

NAME :SANJAY M

USN:1NH18CS170

SEM/SEC: 4 C

PROJECT TOPIC:PAYSCALE SALARY CALCULATOR

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

Earnings may be a sort of payment from associate degree leader to associate degree worker, which can be per associate degree employment agreement. It's having contrasted with piece wages, wherever every job, hour, or different unit is paid individually, instead of on a periodic basis. From the purpose of read of running a business, earnings may also be viewed because the price of deed and holding human resources for running operations, and is then termed personnel expense or earnings expense. In accounting, salaries square measure recorded on payroll accounts and maintained for the every and each worker of the organization or company.

Salary may be a mounted quantity of cash or compensation paid to associate degree worker by associate degree leader reciprocally for work performed. earnings is usually paid in mounted intervals, as an example, monthly payments of one-twelfth of the annual earnings.

Salary is often determined by scrutiny market pay rates for individuals activity similar add similar industries within the same region. earnings is additionally determined by leveling the pay rates and earnings ranges established by a private leader. earnings is additionally full of the quantity of individuals offered to perform the particular job within the employer's employment scene.

Payment of salaries to the staff is that the heart of any Human Resource System of a ccompany. The answer needs to watch out of the calculation of earnings as per rules of the corporate, tax calculation and varied deductions to be done from the earnings together with statutory deductions like tax and provident fund deductions. It's to get pay-slip, cheque outline and MIS reports. The corporate company has completely different classes of staff like daily wagers, monthly payments etc. thus we want to calculate salaries consequently.

1.2 OBJECTIVES

Main aim of developing salary counter is to provide an easy way not only to automate all functionalities involved managing leaves and salary for the employees of Company, but also to provide full functional to management of Company with the details about usage of leave facility. Also calculate the salary of the employee accordingly to their position in the company whether they belong to daily wagers or monthly wagers.

1.3 METHODOLOGY TO BE FOLLOWED

1. Data Gathering
- 2.Data Analysis and Problem Identification
- 3.System Model and Design
- 4.Software Development
- 5.System Testing and Implementation
- 6.Evaluation

1.4 EXPECTED OUTCOMES

The salary is calculated accordingly the input like daily wagers, weekly wagers and monthly wagers. The tax is deducted and other allowances are deducted and the net total amount is calculated and given as the output. The overtime is also calculated and paid extra for the overtime then the normal hours of work. The salary amount is displayed to the user.

The taxation of the net amount is calculated accordingly the type of payment and the rate of taxation is different for different mode of payment to the employee of the organization or company. The deduction amount is calculated and deducted from the net amount calculated before and the final amount is calculated and displayed to the user. The deduction amount and the rate of deduction is also displayed to the user.

1.5 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirements

RAM : 3 GB or more

Processor : Intel core i5 or more

Software requirements

Operating System (Windows 10 or any other OS)

Programming language JAVA

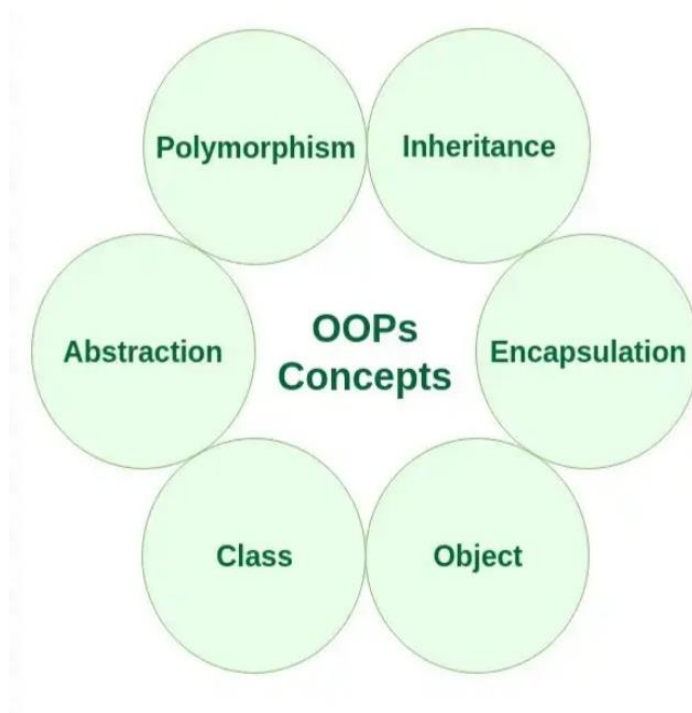
Java SDK or JRE 1.6 or higher

CHAPTER 2

OBJECT ORIENTED CONCEPTS

Object Oriented Programming (OOPs) Concept in Java

Object-oriented programming: As the concept suggested from the name, Object-Oriented Programming or OOPs refers to languages that uses objects or instances in programming. Object-oriented programming concepts mainly aims to implements or involves some real-world entities like inheritance, data hiding, polymorphism etc in programming. The main aim of OOP is to bind together the members of the class such as data and the functions that operate on them so that no other elements of the code can access or use this data except that function with accessibility to the members.



