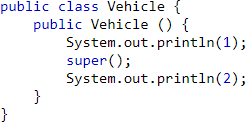
# Section A:

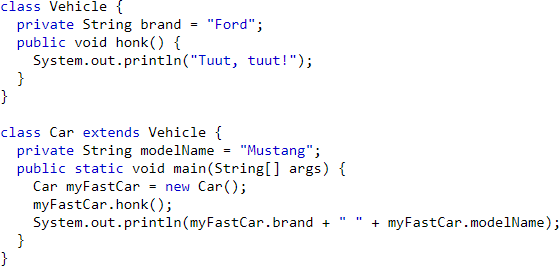
**TK1143 - Tutorial 2 Inheritance**

1. Given the following code. What is wrong with the following code? Why it is showing compilation error?

a)

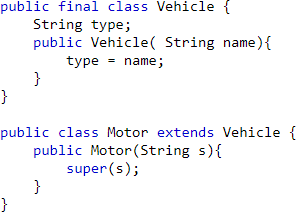
Answer:

Super must be the first statement in the constructor

b) [Grab your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

Answer:

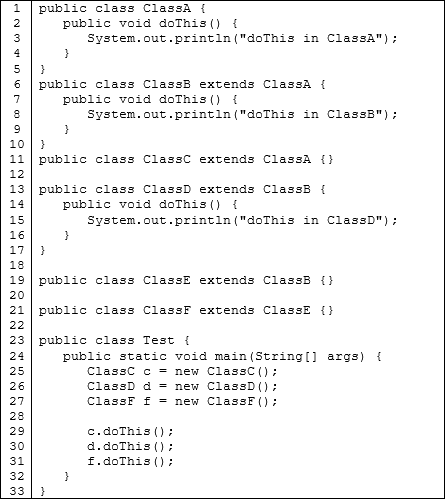
Brand has private access in class vehicle

c)

Answer:

Motor cannot be subclass of the final class vehicle and public class Motor should be declared in a file named Motor.java

1. What is the output of the following code?
   1. *Topic : Constructor*

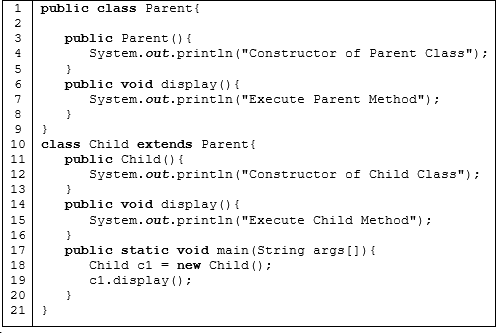


Answer:

doThis in ClassA

doThis in ClassD

doThis in ClassB

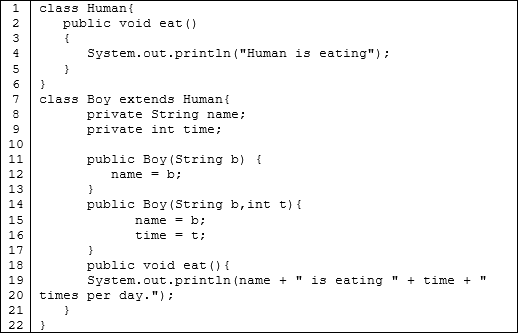
* 1. *Topic : Overriding method*

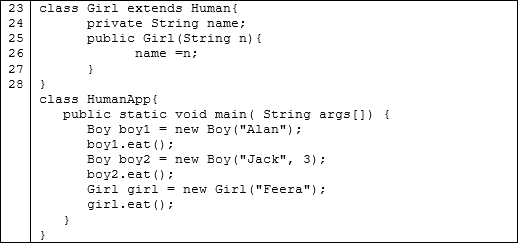
Answer:

Constructor of Parent Class

Constructor of Child Class

Execute Child Method

* 1. *Topic : Overloading method*



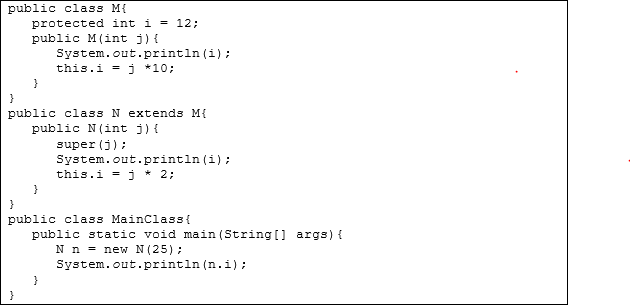
Answer:

