Design, write in Java, test and document (at least) two classes – **a Student class** and a **client program**, as follows:

Write a Java class called **Student** which can be used to represent the details of a Student together with

some associated operations.

The Student class will have the following information:

(a) Title of the student (eg, Mr, Miss, Ms, Mrs etc)

(b) A first name (given name)

(c) A last name (family name/surname)

(d) Student number (ID) – an integer number (of type **long**)

(e) A date of birth (in day/month/year format – three ints)

(f) There are two assignments, each marked out of a maximum of 100 marks and equally weighted.

The marks for each assignment are recorded separately.

(g) There is weekly practical work. The marks for this component are recorded as a total mark

obtained (out of a maximum of 10 marks) for all practical work demonstrated during the semester.

(h) There is one final examination that is marked out of a maximum of 100 marks and recorded

separately.

(i) An overall mark (to be calculated within the program)

(j) A final grade, which is a string (to be calculated within the program)

The final grade, in this version of the software, is to be awarded on the basis of an overall mark, which is a number in the range 0 to 100 and is obtained by calculating the weighted average of the student's performance in the assessment components. The criteria for calculating the weighted average is as defined below:

**The two assignments together count for a total of 40% (20% each) of the final grade, the**

**practical work is worth 10%, and the final exam is worth 50% of the final grade.**

A grade is to be awarded as follows: An overall mark of 80 or higher is an HD, an overall mark of 70

or higher (but less than 80) is a D, an overall mark of 60 or higher (but less than 70) is a C, an overall

mark of 50 or higher (but less than 60) is a P, and an overall mark below 50 is an N.

The student class will have at least the following constructors and methods:

(i) two constructors - one without any parameters (the **default constructor**), and one with

parameters to give initial values to instance variables.

(ii) a reasonable number of set and get methods.

(iii) input and output methods.

(iv) methods to compute the final overall mark and the final grade. These two methods will be

void methods that set the appropriate instance variables. Remember one method can call

another method. If you prefer, you can define a single method that sets both the overall mark

and the final string grade, but if you do this, use a helper method.

(v) an equals method which compares two student objects and returns true if they have the same

student names, the same date of birth and the same student number, otherwise it returns false

**Your client class (program)** will provide the user with a menu to perform the following operations:

1. Quit (exit the program)

2. Add (to the array) all information about a student (except the overall mark and the grade) by

reading it from the keyboard and determine the student’s grade

3. Output from the array the details (all information including the overall mark and the grade) of

all students currently held in the array

4. Compute and output the average overall mark for students currently held in the array

5. Determine and display how many students obtained an overall mark equal to or above the

average overall mark and how many obtained an overall mark below the average overall mark

6. Display the distribution of grades (i.e., the number of HDs, Ds etc) awarded

7. Given a student number (ID), view all details of the student with that number. If the student is

not found in the array, an appropriate error message is to be displayed

8. Given a student’s name (both surname and given name – ignoring case), view all details of that

student. If the student is not found in the array, an appropriate error message is to be displayed

9. Find students with the highest overall mark and the second highest overall mark in the array

and display their names and overall marks

10. Sort the array of student objects into ascending order of students’ numbers (IDs), and output

the sorted array

11. Sort the array of student objects into ascending (alphabetical/dictionary) order of students’

surnames using a different sorting algorithm to the one used in (10) above, and output the

sorted array.

**Note that the program will loop around until the user selects the first option (Quit).**

**Set up a student array of N student objects, and test it with N = 6 (at least). You have to provide**

**the test data with your program. You can store your test data in a file or hard coded in your**

**program.**