2.a OOPS CONCEPTS

- Polymorphism
- Inheritance

- Encapsulation
- Abstraction
- Class
- Object
- Message Passing

2.1 CLASS

A Java class is a solitary, lucid unit of Java code which has a place together. A Java class may contain a blend of factors and strategies. Gathering factors and procedure on these factors into Java classes makes it simpler to structure your Java program when it gets too enormous to fit serenely inside a solitary Java class. A Java class must be put away in its own document. Hence, as the class develops, the record you are altering develops as well, and gets more diligently to keep a review of in your mind. Your Java application will commonly need to contain at any rate a solitary Java class, however it might contain the same number of classes as you decide to isolate your application into. Java additionally accompanies a ton of predefined classes for you, so you don't need to code each and every capacity you may want yourself.

Local variables – Variables are defined or declared in within or inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method, the accessibility of the variables is within the defined block and the variable will be destroyed when the method has completed.

Instance variables – Instance variables are variables defined or declared within a class but outside any method within the scope of the class. These variables are defined when the class is instantiated. Instance variables can be accessed from inside any method of the class, constructor or blocks of that particular class.

Class variables – Class variables are variables declared within a class, outside any method, with the static keyword and they can be accessed without any objects.

Constructors

A constructor is utilized to instate the occurrence variable of the class during the item creation. Each class has a constructor. On the off chance that we don't unequivocally compose a constructor for a class, the Java compiler fabricates a default constructor for that class.

Each time another item is made, at any rate one constructor will be conjured. The fundamental guideline of constructors is that they ought to have a similar name as the class. A class can have more than one constructor which can be overloaded.

2.2 OBJECTS

Objects are the key measure to understanding object-oriented technology.

Real-world objects share 2 characteristics: all of them have state and behavior. For example Dogs are having state such as name, color, breed, hungry and behavior characteristics barking. Another example Bicycles have state characteristics such as current gear, current pedal cadence, current speed and behavior characteristic such as changing gear, dynamical pedal cadence, applying brakes. distinctive the state and behavior for real-world objects could be a good way to start thinking in terms of object-oriented programming.

Software objects is a measure conceptually kind of like real-world objects: they too incorporates state and connected behavior. Associate in Nursing object stores its state in fields and exposes its behavior through strategies. Strategies treat Associate in Nursing object's internal state and function the first mechanism for object-to-object communication. Concealing internal state Associate in requiring all interaction to be performed through an object's strategies is understood as information encapsulation.

Object are used to access the members of the class such as variables and methods. The object is very important while handling many classes under single Java application. A Java application may contain n number of classes and methods so the accessibility is mainly with the help of the object or reference variable to the class.

2.b DIFFERENCES BETWEEN OBJECT AND CLASS

OBJECT	CLASS
Object is an instance of a class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen, table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newInstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.

2.3 INHERITANCE

Inheritance in Java could be a mechanism within which one object acquires all the properties and behaviors of a parent object. it's a very important a part of OOPS(Object oriented programming system).The idea behind inheritance in Java is that you just will produce new categories that area unit engineered upon existing categories.

After you inherit the characters from associate existing category, you'll be able to reprocess strategies and fields of the parent category. Moreover, you'll be able to add new strategies and fields in your current category conjointly. Inheritance represents the IS-A relationship that is additionally referred to as a parent child relationships.

