Study : " +
studentRegister[i].getYear());
                          System.out.println("||\nModule: " +
studentRegister[i].getModule());
                          System.out.println("||\nStudy Mode: " +
studentRegister[i].getMode());
                          System.out.println("||\nTuition Fee: " +
studentRegister[i].getFees());

                      }
             }
         }
         //If the user entered is not a student, message is shown
         if(found == false)
         {
             System.out.println("The student you are looking for does not
exist");
         }

     }
```

```
    //Method executed after each method used in the switch statement (Add
user, Delete User, Print Report), to make the interface look better
    public void moveOn()
    {
        System.out.println("\nType any character to resume: \n");
        String pointlessInfo = scan.nextLine();
        System.out.println("----------------------------------------------
-------------------------------------");
    }

    //Method to display the options the user has to choose from and then
executes the method in correlation to the users decision
     public void menuOptions()
    {


      boolean finished = false;

      System.out.println("\nWelcome to the Student enrolment
hub!\\n\\nPlease Select one of the following options below!");
      while(finished == false)
        {
            //Prints the options the user has then stores them into a
string variable
            System.out.println("\n1:Add a Student\n\n2:Delete a
Student\n\n3:Print a Course Report\n\n4:Search for Student
Details\n\n5:Exit\n");
            String input = " ";
            System.out.println("Please Select a Menu Option from
Above:");
            input = scan.nextLine();

            //IF the user does not enter a suitable option, it will
result in them having to try again
            while(!"1".equals(input) && !"2".equals(input) &&
!"3".equals(input) && !"4".equals(input) && !"5".equals(input))
            {
                System.out.println("Please Select a Suitable Menu
Option:");
                input = scan.nextLine();
            }

            switch(input)
            {
                //Method 1 is the add user method, which will register a
student to a course
                 case "1":
                     sm.writeStudentInformation();
                     moveOn();
                  break;

                  //Method 2 deletes a student from the course
                  case "2":
                    deleteStudentInformation();
                    moveOn();
                  break;

                  //Method 3 prints a course report
```

```
                   case "3":
                displayReport();
                   break;

                   //Method 4 searches for a user and prints their
details off
                  case "4":
                searchStudentDetails();
                moveOn();
                   break;

                   //Method 5 will exit the application and write all new
datas to a textfile and remove any deleted data from it
                  case "5":
                  exit();
                  break;
             }



         }
       }

    public void start()
    {
        Scanner sca = new Scanner(System.in);
        sca = null;
        int index = 0;

        //Read the Student details textfile and input the values into an
array which can be accessed by various methods to delete, search and add
        try
        {
            //This associates the studentFile variable with the
StudentDetails textfile
            //The textfile will then be read in using the Scanner object
            File studentFile = new File("StudentDetails.txt");
            sca = new Scanner(studentFile);

        }
        //If the file if not found, an exception will be thrown to
prevent an error from occurring
        catch(FileNotFoundException ex)
        {
            System.out.println("File is not found");
            System.exit(1);
        }

        //This will iterate through the textfile and add the values into
the student array using the index variable to find each line
        while(sca.hasNextLine())
        {
            String data = sca.nextLine();

            //This will take the current index of the file and add the
value of that index into the array
            student[index] = data;
```

```
            //This is creating local variables and passing them through
the Student class, setting the values
            //I am using a , as an indicator where to split between each
values
            String name = student[index].split(",")[0];
            String dob = student[index].split(",")[1];
            String genderS = student[index].split(",")[2];
            char gender = genderS.charAt(0);
            String mode = student[index].split(",")[3];
            String year = student[index].split(",")[4];
            String modulesS = student[index].split(",")[5];
            int modules = Integer.parseInt(modulesS);
            String fee =  student[index].split(",")[6];
            double finalFee = Double.parseDouble(fee);

            //Setting each value and computing the fee and setting it
            studentRegister[index] = new Student(name, dob, gender, mode,
year, modules);
            studentRegister[index].setFees(finalFee);

            //After the value of that index is added, the index will
increment and add the next value into the array
            index++;
        }

        //Closing the file when finished to prevent errors from occuring
        sca.close();

        //Load the available menu options
        menuOptions();
    }

    public void displayReport()
   {

       //Variables used to calculate the Percentage of males and females
       float percF = 0;
       float percM = 0;

System.out.println("//////////////////////////////////////////////////////
////////////////////////////////////");
       System.out.println("\n Below you will find the information
regarding statistics on Male and Female percentage, total number of
Students and the Lecturer and Course");

System.out.println("//////////////////////////////////////////////////////
////////////////////////////////////");

