**Lab Session 05**

**Apply the concepts of Inheritance and types of Inheritance**

**Date of the Session: Time of the Session:**

**Pre-Lab Tasks:**

1. **How will you inherit a class?**

**2. If the members of super class are declared as private, will it be accessible by subclass?**

**3.Can a subclass call the constructor defined by its superclass.**

**4. Mention the use of keyword super in inheritance.**

**5. What is multiple inheritance and how it is achieved in Java?.**

**In-Lab Tasks:**

1. Create a class called “Automobile” which holds the following data members:

Make

Year/ Model (two pieces to be stored as one string)

Mileage

Price.

Write an automobile base class with cars, Trucks and Vans as child classes, complete with parameter constructors which pass the parents member variables to the child class.

**Test Cases:**

***Test Case 1:***

***Input:***

“Maruthi”, “2019-Swift”, 23.5, 100000

***Output:***

Make=Maruthi

Year/Model=2019-Swift

Milage=23.5

Price=100000

***Test Case 2:***

***Input:***

“Hundai”, “i10-2020”, 20000

***Output:***

Invalid Input parameter for Year/Model [since first four characters in the range of 2000-2030]

**Aim:** Write a Java program which holds the details of Automobiles using inheritance.

**Algorithm:**

1. Start.
2. Create a class named Automobile with attributes for make, year/model, mileage, and price. Include a toString method to display the details of the automobile.
3. Create subclasses Car, Truck, and Van, which inherit from the Automobile class. Each subclass should have a constructor that calls the superclass constructor using super.
4. In the Vehicle class (main class):

a. Create a Scanner object to read user input.

b. Prompt the user to choose the type of vehicle (1 for Car, 2 for Truck, 3 for Van).

c. Read the user's choice.

d. Prompt the user to enter the make of the vehicle.

e. Prompt the user to enter the year/model (as an integer).

f. Check if the entered year/model is within a valid range (e.g., between 2001 and 2029). If not, display an error message and exit.

g. Prompt the user to enter the mileage of the vehicle.

h. Prompt the user to enter the price of the vehicle.

i. Based on the user's choice of vehicle type, create an object of the corresponding class (Car, Truck, or Van) with the provided information.

j. Display the details of the vehicle using the toString method of the created object.

1. Stop.

**Source Code:**

**Vehicle.java**

import java.util.\*;

class Automobile

{

String make;

String Year\_Model;

double milage;

double price;

Automobile(String m,String y,double mge,double p)

{

make=m;

Year\_Model=y;

milage=mge;

price=p;

}

public String toString()

return "make: "+make+"\n"+"Year model: "+Year\_Model+"\n"+"milage: "+milage+"\n"+"price: "+price;

}

class car extends Automobile

{

car(String m,String y,double mge,double p)

super(m,y,mge,p);

}

class truck extends Automobile

{

truck(String m,String y,double mge,double p)

super(m,y,mge,p);

}

class van extends Automobile

{

van(String m,String y,double mge,double p)

super(m,y,mge,p);

}

class Vehicle

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter 1.car 2.truck 3.van: ");

int choice=sc.nextInt();

System.out.println("enter make: ");

String m=sc.next();

System.out.println("enter Year/model");

String y=sc.next();

System.out.println("enter milage: ");

double mge=sc.nextDouble();

System.out.println("enter price: ");

double p=sc.nextDouble();

int k=Integer.parseInt(y.substring(0,4));

if(k<=2000 || k>=2030)

{

System.out.println("Invalid year/model..");

return;

}

switch(choice)

{

case 1:

car c=new car(m,y,mge,p);

System.out.println(c);

break;

case 2:

truck t=new truck(m,y,mge,p);

System.out.println(t);

break;

case 3:

van v=new van(m,y,mge,p);