Terms used in Inheritance

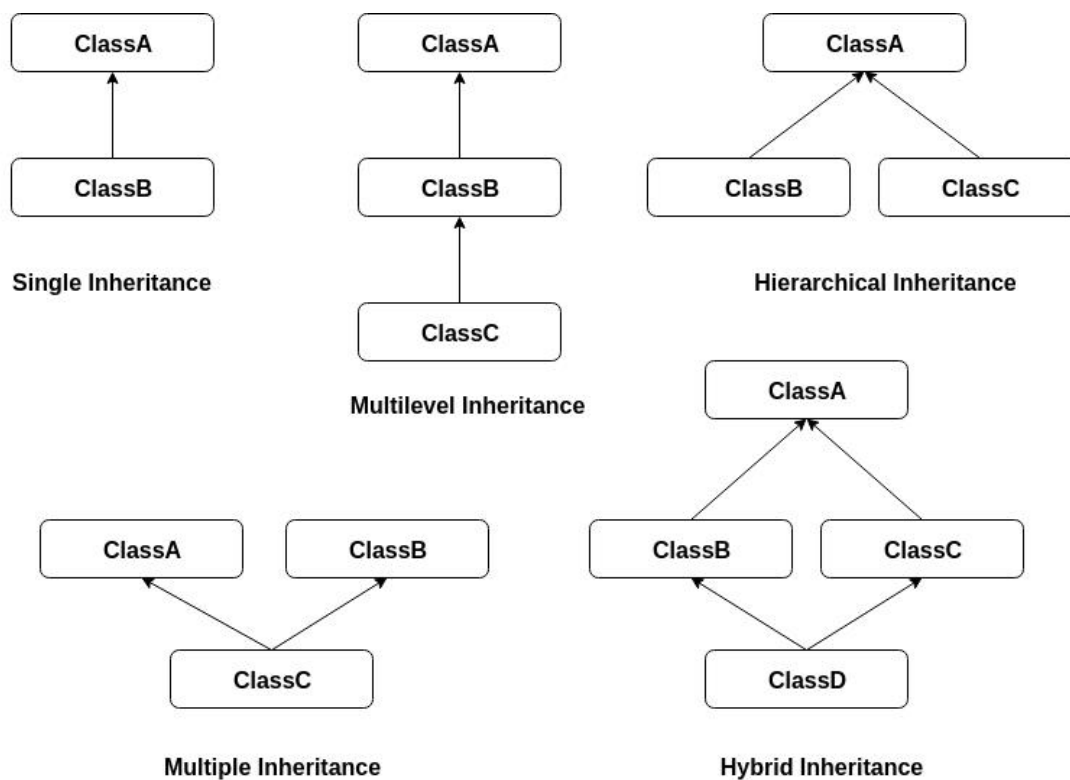
Class: A class is a group of objects within a single unit which have common properties. It is defined as a template or blueprint of real world problems from which objects are created.

Sub Class/Child Class: Subclass is a class which inherits the properties from other class. The other names of sub class is a derived class, extended class, or child class.

Super Class/Parent Class: Super class is the class where the subclass inherits the features. It is also referred or known as base class or a parent class.

TYPES OF INHERITANCE

- 1) Single Inheritance
- 2) Multiple Inheritance (Through Interface)
- 3) Multilevel Inheritance
- 4) Hierarchical Inheritance
- 5) Hybrid Inheritance (Through Interface)



2.c TYPES OF INHERITANCE

1. Single Inheritance in Java

Single Inheritance is the simple and easy inheritance of all of the five types available, where a class extends another class(Only one class) then we call it as Single inheritance. Class B extends or inherits only one class Class A. Here Class B is the Sub

class and Class A will be only Super class available .

2. Multiple Inheritance in java

Multiple Inheritance is one class extending more than one super class to the sub child class. One of the type of Inheritance that is not available in many OOP language. Multiple Inheritance is not supported by many Object Oriented Programming languages such as Java, Small Talk, C# etc.. (C++ Supports Multiple Inheritance). As the Child class has to manage the dependency of more than one Parent class which will be very difficult to the child. But multiple inheritance in Java is implemented by use of the Interfaces.

3. *Multilevel Inheritance in Java*

In Multilevel Inheritance a derived class will be inheriting a parent class and as well as the derived class act as the parent class to other class or the inheritance of the class is by levels. ClassB inherits or extends the property of ClassA and again ClassB act as a parent for ClassC. In this case ClassA parent for ClassB and ClassB parent for ClassC.

4. *Hierarchical Inheritance in Java*

In Hierarchical inheritance one parent class will be inherited by many sub classes and each sub class can implement different operations on the parent members.

ClassA will be inherited by many classes like ClassB, ClassC and ClassD. ClassA will be acting as the parent class for ClassB, ClassC and ClassD.

5. *Hybrid Inheritance in Java*

Hybrid Inheritance is the combination of two available inheritance that are Single and Multiple Inheritance. Again Hybrid inheritance is also not directly implemented in Java only through interface we can achieve this in Java. ClassA is the Parent class for ClassB & ClassC and ClassB & ClassC will be acting as Parent for ClassD.

2.4 POLYMORPHISM

Polymorphism is considered or used as one of the most important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways on different places. In other words, polymorphism allows programmers to define one interface and have multiple implementations for the same interface. The word “poly” actually means many and “morphs” means forms of same, So in total it means many forms. There are two types of polymorphism available in Java that are : compile-time polymorphism and runtime polymorphism. Polymorphism in java is implemented by two concepts such as method overloading and method overriding.

In Java polymorphism is particularly divided into two types:

->Compile time Polymorphism

->Runtime Polymorphism

Compile time polymorphism: It is also mentioned as static polymorphism and this sort of polymorphism is achieved by implementing function overloading or operator overloading.

