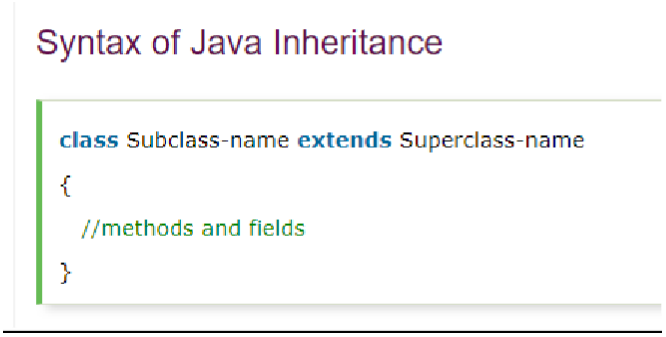
Ex. 5: CREATING APPLICATION USING INHERITANCE

Objectives:

The main objective of this experiment is to apply object oriented concepts, and build java application using Java's Inheritance Concepts such as Super class , Sub classes, Method Overriding, and Dynamic Method Dispatch(Polymorphism)

Concepts:

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.



Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String
 args[]){ Dog d=new Dog();
 d.bark();
 d.eat();
}
}
```

Method Overriding

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

Example:

```
class Bank{
int getRateOfInterest(){return 0;}
}
class SBI extends Bank{
int getRateOfInterest(){return 8;}
}

class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}

class Test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of Interest:
"+s.getRateOfInterest()); System.out.println("ICICI Rate of
Interest: "+i.getRateOfInterest()); System.out.println("AXIS Rate
of Interest: "+a.getRateOfInterest());
}
}
```

Dynamic Method Dispatch (Runtime polymorphism)

Dynamic Method Dispatch or Runtime polymorphism is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Upcasting

When reference variable of Parent class refers to the object of Child class, it is known as upcasting. For example:

```
class A{ }

class B extends A{ }


A a=new B();//upcasting
```

Example:

Java Runtime Polymorphism Example: Bank

Consider scenario; Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest.

```
class Bank{

float getRateOfInterest(){return 0;}

}

class SBI extends Bank{

float getRateOfInterest(){return 8.4f;}

}

class ICICI extends Bank{

float getRateOfInterest(){return 7.3f;}

}

class AXIS extends Bank{

float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Exp. No : 5

CREATING APPLICATION USING INHERITANCE

Choose your three questions based on the formulae given below,

Q1=((Regno%3)+1), Q2=((Regno%4)+4), Q3=((Regno%3)+8)

1 Develop a java application using Inheritance concept to automate the salary calculation of employee in an organization as per the salary band given below. Create a super class called Employee and derive sub classes as per the given table. Apply **method overriding (Run time Polymorphism)** to implement the following services via menu driven interface.

a) Calculate Gross Salary

b) Calculate Net Salary

c) Calculate Tax

d) Print the Pay Details

Salary Band	**Manager**	**Engineer**
Basic Salary	30000	20000
DA Pay	95%	80%
HRA	20%	15%
TAX	25%	15%
EPF	3000	2000

2 Develop a java application using Inheritance as per the following. Create a called Shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called Triangle and Rectangle from the super class Shape. In super class, add necessary setter/getter to initialize Shape class data members and another member function find_area() to compute and display the area of figures. Override the display_area() from sub classes and redefine the function to suit their requirements. Complete above scenario using Inheritance, Run Time Polymorphism with Menu driven options.

3 Develop a java application using Inheritance as per the following. Create a super class called CashTree which demonstrate any ATM Machine. Have attributes like name, codeno, location, etc. Add functionalities like ViewBalance, WithDraw and Deposit in super class. Derive two sub classes SBI_Bank, and HDFC_Bank with additional properties such as customer_name, balance and redefine those super class functionalities according to the nature of specified bank (differenciate via service_charges, interest rate, and maximum withdraw limit etc).

Complete the above scenario using Inheritance, Run Time Polymorphism with Menu driven options.