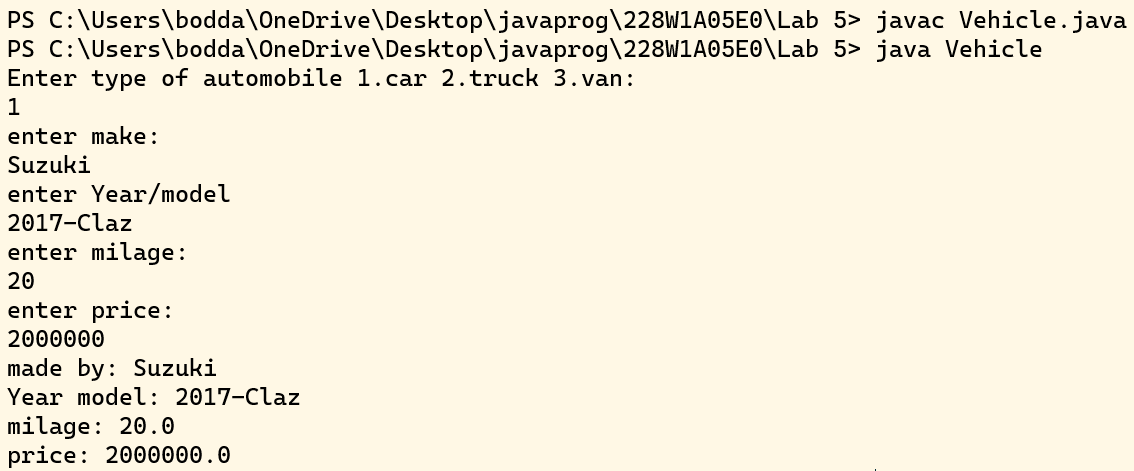
System.out.println(v);

break;

}

}

**Output:**

****

**Result:** Thus the java program which holds the details of Automobiles using inheritance is successfully executed.

1. All the banks operating in India are controlled by RBI. RBI has set a well-defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use 4% interest rate or to set any rates above it.

Write a Java program to implement bank functionality in the above scenario and demonstrate the dynamic polymorphism concept. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes (SBI, ICICI, PNB etc). Assume and implement required member variables and functions in each class.

**Test Cases:**

***Test Case 1:***

***Input:***

Bank\_Name=”SBH”

deposit\_amount=10000

No\_of\_Months=4

Withdrawal\_amount=0

***Output:***

10400

***Test Case 2:***

***Input:***

Bank\_Name=”ICICI”

deposit\_amount=10000

No\_of\_Months=0

Withdrawal\_amount=1000

***Output:***

“maintain Minimum balance”

**Aim:** Write a Java program to implement bank functionality and demonstrate polymorphism concept.

**Algorithm:**

1. Start
2. Create a class Customer with instance variables bankName, depositAmount, and noOfMonths.
3. Create a constructor for the Customer class that takes bankName, depositAmount, and noOfMonths as parameters and initializes the instance variables.
4. Create a method calculateInterest in the Customer class that takes an object of the RBI class (or its subclasses) as a parameter.
5. Inside the calculateInterest method, call the calculateInterest method of the RBI object passed as a parameter, passing in the depositAmount and noOfMonths.
6. Create a class RBI with a method calculateInterest that takes depositAmount and noOfMonths as parameters and returns 0.0.
7. Create a subclass SBI of RBI.
8. Override the calculateInterest method in the SBI class to calculate interest based on the SBI interest
9. Create a subclass ICICI of RBI.
10. Override the calculateInterest method in the ICICI class to calculate interest based on the ICICI interest rate formula.
11. Create a Bank class with a main method.
12. Inside the main method, create a Customer object customer1 with bank name "SBH," a deposit amount of 10000, and 4 months.
13. Create an RBI object bank1 and initialize it with an SBI object.
14. Call the calculateInterest method of customer1 with bank1 as a parameter and print the result.
15. Stop.