Method Overloading: When there are multiple or more than 1 functions with same name but with different parameters then these functions are overloaded. Functions are often overload by change in number of arguments or/and change in kind of arguments.

Operator Overloading: Java also provide choice to overload operators. for instance, we'll make the operator ('+') for string class to concatenate two strings. we all know that this is often the addition operator whose task is to feature two operands. So one operator '+' when used between integer variables , adds them and when used between string operands, concatenates them. In java, Only "+" operator are often overloaded:

>To add integers

->To concatenate strings

Runtime polymorphism: It is also mentioned as Dynamic Method Dispatch. it's a

process during which a call to the overridden method is resolved at Runtime. this sort of polymorphism is achieved by Method Overriding.

Method overriding: On the opposite hand, occurs when a derived class features a definition for one among the member functions of the bottom class. That base function is claimed to be overridden

In this code there is a concept of overloading in the method called show() to display the basic details of the company and the details of the employee and showed the concept of overriding in the method payment() to calculate the salary of the employee.

2.5 ABSTRACT CLASS

A class which is proclaimed as unique is known as a theoretical class or abstract class. It can have dynamic and non-conceptual techniques or methods. It should be expanded and its strategy executed. It can't be instantiated.

A theoretical class must be pronounced with a abstract catchphrase.

It can have dynamic and non-theoretical techniques.

It can't be started up.

It can have constructors and static techniques too.

It can have last techniques which will compel the subclass not to change the body of the strategy.

Abstract Classes Contrasted with Interfaces

Theoretical classes are like interfaces. You can't launch them, and they may contain a blend of strategies pronounced with or without a usage. In any case, with unique classes, you can announce fields that are not static and last, and characterize open, ensured, and private solid strategies. With interfaces, all fields are consequently open, static, and last, and all strategies that you proclaim or characterize (as default techniques) are open.

Likewise, you can broaden just one class, regardless of whether it is dynamic, while you can execute any number of interfaces.

Which would it be advisable for you to utilize, conceptual classes or interfaces?

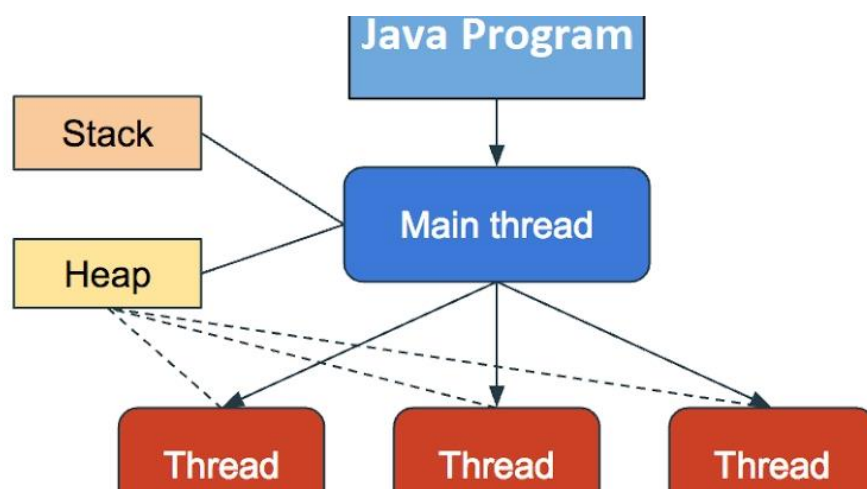
Consider utilizing dynamic classes if any of these announcements apply to your circumstance:

When one application need to share code among a few firmly related classes. You expect that classes that broaden your theoretical class have numerous basic strategies or fields, or require get to modifiers other than open, (for example, ensured and private). You need to proclaim non-static or non-last fields. This empowers you to characterize strategies that can get to and change the condition of the item to which they have a place.

.2.6 MULTI THREADING

Multithreading in JAVA is a process of executing multiple threads which will executed simultaneously. A thread is a lightweight sub-process of a single process, the smallest unit of processing. Multiprocessing and multithreading, both are used for the implementation of multitasking.

However, we use multithreading than multiprocessing because the threads use a shared memory area. They don't allocate or use some separate memory area so save memory, and context switching between the thread process takes less time than process.



2.d MULTIPROCESSING

2.7 JAVA IO FUNCTION

Java languages brings various streams with its I/O package that helps the user to perform necessary input output operations. These streams support all the kinds of objects, data-types, characters, files etc to completely execute the I/O operations. Before exploring various input and output streams lets inspect 3 standard or default streams that Java possesses to supply which are also common most in use:

System.in: This is that the quality input stream that's used to read characters from the keyboard or the opposite standard data data input device .