Alan is eating 0 times per day

Jack is eating 3 times per day

Human is eating

* 1. *Topic : Super keyword and this*



Answer:

12

250

50

1. Draw a class diagram for java code program in question 2a), 2b) and 2c).

2c)

2b)

2a)

Human

+eat():void

Girl

-name:String

+Girl(String)

Boy

-name:String

-time:int

+Boy(String)

+Boy(String,int)

+eat():void

Child

+display:void

Parent

+display:void

ClassF

ClassE

ClassD

+doThis():void

ClassB

+doThis():void

ClassC

ClassA

+doThis():void

1. Draw a diagram to represent the hierarchical relationship between Mammal, Elephant, Reptile, Frog, Animal, Amphibians, Cow and Snake.

Frog

Snake

Cow

Elephant

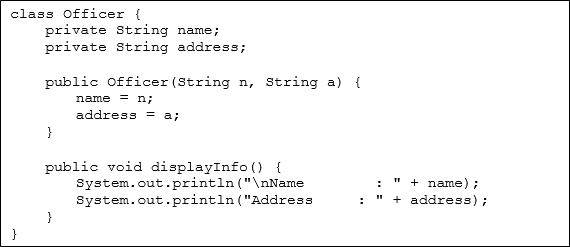
Amphibians

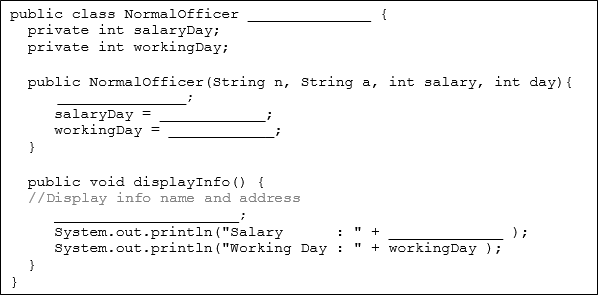
Animal

Reptile

Mammal

1. Based on the following class Officer, Answer 5 (a) – (c)



1. Fill in the blank to complete the class of NormalOfficer, which is a subclass of class Officer

salaryDay

Super.displayInfo()

day

salary

Super(n,a)

extend officer

1. Create another class ExecOfficer, which is subclass of class NormalOfficer. This class contains new attribute, that is bonus, and new method that display the bonus.

public class ExcecOfficer extends NormalOfficer{

private int bonus;

public ExecOfficer(String n, String a, int salary , int b){

super (n,a,salary,day);

bonus = b;

}

public void displayInfo() {

super.displayInfo();

System.out.print(“Bonus: “+ bonus);

}

}

1. Create TesterOfficer class that will test your execOfficer class.

import java.util.Scanner;

public class TesterOfficer {

public static void main(String[]args){

Scanner sc=new Scanner(System.in);

String name=sc.nextLine();

String address=sc.nextLine();

int salaryDay=sc.nextInt();

int workingDay=sc.nextInt();

int bonus=sc.nextInt();

ExecOfficer newExec=new ExecOfficer(name,address,salaryDay,

workingDay,bonus);

newExec.displayInfo();