4 Develop a java application using Inheritance as per the following. Create a super class called Train which has attributes like name, trainno, source, destination, no.of tickets, cost. Add methods in such as Check_SeatAvailablity, BookTicket. Derive two sub classes such as ChennaiExpress and CoimbatoreExpress. Override all the super class functions in the derived classes. Implement the services such as check seat availability and booking ticket using run time polymorpshim with menu driven options. Initialize few train details using array of objects to make it those functions work for any source and destination (atleast five). Finally print the booked ticket details.

5 Consider the fuel rates in state wise as per the given table below. Create a Java application to calculate the **fuel cost consumed by the customer**. Create a super class called **FUEL** with few attributes such as fuel_name, state, cost. Add a function to calculate the cost for the fuel consumption required by customer. **Derive sub classes** as per the table given below. Provide an interface which prompts the user to give name, no. of litters required, etc. Finally print the name of the customer, state, no. of litter consumed and the total cost of the fuel. Complete the above scenario using Inheritance, Run Time Polymorphism with Menu driven options.

	TAMILNADU	**KERALA**	**KARNATAKA**
Petrol (per liter)	Rs.97.46	Rs. 98.35	Rs.99.61
Diesel (per liter)	Rs. 96.08	Rs.97.37	Rs.98.61
Kerosene (per liter)	Rs. 25.7	Rs.26.4	Rs.27.6
Auto LPG Gas (per kg)	Rs.70.33	Rs. 71.27	Rs. 72.08

6 The following INCOME TAX RATES are applicable for the Financial Year ending March 31, 2021. Create a Java application to provide a **tax calculation service** as per the tariff given below. Create a super class **IncomeTax** and have function called **findTax.** Derive three classes as per the category given table. Provide an menu driven interface for clients to enter the name, age, annual income and perform the tax calculation as per the given tariff. Complete the above scenario using Inheritance, Run Time Polymorphism with Menu driven options.

Income Range	**Men**	**Women**	**SeniorCitizen** (above age 60)
Upto Rs. 2,50,000	Nil	Nil	Nil

Rs. 2,50,001 to Rs. 3,00,000	10%	Nil	Nil
Rs. 3,00,001 to Rs. 5,00,000	10%	10%	5%
Rs. 5,00,001 to Rs. 10,00,000	20%	20%	10%
Above Rs. 10,00,000	30%	30%	20%

7	Write a menu driven application to maintain the student information using Java to demonstrate the concept of inheritance. Assume that you are considering PG and UG students, where the specialization details are maintained only for the PG students. Your application must contain the following functionalities along with the use of method overriding, and super keyword. a. For each student your application must have the details such as name, registerNo, department, specialization, cgpa, hostelName, mentorName,noOfArrears. b. Get the student details from user(admin) c. Display the student list with all details in a proper and neat format. d. In the menu give option to display the PG and UG student list separately.
8	Write a menu driven application to Library Management using Java to demonstrate the concept of Inheritance. Your application must contain the following functionalities along with the use of method overriding, and super keyword. a. The following details must be there for each Book i. bookTitle ii. bookAuthor iii. bookNoOfCopies iv. bookAvailability v.bookEdition vi. bookPublisher b. Get the Book details from librarian c. In the menu give options to borrow, return and Search options
9	Write a menu driven application to maintain the department details of a School

	using Java to demonstrate the concept of Inheritance. Your application must contain the following functionalities along with the use of method overriding, and super keyword. Consider the example of school of CST having three departments CSE, AI&SD and CE. a. For each department maintain the following details. i. deptName ii. hodName iii. noOfFaculty iv. noOfStudents v. noOfPrograms b. Get the department details from user(admin) c. Display the Department list with all details in a proper and neat format. d. In the menu give an option to display department wise details.
10	Develop a java application using Inheritance as per the following. Create a class Worker and derive two classes DailyWorker and SalariedWorker from it. Every worker has name, salary rate. Provide a method ComPay(int hours) to compute the week pay of every worker. A DailyWorker is paid on the basis of number of days he/she works. The SalariedWorker gets paid the wage for 40 hours a week no matter what actual hours is. Implement this scenario to calculate the pay of workers. You are expected to use the concept of Inheritance with proper run time polymorphism