**Source Code:**

**Bank.java**

class Customer

{

String bankName;

double depositAmount;

int noOfMonths;

public Customer(String bankName1, double depositAmount1, int noOfMonths1)

{

bankName = bankName1;

depositAmount = depositAmount1;

noOfMonths = noOfMonths1;

}

public double calculateInterest(RBI bank)

{

return bank.calculateInterest(depositAmount, noOfMonths);

}

}

class RBI

{

public double calculateInterest(double depositAmount,int noOfmonth)

{

return 0.0;

}

}

class SBI extends RBI

{

public double calculateInterest(double depositAmount,int noOfmonth)

{

double interestRate = 4.0;

return depositAmount \* interestRate / 100 \* noOfMonths + depositAmount;

}

}

class ICICI extends RBI

{

public double calculateInterest(double depositAmount,int noOfmonth)

{

double interestRate = 3.5; // ICICI interest rate

return depositAmount \* interestRate / 100 \* noOfMonths + depositAmount;

}

}

public class Bank

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.print(“Enter bank name:”);

String bank=sc.next();

System.out.print(“Enter deposit amount:”);

double depo=sc.nextDouble();

System.out.print(“No.of months:”);

int mon=sc.nextInt();

Customer customer1 = new Customer(bank,depo,mon);

RBI bank1 = new SBI();

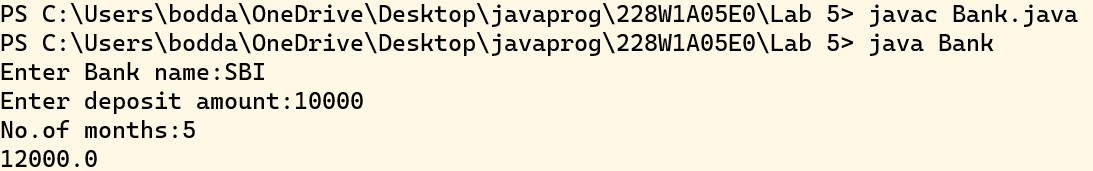
System.out.println(customer1.calculateInterest(bank1));

}

}

**Output:**

****

****

**Result:** Java program to implement bank functionality is successfully executed.

**Post-Lab Tasks:**

1. Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

**Test Cases:**

***Test Case 1:***

***Input:***

call print method using each class object

***Output:***

‘Get Degree’

‘Undergraduate’

‘Postgraduate’

***Test Case 2:***

***Input:***

Call parent class print method using any one of the child class object

***Output:***

‘Get Degree’

‘Postgraduate’

***Test Case 3:***

***Input:***

Call any child t class print method using parent class object

***Output:***

Exception/ Error

**Aim:** Write a java program to print the completed degrees of a student

**Algorithm:**

1. Start
2. Create a class named Degree with a method getDegree that prints "I got a degree."
3. Create a subclass named Undergraduate that extends the Degree class.
4. In the Undergraduate class, override the getDegree method to print "I am an Undergraduate."
5. Create another subclass named Postgraduate that also extends the Degree class.
6. In the Postgraduate class, override the getDegree method to print "I am a Postgraduate."
7. Create a Main class for testing.
8. In the Main class's main method:

Create an object of the Degree class called degree. Create an object of the Undergraduate class called undergraduate. Create an object of the Postgraduate class called postgraduate. Call the getDegree method on each of these objects.

1. Stop.

**Source Code:**

**Main.java**

class Degree

{

void getDegree()

{

System.out.println("I got a degree");

}

}

class Undergraduate extends Degree

{

void getDegree()

{

System.out.println("I am an Undergraduate");

}

}

class Postgraduate extends Degree

{

void getDegree()

{

System.out.println("I am a Postgraduate");

}

}

public class Main

{

public static void main(String[] args)

{

Degree degree = new Degree();

Undergraduate undergraduate = new Undergraduate();

Postgraduate postgraduate = new Postgraduate();

degree.getDegree();

undergraduate.getDegree();

postgraduate.getDegree();

System.out.println();

}

}

**Output:**

****

**Result:** Java program that prints the completed degrees of a student is successfully executed.

1. Create a class 'Student' with three data members which are name, age and address. The constructor of the class assigns default values name as "unknown", age as '0' and address as "not available". It has two members with the same name 'setInfo'. First method has two parameters for name and age and assigns the same whereas the second method takes has three parameters which are assigned to name, age and address respectively. Print the name, age and address of 3 students. **Hint -** Use array of objects

