**GHANA COMMUNICATION TECHNOLOGY UNIVERSITY**



**FACULTY OF COMPUTING AND INFORMATION SYSTEM**

**PROGRAM: BIT L300 ( TOP UP )**

**COURSE: JAVA PROGRAMMING**

**COURSE CODE: IT 303**

**NAME: Aaron Nii Annan Tetteh**

**INDEX NUMBER: 1725354956**

**QUESTIONS**

Q2

1. Create a class name **Emolument,** to represent an emolument. The class contains the following:
2. Two double data fields name **basic\_salary** and **tax\_relief** that represent basic salary and tax relief amounts. This data fields should be encapsulated.

**2 Marks**

1. A constructor that creates emolument with the specified Basic Salary and Tax Relief.

**3 Marks**

1. A method named **getBasicsalary()** that returns the basic\_salary.

**1 Mark**

1. A method named **getTaxRelief()** that returns the tax\_relief.

**1 Mark**

1. A method named **SSNIT()** that returns the computed SSNIT contribution of the emolument. Note: SSNIT contribution is 3.5% of Basic Salary.

**2 Marks**

1. A method named **taxableIncome()** that returns the computed Taxable Income of the emolument. Note: Taxable Income = Basic salary – (Tax relief + SSNIT contribution).

**2 Marks**

1. Create a class named **MyEmolument** that inherits from Emolument class. The class contains the following:
2. Two double data fields name **basic\_salary** and **tax\_relief**, that represent basic salary and tax relief amounts. This data fields should be encapsulated.

**2 Marks**

1. A **non-arg constructor** that creates a default emolument. The default values are **0** for basic salary and tax relief.

**2 Marks**

1. A constructor that creates **MyEmolument** with the specified basic salary and Tax relief.

**1 Mark**

1. A method named **incomeTax()** that return the computed Income Tax. Income Tax is calculated as follows:
2. The first 500.00 of Taxable Income, the tax rate is 5%.
3. The next 500.00 Taxable Income, the tax rate is 12.5%.
4. The rest, the tax rate is 17.5%.

**3 Marks**

1. A method named **totalDeduction()** that return the value of Total Deduction. Note: *Total Deduction = SSNIT Contribution + Income Tax*.

**1 Mark**

1. A method named **netSalary()** that return the value of Net Salary.

Note: *Net Salary = Basic salary – Total Deduction.*

**1 Mark**

1. Write a test program that creates MyEmolument object named **Staff\_Salary** with below requirement.
2. Accept input from the user using the input dialog.

**2 Marks**

1. Display the Basic Salary, Tax Relief, SSNIT Contribution, Taxable Income, Income Tax, Total Deduction and Net Salary as shown in the figure below.

**7 Marks**

**ANSWERS**

**2 A**

public class Emolument {

// Encapsulated fields

private double basic\_salary;

private double tax\_relief;

// Constructor

public Emolument(double basic\_salary, double tax\_relief) {

this.basic\_salary = basic\_salary;

this.tax\_relief = tax\_relief;

}

// Method to get basic salary

public double getBasicSalary() {

return basic\_salary;

}

// Method to get tax relief

public double getTaxRelief() {

return tax\_relief;

}

// Method to calculate SSNIT contribution (3.5% of Basic Salary)

public double SSNIT() {

return 0.035 \* basic\_salary;

}

// Method to calculate taxable income

public double taxableIncome() {

double ssnitContribution = SSNIT();

return basic\_salary - (tax\_relief + ssnitContribution);

}

// Main method for testing

public static void main(String[] args) {

Emolument emolument = new Emolument(5000.00, 1000.00);

System.out.println("Basic Salary: " + emolument.getBasicSalary());

System.out.println("Tax Relief: " + emolument.getTaxRelief());

System.out.println("SSNIT Contribution: " + emolument.SSNIT());

System.out.println("Taxable Income: " + emolument.taxableIncome());

}

}

**2 B**

public class MyEmolument extends Emolument {

// Encapsulated fields

private double basic\_salary;

private double tax\_relief;

// Non-argument constructor with default values

public MyEmolument() {

super(0.0, 0.0); // Call to the parent constructor with default values

this.basic\_salary = 0.0;

this.tax\_relief = 0.0;