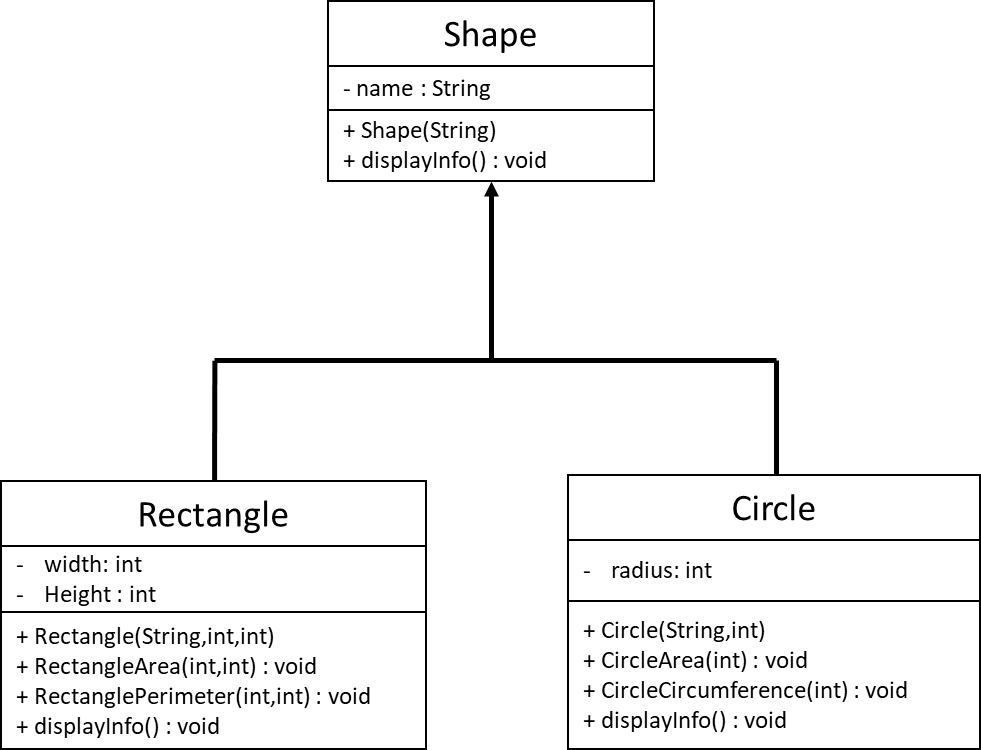
}

}

# Section B:

There are three (3) cases given in this section. The **UML diagram** and **description** is provided for each case. Your task is to write the program to implement the concept of inheritance for each case. You have to write a program of four classes, which is one superclass, two subclasses and one application class.

## CASE 1



* 1. Define a class named **Shape** (in file Shape.java) according to the following class description.

|  |  |
| --- | --- |
| Shape | Description |
| - name : String | - Variable to store shape name. |
| + Shape (String)  + displayInfo() : void | * Constructor that will initialize all the data members with the given value. * Method that will display all the information about shape. |

public class Shape{

private String name;

public Shape(String name){

this.name=name;

}

public void displayInfo() {

System.out.println("The shape is "+name);

}

}

* 1. Define a class named **Rectangle** (in file Rectangle.java) which is a subclass for the class Shape according to the following class description.

|  |  |
| --- | --- |
| Rectangle | Description |
| * width : int * height : int | - Variable to store the width |

|  |  |
| --- | --- |
|  | - Variable to store the height |
| + Rectangle (String, int, int)  + RectangleArea(int,int) : void  + RectanglePerimeter(int,int) : void  + displayInfo() : void | * Constructor that will initialize all the data members with the given value. * Method that calculate rectangle’s area * Method that calculate rectangle’s perimeter * Method that will display all the information of rectangle. |

class Rectangle extends Shape {

private int width;

private int height;

public Rectangle(String name,int width,int height) {

super(name);

this.width=width;

this.height=height;

}

public void RectangleArea(int width,int height) {

System.out.println("Rectangle Area: "+(height\*width));

}

public void RectanglePerimeter(int width,int height) {

System.out.println("Rectangle Perimeter: "+(width+width+height+height));

}

public void displayInfo() {

System.out.println("Width: "+width);

System.out.println("Height: "+height);

}

}

* 1. Define a class named **Circle** (in file Circle.java) which is a subclass for the class Shape according to the following class description.

|  |  |
| --- | --- |
| Circle | Description |
| - radius : int | - Variable to store the radius |
| + Circle (String, int)  + CircleArea (int) : void  + CircleCircumference(int) : void  + displayInfo() : void | * Constructor that will initialize all the data members with the given value. * Method that will calculate circle’s area * Method that will calculate circle’s circumference * Method that will display all the information of circle. |

import java.text.DecimalFormat;

public class Circle extends Shape {

private int radius;

DecimalFormat df=new DecimalFormat("0.000");

public Circle(String name, int radius) {

super(name);

this.radius=radius;

}

public void CircleArea(int radius) {

System.out.println("Circle Area: "+df.format(Math.PI\*Math.pow(radius,2)));

}

public void CircleCircumference(int radius) {

System.out.println("Circle Circumference: "+df.format(2\*Math.PI\*radius));

}

public void displayInfo() {

System.out.println("Radius: "+radius);

