Assignment 2

Due date: Week 11 Thursday (3 October 2019) 11:45 PM AEST

Weighting: 30%

Assignment task

Write a java console application that allows the user to read, validate, store, display, sort and search data (name, age, taxable income, tax, tax group) for N taxpayers. N should be declared as a constant and it should be equal to the largest digit of your student id number (e.g. if your ID number is S1267222 then N should be equal to 7 and you can declare it as final int N=7;). The name, age and taxable income must be read from the keyboard and tax and tax group must be calculated using Table 1.

Table 1. Income Tax and Tax Group for Taxable Income

Taxable Income	**Income Tax**	**Tax Group**
$0 – $18200	$0	Group 1
$18201– $37000	$0 plus 19c for each $1 over $18200	Group 2
$37001 - $87000	$3572 plus 32.5c for each $1 over $37000	Group 3
$87001 - $180000	$19822 plus 37c for each $1 over $87000	Group 4
$180001 and over	$54097 plus 45c for each $1 over $180000	Group 5

The name, age, taxable income, tax and tax group must be stored in an Array or ArrayList (index 0 for taxpayer 1 and index N-1 for taxpayer N). The minimum and maximum ages which can be stored are 18 and 64. The minimum and maximum taxable incomes which can be stored are $1 and $999000. A validation for minimum and maximum values must be done during the reading of age and taxable income.

Your application should display and execute a menu with the following options. A switch statement must be used to execute the following menu options.

1. Read, validate and store data for N taxpayers
2. Calculate and store tax and tax group for all taxpayers
3. Display all taxpayers
4. Display the name and age of all taxpayers from tax group 1
5. Search a taxpayer by age
6. Sort and display taxpayers
7. Exit from the application

1. Read, validate and store data for N taxpayers
This option reads name, age and taxable income for N taxpayers from the keyboard and stores them in an Array or ArrayList. If the age is less than 18 and greater than 64 then an appropriate message should be displayed and the user should be asked to enter the age again. Similarly if the taxable income is less than $1 and greater than $999000 then an appropriate message should be displayed and the user should be asked to enter the taxable income again.

2. Calculate and store tax and group number for all taxpayers
This option calculates the tax and group number based on Table 1 for each taxpayer and then stores tax and group number in Array or ArrayList.

3. Display all taxpayers
This option displays the data (name, age, taxable income, tax, group number) for all taxpayers.

4. Display the name and age of all taxpayers from tax group 1
This option displays the name and age of all taxpayers from tax group 1. If there is no taxpayer in group 1 then it displays an appropriate message.

5. Search a taxpayer by age
This option asks user to enter the taxpayer's age and searches for it. If the age is found then it displays an appropriate message with taxpayer details (name, age, taxable income, tax, group number). If age is not found then it displays an appropriate message "taxpayer with given age is not found". If there is more than one taxpayer with the given age then it displays all of them. **A built-in search algorithm is not allowed in this assignment**.

6. Sort and display taxpayers
This option sorts (by name) all taxpayers stored in Array or ArrayList in descending order and displays all sorted taxpayers (name, age, taxable income, tax, group number). You can use any sorting algorithm. **A built-in sort algorithm for sorting is not allowed in this assignment**.

7. Exit from the application
The application should display an appropriate message with student name and then exit from the application.

The application should work in a loop to enable the user to read, validate and store data for all taxpayers, calculate and store tax and tax group for all taxpayers, display all taxpayers, display the name and age of all taxpayers from tax group 1, search a taxpayer by age, sort and display taxpayers and exit from the application

Program design

You may use any design that meets the specification. However, a good design will adhere to the following guidelines:

- be logically correct
- be easy to read and maintain
- be well-designed
- use a UML activity diagram
- use appropriate classes, methods and fields

Your design MUST use the classes as listed below.

```
    public class Taxpayer
    {
                //fields to store name, age, taxable income, tax, group number
                //relevant methods including set and get methods
    }
    public class TaxpayerTest
    {
        public static void main(String[] args)
        {
                //menu options using switch statement
                //data reading, Taxpayer object creation, data storing in array/arrayList, etc.
        }
        //other static methods
    }
```

You may add/use methods, parameters, fields/variables, constants, etc. which you need to complete the application.

Testing

Testing is important. You should test your application using many different types of test cases before you submit it for marking.

What to submit

You should submit online the following files.

- Taxpayer.java (this file contains java code for class Taxpayer)
- TaxpayerTest.java (this file contains java code for class TaxpayerTest).
- Report.docx (this file contains a brief report that includes student name, student ID, unit name, unit code, UML activity diagram for menu option 2 (calculate and store tax and group number for all taxpayers) and test results (screenshots for different test cases with results to show that your application is correctly working)).

Warning: You must submit your own work and correct files. You should not take any help from anyone and you should not show your code to anyone. If you have any problem, you should talk to your tutor or lecturer or unit coordinator.

Assessment 2 marking criteria

	Total Marks – 30	**Marks Allocated**
1	**Design, logic and testing – 14**	
	Code readability – appropriate use of comments, indentation and naming conventions	3
	Declaring and using fields/variables and constants	2
	Creating/declaring and using objects, methods and classes	3
	Overall design and logic (structure, efficiency)	2
	Validity of testing as evidenced by submitted test results	2
	UML activity diagram for menu option 2	2
2	**Compilation and execution - 14**	
	(0 - if application doesn't run)	
	Welcome and exit messages	1
	Input data validation and message	1
	Read and store all taxpayers	1
	Calculate and store tax and group number for all taxpayers	2
	Display all taxpayers	1
	Display the name and age of all taxpayers from tax group 1	2
	Search for a taxpayer by age	2
	Sort and display taxpayers	2
	Overall program execution (user friendly, input/output and display)	2
3	**Report – 2**	
	Presentation (fonts, spaces, information, language)	2
4	**Penalty**	
	Penalty for submission of incorrect files including names and formats is 2 marks	
	Penalty for late submission is 5% mark / day or part of a day	