      //To calculate the total male and Female I divided the total males
and Females from the total students then multiplied by 100
      //The totalF and totalM methods are used to calculate the total of
each gender, that value is then divided by the total number of Students
(a method)
       percF = (totalF() / totalStudents())*100;
       percM = (totalM() / totalStudents())*100;

     System.out.println("\n>The Course is Computing");
```

```
     System.out.println("\n>The Total Number of Part Time Students is: "
+ totalPT());
     System.out.println("\n>The Total Number of Full Time Students is: "
+ totalFT());
     System.out.println("\n>The Percentage of Students Who are Males are:
" + percM + "%");
     System.out.println("\n>The Percentage of Students Who are Females
are: " + percF + "%");

     //This writes the Course Details data to the array which will then
be transferred to the course array, which wil be used to write the values
to the textfile

       moveOn();
   }


//METHODS USED FOR REPORT TO SIMPLIFY REUSING CODE FOR BOTH displayReport
METHOD AND updateCourseArray METHOD
   ////////////////////////////////////////////////
    //Method to calculate the total number of students
  public float totalStudents()
  {
     float numStudents =0;

     //For loop to iterate through each student and if there is a value
it will add one to the numStudents variable
      for(int i = 0; i <student.length; i++)
       {
           try
           {
               //This checks that the field is not null and if it isnt
the total number of students will increment
               if(student[i] != null)
               {
                   //Add one to the number of students in the course for
each line of data in the array
                  numStudents = numStudents +1;
               }
           }
           catch(Exception ex)
           {
           }
  }
     return numStudents;

  }

  //Method to calculate the total number of Part Time students
  public float totalPT()
  {
       float totalPT =0;

       //For loop to go through the student array and if their study mode
is PT, it will add 1 to the totalPT variable each time
        for(int i = 0; i <student.length; i++)
        {
            if(studentRegister[i] != null)
```

```
            {
              if(studentRegister[i].getMode().equals("PT"))
                 {
                   totalPT++;
                 }
             }
         }
     return totalPT;
  }

  //Method to calculate the total number of Full Time students
    public float totalFT()
  {
       float totalFT =0;

        //For loop to go through the student array and if their study
mode is FT, it will add 1 to the totalFT variable each time
        for(int i = 0; i <student.length; i++)
        {
            if(studentRegister[i] != null)
            {
              if(studentRegister[i].getMode().equals("FT"))
                 {
                   totalFT++;
                 }
             }
        }
     return totalFT;
  }

    //Method to calculate the total number of Male students
      public float totalM()
  {
       float totalM =0;
    //For loop to go through the student array and if they are a Male, it
will add 1 to the totalM variable each time
       for(int i = 0; i <student.length; i++)
       {
           try
           {
               //This checks that the field is not null and if it isnt
the total number of students will increment
                if(student[i].contains("m"))
                   {
                       totalM++;
                   }
           }
           catch(Exception ex)
           {
           }
  }
     return totalM;
  }

      //Method to calculate the total number of Male students
    public float totalF()
  {
       float totalF =0;
```

```
     //For loop to go through the student array and if they are a Female,
it will add 1 to the totalF variable each time
      for(int i = 0; i <student.length; i++)
       {
           try
           {
               //This checks that the field is not null and if it isnt
the total number of students will increment
                if(student[i].contains("f"))
                   {
                       totalF++;
                   }
           }
           catch(Exception ex)
           {
           }
  }
     return totalF;
  }

   ////////////////////////////////////////////////
   //IF THE USER DOES NOT DISPLAY THE REPORT YET ADD'S or DELETE'S A
USER(s), THIS WILL BE CALLED TO UPDATE THE STATISTICS BEFORE WRITING TO
TEXTFILE
   public void updateCourseArray()
   {
     float percF = 0;
     float percM = 0;

      //To calculate the total male and Female I divided the total males
and Females from the total students then multiplied by 100
       percF = (totalF() / totalStudents())*100;
       percM = (totalM() / totalStudents())*100;

       //This writes the Course Details data to the array which will then
be transferred to the course array, which wil be used to write the values
to the textfile
       courseDetails[0] = "Computing";
       //COnverting all Floats to string to allow the values to be added
to a string array
       courseDetails[1] = Float.toString(totalStudents());
       courseDetails[2] = Float.toString(totalPT());
       courseDetails[3] = Float.toString(totalFT());
       courseDetails[4] = Float.toString(percM);
       courseDetails[5] = Float.toString(percF);


       //This writes all the data from the coursedetails array to the
textfile by adding it to the course array (course array is used for
writing out)
       course[0] = courseDetails[0] + "," +  courseDetails[1] + "," +
courseDetails[2] + "," +  courseDetails[3] + "," + courseDetails[4] +
"%," +  courseDetails[5] + "%";