System.out: This is the quality output stream that's wont to produce the results of a program on an output device just like the display screen. Here may be a list of the varied print functions that we use to output statements:

->print():

This method in Java is employed to display a text on the console. This text is passed because the parameter to the present method within the sort of String. This method will print the text on the console and thus the cursor remains at the highest of the text at the console. subsequent printing takes place from just here.

Syntax: `System.out.print(parameter);`

->println():

This method in Java is additionally wont to display a text on the console. It prints the text on the console and therefore the cursor moves to the beginning of subsequent line at the console. subsequent printing takes place from subsequent line.

Syntax: `System.out.println(parameter);`

System.err: This is the standard error stream that's used to output all the error data that a program might throw, on a monitor or any standard output device. This stream also uses all the two above-mentioned functions to output the error data

in this code the input will be taken in the main function and methods will be called and the salary will be calculated accordingly .After the salary is calculated the details of the employee will be stored according to the type of payment in different files.

2.8 JAVA PACKAGES

Packages are utilized in Java so on stop naming conflicts, and also to manage access, and to make searching/locating and usage of classes, interfaces, enumerations, pre defined annotations easier, etc.

A Package are often referred as a grouping of related types (classes, interfaces) providing access protection and namespace management. A java package is a collection or group of classes, interface and sub-packages. Package in java can be categorized or classified in two form such as built-in package and user-defined package. There are many built-in packages available in Java language such as lang, io, util, sql etc.

ADVANTAGE OF JAVA PACKAGE

- Java package is important to categorize the classes and interfaces. So that they can be easily maintained in separate groups.
- Java package provides access protection for the members of the classes.
- Java package removes naming collision.

2.8 EXCEPTION HANDLING

The Exception Handling in Java is one of the most powerful mechanism to handle the runtime errors which occurs during execution of the application, so that normal flow of the application can be maintained and the termination of the application is normal.

Types of Java Exceptions

There are mainly two types of exceptions which can be encountered: checked and unchecked.

1. Checked Exception
2. Unchecked Exception
3. Error

1) Checked Exception

The classes which directly inherit the properties of Throwable without any separate implementation for that class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes which inherit the properties of RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked or found at time of compile-time, but they are checked at runtime.

3) Error : Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

CHAPTER 3

DESIGN GOALS

The application should be capable of performing the following functions:

- Store basic information of the employees.
- Store salary information such as, working hours, salary per hour, salary before tax, tax Percentage, total amount of tax paid , salary after tax, social security fee, on monthly basis.
- After calculating the salary the details and the salary of the employee should be stored in the files .
- If we enter the details of the new employee the details of the employee and the salary should be appended in the file.

Algorithm

- Start
- The user must input the employee's details like name, id etc.
- The number of hours worked by the employee is also inputted.
- The method of payment for the employee is required for calculating the salary.
- The option available are daily wages, weekly wages and monthly wages.
- The salary amount is calculated accordingly the method of payment.
- The salary amount is calculated on the basis of numbers of hours worked by the employees.
- The overtime is calculated and paid extra amount for employees.
- Taxation of the salary is done accordingly the type of the payment.
- The deduction amount is deducted from the net amount calculated.
- The salary amount after the employee is deduction amount subtracted from the total amount calculated with respect to hours and type of payment.
- The salary amount is displayed to the user along the employees details which has be inputted by the user.
- End

CHAPTER 4

IMPLEMENTATION

MODULE 1 FUNCTIONALITY

```
abstract class company {
    String comname,add;
    long ph;
    abstract void show();
}
class details extends company{
    String name;
    int id,hr,dy,ot;
    int type;
    static double pay,rate;
    details(String s,int i,int n,int j,double r)
    {
        name=s;
        id=i;
        type=n;
        hr=j;
        rate=r;
    }
    details()
    {
        comname="PAYSCALE";
        add="BELLENDUR,BANGALORE ";
        ph=1234567890;
    }
    void show()
    {
        System.out.println("\nCOMPANY NAME:"+comname);
        System.out.println("ADDRESS:"+add);
```

```
        System.out.println("PH:"+ph);

System.out.println("_____
__");
    }

    public static void show(employee ob)
    {
        try{
            File fl =new File("C:\\mini files\\mpfile.txt");
            FileWriter fr = new FileWriter(fl,true);
            PrintWriter fp = new PrintWriter(fr);

            fp.println();
            fp.println("EMPLOYEE NAME="+ob.name);
            fp.println("EMPLOYEE ID="+ob.id);
            fp.println("PAYMENT TYPE="+ob.type);
            fp.println("RATE PER HOUR:"+rate);
            fp.close();
        }catch(IOException e) {
            System.out.println("Error in writing file");
        }
        System.out.println("EMPLOYEE NAME="+ob.name);
        System.out.println("EMPLOYEE ID="+ob.id);
        System.out.println("PAYMENT TYPE="+ob.type);
        System.out.println("RATE PER HOUR:"+rate);
    }

    public void payment()
    {
        //switching on the options available like daily, weekly and monthly wagers.

        switch(type)
        {
            //daily wagers salary amount is calculated here

            case 1:
```

