**JAVA PROGRAMMING**

**LAB (ETCS – 357)**

**LAB-5**

**Faculty Name : Student Name:**

Dr. Sandeep Tayal Udit Agarwal

12914802719

Sem 5

5C7



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

**INDEX**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **Experiments** | **Date of**  **performance** | **Date checked** | **Marks** | | | | | **Total Marks** | **Sign.** |
| **R1** | **R2** | **R3** | **R4** | **R5** |
| 5.1 | WAP that implements method overriding | 15.10.21 |  |  |  |  |  |  |  |  |
| 5.2 | WAP to illustrate simple inheritance. | 15.10.21 |  |  |  |  |  |  |  |  |
| 5.3 | WAP to illustrate multilevel inheritance | 15.10.21 |  |  |  |  |  |  |  |  |
| 5.4 | WAP illustrating all uses of super keywords | 15.10.21 |  |  |  |  |  |  |  |  |

**EXPERIMENT 5.1**

**AIM : WAP that implements method overriding**

**Theory :**

**Class :** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**Method Overriding :** Declaring a method in sub class which is already present in parent class is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method.

The main advantage of method overriding is that the class can give its own specific implementation to a inherited method **without even modifying the parent class code**.

This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it and the other classes that want to have different implementation can use overriding feature to make changes without touching the parent class code.

**Source Code :**

class Rectangle {

// Original Method

public void show() {

System.out.println("This is Rectangle !");

}

}

class Circle extends Rectangle {

// Overriding method

public void show() {

System.out.println("This is Cirlce !");

}

public static void main(String args[]) {

Circle obj = new Circle();

obj.show();

}

}

**Output :**

****

**EXPERIMENT 5.2**

**AIM : WAP to illustrate simple inheritance.**

**Theory :**

**Class :** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**Inheritance :** Inheritance is the basic properties of object-oriented programming. Inheritance tends to make use of the properties of a class object into another object. Java uses inheritance for the purpose of code-reusability to reduce time by then enhancing reliability and to achieve run time polymorphism. As the codes are reused, it makes less development cost and maintenance.

**Simple Inheritance :** Single inheritance specifies child-parent class relationships when they extend and the simplest type of all the methods, such as pear and apple inheriting from the fruits. In Inheritance mechanisms, objects are treated in a top-down manner. Previously we learned about syntax and its declaration.  To go with it is necessary to read the concept of access specifiers, namely private, public, protected.

**Source Code :**

Public class Shape {

int length;

int breadth;

}

Public class Rectangle extends Shape {

int area;

public int calcualteArea() {

return length \* breadth;

}

public static void main(String args[]) {

Rectangle obj = new Rectangle();

// Assigning values to Shape class attributes

obj.breadth = 10;

obj.length = 5;

// Calculate the area

System.out.println("The Area is : " + obj.calcualteArea());

}

}

**Output :**

****

**EXPERIMENT 5.3**

**AIM : WAP to illustrate multilevel inheritance**

**Theory :**

**Class :** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**Inheritance :** Inheritance is the basic properties of object-oriented programming. Inheritance tends to make use of the properties of a class object into another object. Java uses inheritance for the purpose of code-reusability to reduce time by then enhancing reliability and to achieve run time polymorphism. As the codes are reused, it makes less development cost and maintenance.

**Multiple Inheritance :** In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent’s members.

**Source Code :**

class Mobile {

public Mobile() {

System.out.println("Mobile");

}

public void deviceType() {

System.out.println("Device Type: Android");

}

}

class Xiaomi extends Mobile {

public Xiaomi() {

System.out.println("Class Xiaomi");

}

public void brand() {

System.out.println("Brand: Xiaomi");

}

public void priceRange() {

System.out.println("Price Range : 5000-80000");

}

}

public class Redmi extends Xiaomi {

public Redmi() {

System.out.println("Xiaomi SubBrand : Redmi");

}

public void priceRange() {

System.out.println("Price Range : 5000-50000");

}

public static void main(String args[]) {

Redmi obj = new Redmi();

obj.deviceType();