   }


    public void exit(){
```

```
    //Calls the update Course Array method as the user may delete a user
and not view the report, so this will calculate the changes and write it
to the course details textfile
    updateCourseArray();

    PrintWriter writetoTextfile = null;
       //save student details to StudentDetails.txt
        try {
        writetoTextfile = new PrintWriter("StudentDetails.txt");
            }
        catch (FileNotFoundException ex)
        {
            System.out.println(ex.getMessage());
        }
        //write each record on array to text file using enhanced for loop
        for(String studentLine : student){
            //try to test code for errors
            try
            {
                //If the student is not null,it will write that line to
the textfile
                if (studentLine != null)
                {
                 writetoTextfile.println(studentLine);
                }
            }
            //catch to stop errors
            catch (Exception ex)
            {
            }
        }
        // Close the files.
        writetoTextfile.close();

////////////////////////////////////////////////////////////////////////
////////////////////////////
        try
        {
        //This clears the CouseDetails textfile so various reports from
previous methods of execution are not saved
        PrintWriter clearCourse = new PrintWriter("CourseDetails.txt");
        clearCourse.print("");
        clearCourse.close();
        }
        catch(FileNotFoundException ex)
        {
        }
          //Creating a PrintWriter variable and assigning it to the
CourseDetails textfile
          PrintWriter writeToCourse = null;

          try {
        writeToCourse = new PrintWriter("CourseDetails.txt");
             }
        catch (FileNotFoundException ex)
        {
            System.out.println(ex.getMessage());
```

```
        }
        //This iterates through the course array and writes out each line
in it
        for(String courseLine : course){
            //try to test code for errors
            try
            {
                //If there is no null line, the value within that line
will be written out
                if (courseLine != null)
                {
                 writeToCourse.println(courseLine);
                }
            }
            //catch to stop errors
            catch (Exception ex)
            {
            }
        }
        // Close the files.
        writeToCourse.close();

        //exit the program
        System.out.println("\nThank you for using the Student Enrolment
System");
        System.exit(0);
    }//end of exit()
  }//end of exit()
```

```
/*
//RYANS
This is the Student Class. In this class the main constructor is made
where the new student values are passed through and assigned using sets
and constructors,
and in addition the student details can be displayed using the gets.
 */

/**
 *
 * @author b00884706
 */

import java.util.Scanner;
import java.util.Arrays;

public class Student {
    private String name;
    private String dob;
    private char gender;
    private String mode;
    private String year;
    private int modules;
    private double fees;

    public Student(String newName, String newDob, char newGender, String
newMode, String newYear, int newModules){

        name = newName;
        dob = newDob;
        gender = newGender;
        mode = newMode;
        year = newYear;
        modules = newModules;

    }

    public String getName(){
        return name;
    }
    public String getDoB(){
        return dob;
    }
    public char getGender(){
        return gender;
    }
    public String getMode(){
        return mode;
    }
    public String getYear(){
        return year;
    }
    public int getModule(){
        return modules;
    }
    public void setFees(double f)
    {
        this.fees = f;
```

```
    }
    public double getFees(){
        return fees;
    }

    public double computingFee(){
         return getFees();
    }

}
```

```
/*
 This is the Part time class which is a child class of the Student class,
allowing it to access the variables within that class. This class
 is used whenever a Part Time staff is registered as the fees per student
differ depending on their study mode.
 */

/**
 *
 * @author b00884706
 */
public class PT extends Student {

    private int moduleRate = 750;


    public PT(String newName, String newDob, char newGender, String
newYear, String newMode, int newModules){

        super(newName, newDob, newGender, newMode, newYear, newModules);
    }

   // public void setCost(int y){
   //     int cost = y*750;
   //     setFees(cost);
   // }

      @Override
    public double computingFee(){
        setFees(getModule()*moduleRate);
       //Just a random number to prevent a no return error
         return getFees();
    }
}
```

```
/*
 This is the Full Time class which is a child class of the Student Class.
This class is used whenever a new Full Time
 student is registred as the fee cost differentiates regading their Sutdy
mode.
 */

/**
 *
 * @author b00884706
 */
public class FT extends Student {

   // private double fee;
    public FT(String newName, String newDob, char newGender, String
newMode, String newYear, int newModules)
    {

        super(newName, newDob, newGender, newMode, newYear, newModules);


    }

   @Override
    public double computingFee()
    {
        if(!"3".equals(getYear()))
        {
           setFees(5000);
        }
        else
        {
           setFees(2500);
        }
        return getFees();
    }
}
```