PAYSCALE SALARY CALCULATOR

```
pay=hr*rate;
System.out.println("PAYMENT FOR "+hr+" HOURS IS Rs:"+pay);
break ;
//weekly wagers salary amount is calculated here

case 2:
//overtime is calculated and paid extra amount for extra hours of working
if(hr>40)
{
    ot=hr-40;
    pay=hr*rate+ot*(rate+5);
    System.out.println("PAYMENT FOR 40 HOURS AND OVERTIME OF "+ot+" HOURS IS
Rs:"+pay);
}
//if overtime is not there normal number of hours worked is used for calculating the
salary amount for the employee

else
{
    pay=hr*rate;
    System.out.println("PAYMENT FOR "+hr+" HOURS IS Rs:"+pay);
}
break;
case 3:
if(hr>192)
{
    ot=hr-192;
    dy=hr/8;
    pay=hr*rate+ot*(rate+25);
    System.out.println("PAYMENT FOR "+dy+" DAYS AND OVERTIME OF "+ot+" HOURS
IS Rs:"+pay);
}
else
{
```

```
dy=hr/8;
int m=hr%8;
pay=hr*rate;
System.out.println("PAYMENT FOR "+dy+" DAYS AND "+m+" HOURS IS Rs:"+pay);
}
break;
} } }
```

MODULE 2 FUNCTIONALITY

```
class taxes extends Thread{
    double b,p,amt;
    int t,r;
    taxes(double x,int y)
    {
        p=x;
        t=y;
    }
    void deduct()
    {
        b=p*r/100;
        amt=p-b;
    }
    public void run()
    {
        if(t==1)
        {
            r=3;
            deduct();
        }
        else if(t==2)
        {
```

```
        r=5;
        deduct();
    }
    else if(t==3)
    {
        r=10;
        deduct();
    }
    if(t==1 || t==2 || t==3)
    {
        System.out.println("DEDUCTION RATE:"+r+"%");
        System.out.println("DEDUCTION AMOUNT: "+b);
        System.out.println("AMOUNT AFTER TAXATION:"+amt);
        try{
            File fl =new File("C:\\mini files\\mpfile.txt");
            FileWriter fr = new FileWriter(fl,true);
            PrintWriter pr = new PrintWriter(fr);
            pr.println("DEDUCTION RATE:"+r+"%");
            pr.println("DEDUCTION AMOUNT: "+b);
            pr.println("AMOUNT AFTER TAXATION:"+amt);
            pr.close();
            fr.close();
        }catch(IOException e) {
            System.out.println("Error in writing file");
        }
    }
    else
        System.out.println("NO DEDUCTION");
    } }
```

MODULE 3 FUNCTIONALITY

```
class employee extends details {
employee(String a,int b,int c,int d,double e)
{
    super(a,b,c,d,e);
    super.payment();
}
public void payment()
{
    System.out.println("INVALID OPTION");
    System.out.println("CHOOSE THE CORRECT OPTION");
}
public static void main(String[] args) {
    details ob= new details();
    ob.show();
    System.out.println("ENTER NAME:");
    Scanner o = new Scanner(System.in);
    String t=o.next();
    System.out.println("ENTER ID:");
    int j=o.nextInt();
    int h;
    try{
        System.out.println("ENTER THE NO OF HOURS WORKED:");
        h=o.nextInt();
        int p=1/h;
    }catch(Exception e)
    {
        System.out.println("INVALID OPERATION");
        return ;
    }
}
```

```
}  
System.out.println("1.DAILY WAGES\n2.WEEKLY WAGES\n3.MONTHLY WAGES");  
System.out.println("ENTER THE TYPE OF PAYMENT:");  
int i = o.nextInt();  
System.out.println("ENTER THE RATE PER HOUR:");  
double rt = o.nextDouble();  
employee obj = new employee(t,j,i,h,rt);  
if(i>=4)  
{  
    obj.payment();  
}  
else  
    show(obj);  
taxes a = new taxes(pay,i);  
a.start();  
}  
}
```