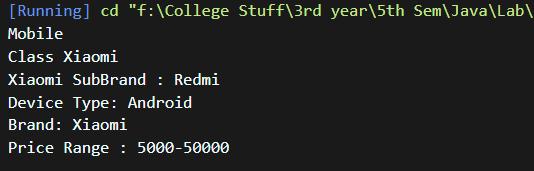
obj.brand();

obj.priceRange();

}

}

**Output :**

****

**EXPERIMENT 5.4**

**AIM : WAP illustrating all uses of super keywords**

**Theory :**

**Class :** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**Super Keyword :** The super keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

**Usage of Java super Keyword**

* **super can be used to refer immediate parent class instance variable** : We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.
* **super can be used to invoke immediate parent class method :** The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.
* **super() can be used to invoke immediate parent class constructor :** The super keyword can also be used to invoke the parent class constructor.

**Source Code :**

1. **Using to access immediate parent class variables.**

class A {

int score = 98;

}

class B extends A {

int score = 81;

void display() {

System.out.println("Score : " + super.score);

}

}

class use1 {

public static void main(String[] args) {

B obj = new B();

obj.display();

}

}

1. **Using to call immediate parent class method.**

class X

{

void displayClass()

{

System.out.println("This is class X");

}

}

class Y extends X

{

void displayClass()

{

System.out.println("This is class Y");

}

void display()

{

displayClass();

super.displayClass();

}

}

class use2

{

public static void main(String args[])

{

Y obj = new Y();

obj.display();

}

}

1. **Using to call immediate parent class constructor.**

class Company

{

Company()

{

System.out.println("Company class Constructor");

}

}

class Employee extends Company

{

Employee()

{

super();

System.out.println("Employee class Constructor");

}

}

class use3

{

public static void main(String[] args)

{

Employee obj = new Employee();

}

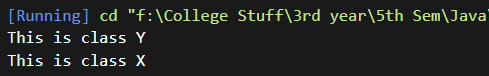
}

**Output :**

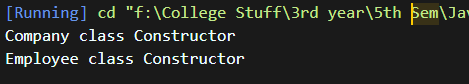
1. **Using to refer immediate parent class instance variable.**



1. **Using to invoke immediate parent class method.**

****

1. **Using to invoke immediate parent class constructor**

****

**VIVA QUESTIONS :**

**Q.1 Why do we need to use inheritance?**

**Ans.** Inheritance is one of the main pillars of OOPs concept. Some objects share certain properties and behaviors. By using inheritance, a child class acquires all properties and behaviors of parent class.

**Q.2 What is super class and subclass?**

**Ans.** Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent class. A class that inherits all the members (fields, method, and nested classes) from other class is called subclass. It is also called a derived class, child class, or extended class.

**Q3 How is Inheritance implemented/achieved in Java?**

**Ans.** Inheritance can be implemented or achieved by using two keywords:

* Extends : extends is a keyword that is used for developing the inheritance between two classes and two interfaces.
* Implements : implements keyword is used for developing the inheritance between a class and interface.

**Q4. Which class in Java is superclass of every other class?**

**Ans** In Java, Object class is the superclass of every other class.

**Q5. What is single inheritance and multi-level inheritance?**

**Ans:** When one class is extended by only one class, it is called single level inheritance. In single-level inheritance, we have just one base class and one derived class. A class which is extended by a class and that class is extended by another class forming chain inheritance is called multilevel inheritance.

**Q6. Why multiple inheritance is not supported in java through class?**

**Ans:** Multiple inheritance means that one class extends two superclasses or base classes but in Java, one class cannot extend more than one class simultaneously. At most, one class can extend only one class. Therefore, to reduce ambiguity, complexity, and confusion, Java does not support multiple inheritance through classes.

**Q7. What is super keyword in java?**

**Ans:** Super keyword in java is a reference variable which is refers immediate parent or super class object. In other words, Super keyword is used to access the members of parent class.

**Q7. What is method overriding?**

**Ans.** Modifying a super class method in the sub class is called method overriding. Using method overriding, we can change super class method according to the requirements of sub class.

**Q8. Can we override static methods?**

**Ans.** No, Static methods can not be overridden. If we try to override them they will be hidden in the sub class.