}

}

* 1. Define a class name **ShapeApp** (ShapeApp.java) which contains the main method to test the functionality of the class above. Refer to the sample output given :

public class ShapeApp {

public static void main(String[]args) {

String name ;

int width,height,radius;

Shape myshape=new Shape("Rectangle");

Rectangle myRectangle = new Rectangle( "Rectangle",4,5);

myshape.displayInfo();

myRectangle.displayInfo();

myRectangle.RectangleArea(4,5);

myRectangle.RectanglePerimeter(4,5);

System.out.println();

Shape mshape=new Shape("Circle");

mshape.displayInfo();

Circle myCircle=new Circle("Circle",6);

myCircle.displayInfo();

myCircle.CircleArea(6);

myCircle.CircleCircumference(6);

}

}

## SAMPLE OUTPUT

The Shape is Rectangle Width: 4

Height: 5

Rectangle Area: 20

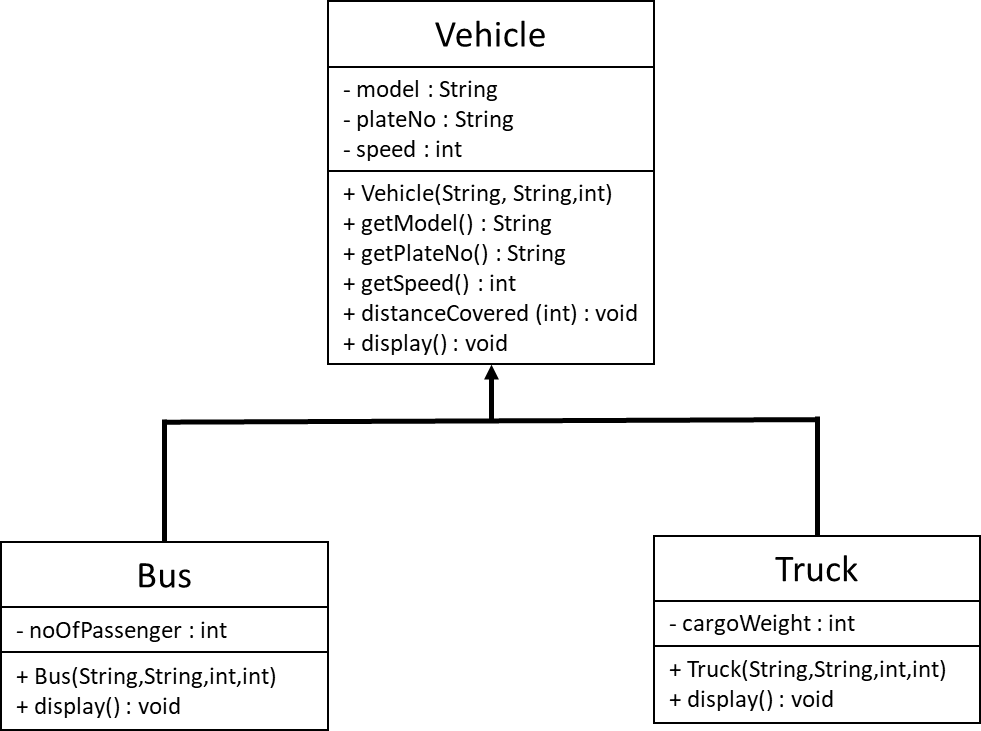
Rectangle Perimeter: 18

The Shape is Circle Radius: 6

Circle Area: 113.112

Circle Circumference: 37.704

## CASE 2



* 1. Define a class named **Vehicle** (in file Vehicle.java) according to the following class description.

|  |  |
| --- | --- |
| Vehicle | Description |
| * model : String * plateNo : String * speed : int | * Variable to store the model of vehicle. * Variable to store the vehicle plate number. * Variable to store the vehicle speed (km/hour). |
| + Vehicle (String, String, int)  + getModel() : String  + getPlateNo() : String  + getSpeed() : int  + distanceCovered(int) : void  + display() : void | * Constructor to set the initial value of model, plate number and speed for vehicle object. * Method to get the vehicle model. * Method to get the vehicle plate number. * Method to get the vehicle speed (km/hour). * Method that will take time (in hour) as an argument. This method calculate the distance for the given time and speed and print the value of distance. * Hint : can use this formula, *speed = distance/time* * Method that will display all the information of a vehicle. |

public class Vehicle { private String model; private String plateNo; private int speed;

public Vehicle (String m, String p, int s) { model = m;

plateNo = p; speed = s;