CHAPTER 5

SAMPLE OUTPUT

5.1 DAILY WAGES OUTPUT



```
<terminated> employee (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Apr-2023 10:00:00 AM)
ENTER NAME:
akshay
ENTER ID:
22
ENTER THE NO OF HOURS WORKED:
50
1.DAILY WAGES
2.WEEKLY WAGES
3.MONTHLY WAGES
ENTER THE TYPE OF PAYMENT:
1
ENTER THE RATE PER HOUR:
320
PAYMENT FOR 50 HOURS IS Rs:16000.0
EMPLOYEE NAME=akshay
EMPLOYEE ID=22
PAYMENT TYPE=1
RATE PER HOUR:320.0
DEDUCTION RATE:3%
DEDUCTION AMOUNT: 480.0
AMOUNT AFTER TAXATION:15520.0
```

In this output we first enter the name ,id ,and no of hours you worked .After entering all the required details we calculate the salary according to the daily wages. As it is a daily waged salary 3% of the salary will be reduced due to taxes and the final salary would be displayed

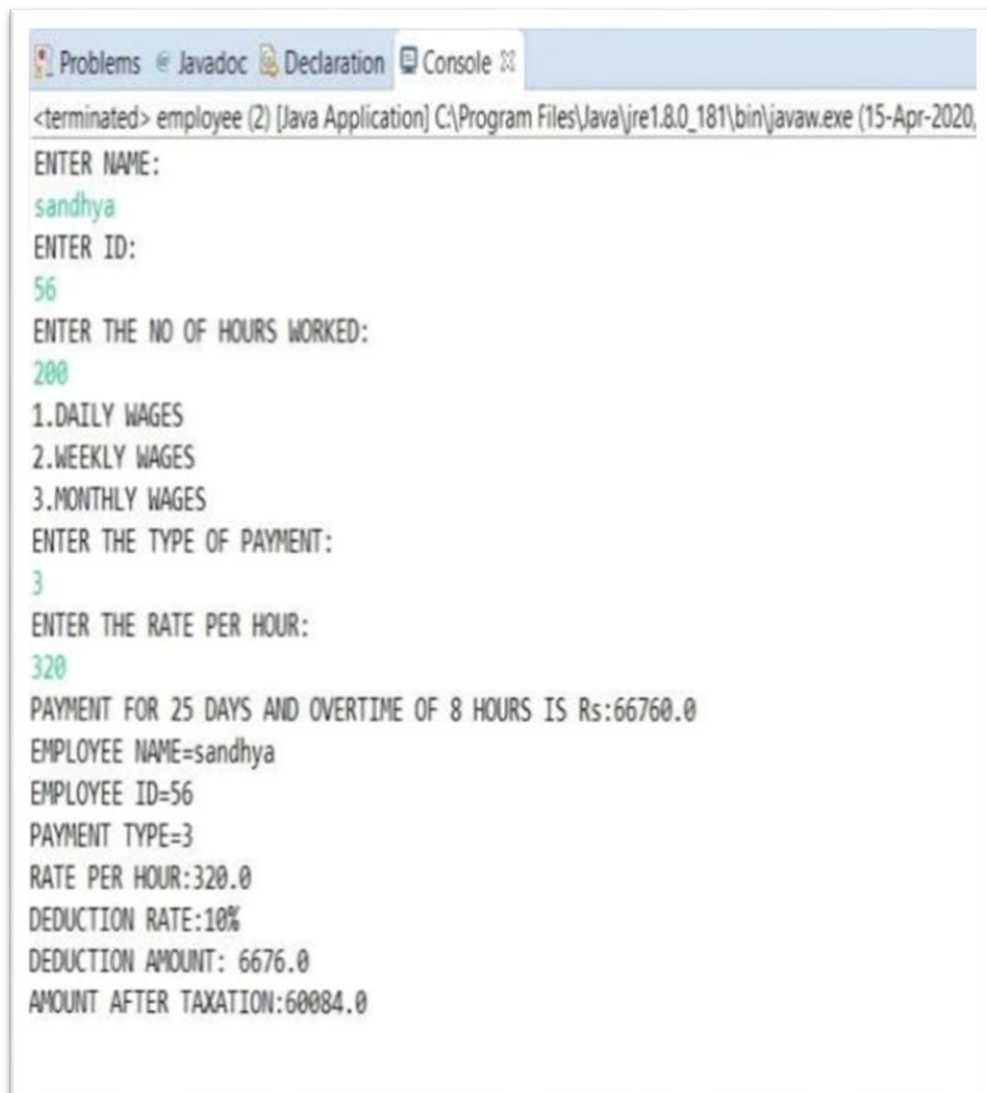
5.2 WEEKLY WAGES OUTPUT

In this output we first enter the name ,id ,and no of hours you worked .After entering all the required details we calculate the salary according to the weekly wages. As it is a weekly waged salary 5% of the salary will be reduced due to taxes and the final salary would be displayed.

```
<terminated> employee (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Apr-2020)
ENTER NAME:
sanjay
ENTER ID:
34
ENTER THE NO OF HOURS WORKED:
55
1.DAILY WAGES
2.WEEKLY WAGES
3.MONTHLY WAGES
ENTER THE TYPE OF PAYMENT:
2
ENTER THE RATE PER HOUR:
320
PAYMENT FOR 40 HOURS AND OVERTIME OF 15 HOURS IS Rs:22475.0
EMPLOYEE NAME=sanjay
EMPLOYEE ID=34
PAYMENT TYPE=2
RATE PER HOUR:320.0
DEDUCTION RATE:5%
DEDUCTION AMOUNT: 1123.75
AMOUNT AFTER TAXATION:21351.25
```