**Test Cases:**

***Test Case 1:***

***Input:***

‘john’, 10, ‘Vijayawada’, ’San’ ,20, ’Guntur’, ’Ram’,30, ’Hyderabad’

***Output:***

Student1.name=’john’

Student1.age=10

Student1.address=’Vijayawada’

Student2.name=’San’

Student2.age=20

Student2.address=’Guntur’

Student3.name=’Ram’

Student3.age=30

Student3.address=’Hyderabad’

***Test Case 2:***

***Input:***‘venkat’, 90, ‘Vijayawada’, ’Rahim’ ,20, ’Guntur’, ’Dileep’,30, ’Hyderabad’,’Venu’,40,’Vizag’

***Output:*** Array index Out of Bound Exception

**Aim:** Write a java program and Create a class 'Student' with three data members which are name, age and address using array of objects.

**Algorithm:**

1. Start
2. Define a class named 'Student' with three private data members: 'name', 'age', and 'address'.
3. Create a default constructor for the 'Student' class:
4. Initialize 'name' to "unknown". Initialize 'age' to 0, Initialize 'address' to "not available".
5. Create two 'setInfo' methods within the 'Student' class: a. First 'setInfo' method with two parameters (name and age):
6. Set the 'name' and 'age' data members to the provided values.
7. b. Second 'setInfo' method with three parameters (name, age, and address):
8. Set the 'name', 'age', and 'address' data members to the provided values.
9. Create getter methods within the 'Student' class to access the student's name, age, and address.
10. In the main class: a. Create an array of 'Student' objects, e.g., 'students'. b. Initialize each element of the 'students' array with a 'Student' object.
11. For each student in the 'students' array: a. Call the appropriate 'setInfo' method to set the student's information. b. Use the getter methods to retrieve the student's name, age, and address. c. Display the student's information in the specified format.
12. Handle any exceptions (e.g., ArrayIndexOutOfBoundsException) that may occur when trying to access elements of the 'students' array outside its bounds.
13. Stop

**Source Code:**

**Array.java**

class Student

{

String name;

int age;

String address;

Student()

{

name = "unknown";

age = 0;

address = "not available";

}

void setInfo(String name1, int age1)

{

name = name1;

age = age1;

}

void setInfo(String name1, int age1, String address1)

{

name = name1;

age = age1;

address = address1;

}

}

public class Array

{

public static void main(String[] args)

{

Student[] students1 = new Student[3];

students1[0] = new Student();

students1[1] = new Student();

students1[2] = new Student();

students1[0].setInfo("john", 10, "Vijayawada");

students1[1].setInfo("San", 20, "Guntur");

students1[2].setInfo("Ram", 30, "Hyderabad");

for (int i = 0; i < students1.length; i++)

{

System.out.println("Student"+(i+1)+" name="+students1[i].name);

System.out.println("Student"+(i+1)+" age="+students1[i].age);

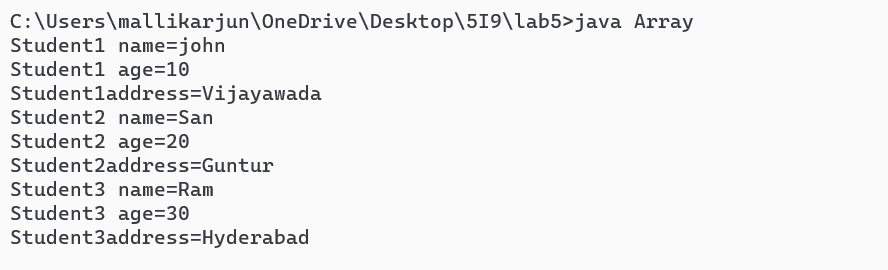
System.out.println("Student"+(i+1)+"address="+students1[i].address);

}

}

}

**Output:**

****

**Result:** Thus a java program to create a class 'Student' with three data members which are name, age and address using array of objects is successfully executed.

**Student’s Signature**

***(For Evaluator’s use only)***

**Marks Secured: \_\_\_\_\_\_\_\_\_ out of \_\_\_\_\_\_\_\_\_\_**

**Faculty Signature**