}

// Constructor with specified basic salary and tax relief

public MyEmolument(double basic\_salary, double tax\_relief) {

super(basic\_salary, tax\_relief); // Call to the parent constructor

this.basic\_salary = basic\_salary;

this.tax\_relief = tax\_relief;

}

// Method to calculate Income Tax based on Taxable Income

public double incomeTax() {

double taxableIncome = super.taxableIncome(); // Use the parent method to get taxable income

double tax = 0.0;

if (taxableIncome <= 500) {

tax = taxableIncome \* 0.05; // 5% for the first $500

} else if (taxableIncome <= 1000) {

tax = 500 \* 0.05 + (taxableIncome - 500) \* 0.125; // 12.5% for the next $500

} else {

tax = 500 \* 0.05 + 500 \* 0.125 + (taxableIncome - 1000) \* 0.175; // 17.5% for the rest

}

return tax;

}

// Method to calculate Total Deduction

public double totalDeduction() {

return SSNIT() + incomeTax();

}

// Method to calculate Net Salary

public double netSalary() {

return basic\_salary - totalDeduction();

}

// Main method for testing

public static void main(String[] args) {

MyEmolument myEmolument = new MyEmolument(5000.00, 1000.00);

System.out.println("Basic Salary: " + myEmolument.getBasicSalary());

System.out.println("Tax Relief: " + myEmolument.getTaxRelief());

System.out.println("SSNIT Contribution: " + myEmolument.SSNIT());

System.out.println("Taxable Income: " + myEmolument.taxableIncome());

System.out.println("Income Tax: " + myEmolument.incomeTax());

System.out.println("Total Deduction: " + myEmolument.totalDeduction());

System.out.println("Net Salary: " + myEmolument.netSalary());

// Testing default constructor

MyEmolument defaultEmolument = new MyEmolument();

System.out.println("\nDefault Emolument:");

System.out.println("Basic Salary: " + defaultEmolument.getBasicSalary());

System.out.println("Tax Relief: " + defaultEmolument.getTaxRelief());

System.out.println("SSNIT Contribution: " + defaultEmolument.SSNIT());

System.out.println("Taxable Income: " + defaultEmolument.taxableIncome());

System.out.println("Income Tax: " + defaultEmolument.incomeTax());

System.out.println("Total Deduction: " + defaultEmolument.totalDeduction());

System.out.println("Net Salary: " + defaultEmolument.netSalary());

}

}

**2 C**

import javax.swing.JOptionPane;

import java.text.SimpleDateFormat;

import java.util.Date;

public class TestMyEmolument {

public static void main(String[] args) {

// Accept input from the user using dialog boxes

String salaryInput = JOptionPane.showInputDialog("Enter Basic Salary:");

String reliefInput = JOptionPane.showInputDialog("Enter Tax Relief:");

// Parse the inputs to double

double basicSalary = Double.parseDouble(salaryInput);

double taxRelief = Double.parseDouble(reliefInput);

// Create MyEmolument object

MyEmolument staffSalary = new MyEmolument(basicSalary, taxRelief);

// Get current date and time

SimpleDateFormat formatter = new SimpleDateFormat("EEEE, MMMM dd, yyyy, hh a z");

String currentDate = formatter.format(new Date());

// Prepare the output message

StringBuilder outputMessage = new StringBuilder();

outputMessage.append("Current date: ").append(currentDate).append("\n");

outputMessage.append("Basic Salary: ").append(staffSalary.getBasicSalary()).append("\n");

outputMessage.append("Tax Relief: ").append(staffSalary.getTaxRelief()).append("\n");

outputMessage.append("SSNIT Contribution: ").append(staffSalary.SSNIT()).append("\n");

outputMessage.append("Taxable Income: ").append(staffSalary.taxableIncome()).append("\n");

outputMessage.append("Income Tax: ").append(staffSalary.incomeTax()).append("\n");

outputMessage.append("Total Deduction: ").append(staffSalary.totalDeduction()).append("\n");

outputMessage.append("Net Salary: ").append(staffSalary.netSalary()).append("\n");

// Display the results in a dialog box

JOptionPane.showMessageDialog(null, outputMessage.toString(), "Staff Salary Details", JOptionPane.INFORMATION\_MESSAGE);

}

}