}

public String getModel() { return model;

}

public String getPlateNo() { return plateNo;

}

public int getSpeed() { return speed;

}

public void distanceCovered(int time) {

System.out.println("Distance Covered (km) with speed "+ getSpeed() + "km/h and " + time + " is " + (speed\*time) + "km");

}

public void display() { System.out.println("Vehicle Model: " + model);

System.out.println("Vehicle Registration No: " + plateNo); System.out.println("Speed (km/h): " + speed);

}

}

* 1. Define a class named **Bus** (in file Bus.java) which is a subclass for the class Vehicle according to the following class description.

|  |  |
| --- | --- |
| Bus | Description |
| - noOfPassenger : int | - Variable to store the number of passenger for bus. |
| + Bus (String, String, int, int)  + display() : void | * Constructor that will initialize all the data members with the given value. * Method that will display all the information of a bus. |

public class Bus extends Vehicle { private int noOfPassenger;

public Bus(String m, String p, int s, int pas) { super(m, p, s);

noOfPassenger = pas;

}

public void display() { super.display();

System.out.println("No of Passenger: "+noOfPassenger);

}

}

* 1. Define a class named **Truck** (in file Truck.java) which is a subclass for the class Vehicle according to the following class description.

|  |  |
| --- | --- |
| Truck | Description |
| - cargoWeight : int | - Variable to store the cargo weight limit (kg). |
| + Truck (String, String, int, int)  + display() : void | * Constructor that will initialize all the data members with the given value. * Method that will display all the information of a truck. |

public class Truck extends Vehicle { private int cargoWeight;

public Truck(String m, String p, int s, int c){ super(m, p, s);

cargoWeight = c;

}

public void display() { super.display();

System.out.println("Cargo Weight Limit (kg): "+cargoWeight);

}

}

* 1. Define a class name **VehicleApp** (VehicleApp.java) which contains the main method to test the functionality of the class above. Refer to the sample output given :

import java.util.Scanner; public class VehicleApp {

public static void main(String[] args) { Scanner input = new Scanner(System.in);

int vehicle = input.nextInt(); if (vehicle == 1) {

String model = input.next(); input.nextLine();

String plateNo = input.nextLine(); int speed = input.nextInt();

int time = input.nextInt();

int noOfPassanger = input.nextInt();

Bus myBus = new Bus (model, plateNo, speed, noOfPassanger);

myBus.display(); myBus.distanceCovered(time);

}

else if(vehicle == 2) {

String model = input.next(); input.nextLine();

String plateNo = input.nextLine(); int speed = input.nextInt();

int time = input.nextInt();

int cargoWeight = input.nextInt();

Truck myTruck = new Truck (model, plateNo, speed, cargoWeight); myTruck.display();

myTruck.distanceCovered(time);

}

}

}

## SAMPLE OUTPUT

Vehicle Model: Toyota

Vehicle Registration No: BPX 2210 Speed (km/h): 160

No of Passenger: 44

Distance Covered (km) with speed 160km/h and 5 hours is 800km

Vehicle Model: Honda

Vehicle Registration No: VXC 1123 Speed (km/h): 110

Cargo Weight Limit (kg): 500

Distance Covered (km) with speed 110km/h and 3 hours is 330km

## CASE 3

Without UML Diagram, create **FIVE** (**5**) classes based on description below. Try drawing UML diagram first before you starting create program code.

* 1. Define a class named **Faculty** with following characteristics
     + Parameterized constructor(name, basic) to initialize all data members of a class.
     + Data member **name** of String type to store names of the faculty.
     + Private data member **basic** of int type to store the basic component of salary.
     + **getDetails** method to return name & salary separated by space in "NAME SALARY" format.
     + **getSalary** method to calculate and return double type salary.

import java.text.DecimalFormat; public class Faculty {

DecimalFormat df = new DecimalFormat("0.00"); protected String name;

protected int basic;

public Faculty(String n, int b) { name = n;

basic = b;

}

public double getSalary() { return basic;

}

public String getDetails() {

return(name +" "+ df.format(getSalary()));