5.3 MONTHLY WAGES OUTPUT

In this output we first enter the name ,id ,and no of hours you worked .After entering all the required details we calculate the salary according to the monthly wages. As it is a monthly waged salary 10% of the salary will be reduced due to taxes and the final salary would be displayed.



```
<terminated> employee (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Apr-2020,
ENTER NAME:
sandhya
ENTER ID:
56
ENTER THE NO OF HOURS WORKED:
200
1.DAILY WAGES
2.WEEKLY WAGES
3.MONTHLY WAGES
ENTER THE TYPE OF PAYMENT:
3
ENTER THE RATE PER HOUR:
320
PAYMENT FOR 25 DAYS AND OVERTIME OF 8 HOURS IS Rs:66760.0
EMPLOYEE NAME=sandhya
EMPLOYEE ID=56
PAYMENT TYPE=3
RATE PER HOUR:320.0
DEDUCTION RATE:10%
DEDUCTION AMOUNT: 6676.0
AMOUNT AFTER TAXATION:60084.0
```

5.4 OUTPUT IF WRONG OPTION CHOSEN

In this output we first enter the name ,id ,and no of hours you worked .After entering all the required details it will ask for the type of payment when we give a number which is not there then then it will display that you have given an invalid option and the salary will not be calculated.



```
<terminated> employee (2) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15-Ar  
COMPANY NAME:PAYSCALE  
ADDRESS:BELLENDUR, BANGALORE  
PH:1234567890  
  
ENTER NAME:  
sandhya  
ENTER ID:  
2345  
  
ENTER THE NO OF HOURS WORKED:  
2543  
1.DAILY WAGES  
2.WEEKLY WAGES  
3.MONTHLY WAGES  
ENTER THE TYPE OF PAYMENT:  
4  
ENTER THE RATE PER HOUR:  
320  
INVALID OPTION  
CHOOSE THE CORRECT OPTION  
DEDUCTION RATE:0%  
DEDUCTION AMOUNT: 0.0  
AMOUNT AFTER TAXATION:0.0
```

5.5 OUTPUT WHEN NUMBER OF WORKING HOURS IS ZERO

In this output if the no of hours is zero there is no need of calculating the salary amount for the employee as the final result will be zero. The error message is displayed to the user and the application exits .

```
ENTER NAME:
vishal
ENTER ID:
56
ENTER THE NO OF HOURS WORKED:
0
INVALID OPERATION
```

5.6 OUTPUT COPIED FILE

The output is copied to the files to maintain the record of the payment made to the employees. The file contains the information regarding the employee name, id, mode of payment, deduction amount, deduction rate and salary amount after taxation.

mpfile.txt

```
EMPLOYEE NAME=sanjay
EMPLOYEE ID=1
PAYMENT TYPE=1
RATE PER HOUR:300.0
DEDUCTION RATE:3%
DEDUCTION AMOUNT: 333.0
AMOUNT AFTER TAXATION:10767.0
```

CHAPTER 6

CONCLUSION

The main objectives of calculating the salary amount for the employee is implemented successfully. The application is capable of calculating the salary amount accordingly the type of payment like daily, weekly and monthly wages. The salary amount is calculated on the base of the number of hours the employee have worked for the company or organization .

The rate per hour is used for the calculating the salary amount. The taxation of the salary is done and the deduction amount is calculated accordingly the type of payment. The deduction amount is subtracted from the total amount to generate the final salary amount.

All the necessary information about the employee and payment details of the employee is stored into a file for the documentation purposes.

REFERENCES

1. Herbert Schildt, Java™: The Complete Reference, McGraw-Hill, Tenth Edition, 2018
2. Cay S. Horstmann, Core Java SE 9 for the Impatient, Addison Wesley, Second Edition, 2013.
3. Cay S. Horstmann, Core Java™ Volume I—Fundamentals, Prentice Hall, Tenth Edition, 2015.
4. SAMS teach yourself Java – 2: 3rd Edition by Rogers Cedenhead and Leura Lemay Pub. Pearson Education.
5. Websites related to Java language and OOPs concepts.