}

}

* 1. Define **AssistantProfessor** which extends **Faculty** has following characteristics
     + Parameterized constructor(name, basic, DA) to initialize all data members of a class. Call the base class constructor to initialize name & basic.
     + Data member **DA** (integer type) which is % of basic salary.
     + Override **getSalary** to calculate and return salary as, "basic + ((basic \* DA)/100)".
     + **getDetails** method to return name & salary separated by space in "Assistant Professor NAME SALARY" format.

import java.text.DecimalFormat;

public class AssistantProfessor extends Faculty{

DecimalFormat df = new DecimalFormat("0.00"); private int DA;

public AssistantProfessor(String n, int b, int DATA) { super(n,b);

DA = DATA;

}

public double getSalary() {

return super.getSalary()+(super.getSalary() \* DA)/100;

}

public String getDetails() {

return("Assistant Professor "+name +" RM"+ df.format(getSalary()));

}

}

* 1. Define **AssociateProfessor** which extends **AssistantProfessor** has following characteristics
     + Parameterized constructor(name, basic, DA, MedAllowance) to initialize all data members of a class. Call the base class constructor to initialize other values.
     + Data member **MedAllowance** (integer type) which fixed amount.
     + Override **getSalary** to calculate and return salary as, basic + ((basic \* DA)/100) + MedAllowance.
     + **getDetails** method to return name & salary separated by space in "Associate Professor NAME SALARY" format.

import java.text.DecimalFormat;

public class AssociateProfessor extends AssistantProfessor{

DecimalFormat df = new DecimalFormat("0.00"); private int MedAllowance;

public AssociateProfessor(String n, int b, int DA, int med) { super(n, b, DA);

MedAllowance = med;

}

public double getSalary() {

return super.getSalary() + MedAllowance;

}

public String getDetails() {

return("Associate Professor "+name +" RM"+ df.format(getSalary()));

}

}

* 1. Define **Professor** which extends **AssociateProfessor** has following characteristics
     + Parameterized constructor(name, basic, DA, MedAllowance, OtherAllowance) to initialize all data members of a class.
     + Data member **OtherAllowance** (integer type) which is % of total income.
     + Override **getSalary** to calculate and return salary as, "Salary of AssociateProfessor"

+ (OtherAllowance% of "Salary of AssociateProfessor").

* + - **getDetails** method to return name & salary separated by space in "Profesor NANE SALARY" format.

import java.text.DecimalFormat;

public class Professor extends AssociateProfessor {

DecimalFormat df = new DecimalFormat("0.00"); private int OtherAllowance;

public Professor(String n, int b, int DA, int med, int o) { super(n, b, DA, med);

OtherAllowance = o;

}

public double getSalary() {

return super.getSalary() + (OtherAllowance \* super.getSalary() / 100);

}

public String getDetails() {

return("Profesor "+name +" RM"+ df.format(getSalary()));

}

}

* 1. Define **FacApp**, which contains the main method to test the functionality of the classes above. Refer to the sample output given :

import java.util.Scanner; public class FacApp {

public static void main(String[]args) {

Scanner input = new Scanner(System.in); int choice = input.nextInt();

String name = input.nextLine();

name = input.nextLine();

int basic = input.nextInt(); int DA = input.nextInt();

basic, DA);

if (choice == 1) {

AssistantProfessor myAsisProf = new AssistantProfessor (name,

System.out.println(myAsisProf.getDetails());

}

else if (choice == 2) {

int MedAllowance = input.nextInt();

AssociateProfessor myAsoProf = new AssociateProfessor (name, basic, DA, MedAllowance);

System.out.println(myAsoProf.getDetails());

}

else if (choice == 3) {

int MedAllowance = input.nextInt(); int otherAllowance = input.nextInt();

Professor myProf = new Professor (name, basic, DA, MedAllowance,

otherAllowance);

}

System.out.println(myProf.getDetails());

input.close();

}

}

## SAMPLE INPUT

3

Luth Wafiy 450000

125

10000

5

where,

* First line represents a choice for which class to instantiate : 1 – AssistantProfessor, 2 – AssociateProfessor, 3 – Professor.
* Second line represents the name of the professor.
* Third line represents basic salary.
* Forth line represents DA value.
* Fifth line represents MedAllowance value.
* Sixth line represents OtherAllowance value.

## SAMPLE OUTPUT

Profesor Luth Wafiy RM1073625