JAVA

Class 26

Agenda

Method Overriding vs Method Overloading
This and Super keyword in Java

Overloading	Overriding
Whenever same method or Constructor is existing multiple times within a class either with different number of parameter or with different type of parameter or with different order of parameter is known as Overloading.	Whenever same method name is existing multiple time in both base and derived class with same number of parameter or same type of parameter or same order of parameters is known as Overriding.
Arguments of method must be different.	Argument of method must be same including order.
Method signature must be different.	Method signature must be same.
Private, static and final methods can be overloaded.	Private, static and final methods can not be override.
Access modifiers point of view no restriction.	Access modifiers point of view not reduced scope of Access modifiers but increased.
Also known as compile time polymorphism or static polymorphism or early binding.	Also known as run time polymorphism or dynamic polymorphism or late binding.
Overloading can be exhibited both are method and constructor level.	Overriding can be exhibited only at method label.
The scope of overloading is within the class.	The scope of Overriding is base class and derived class.
Overloading can be done at both static and non-static methods.	Overriding can be done only at non-static method.
For overloading methods return type may or may not be same.	For overriding method return type should be same.

Task

1. A boy has his money deposited $1000, $1500 and $2000 in banks-Bank A, Bank B and Bank C respectively. We have to print the money deposited by him in a particular bank. Create a class 'Bank' with a method 'getBalance' which returns 0. Make three subclasses named 'BankA', 'BankB' and 'BankC' with a method with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the method 'getBalance' by the object of each of the three banks.

2. Write program in class A has final display method and try overload and override this method and observe result .

This Keyword

this is a keyword in Java.

this keyword in java can be used inside the Method or Constructor of Class.

this is a reference variable that refers to the current object.

this is used to access the members of the same class

This Keyword

Usage of java ***this*** keyword.

- ***this*** can be used to refer current class instance variable.
- ***this*** can be used to invoke current class method (implicitly).
- ***this***() can be used to invoke the current class constructor.

This Keyword

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}

class TestThis2{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

This Keyword

```
class Message{
    Message(){
        System.out.println("hello a");
    }

    Message(int p){
        this();
        System.out.println(""p);
    }
}

class ThisKeyword{
    public static void main(String args[]){
    Message a=new Message(12);
    }
}
```

Task

```
public class Constructors {

	Constructors(){
		this(1);
		System.out.println("Hi");
	}
	Constructors(int x){
		this(1,2);
		System.out.println("Hello");
	}
	Constructors(int x,int y){
		System.out.println("How are you");
	}

	public static void main(String[] args) {
		Constructors obj=new Constructors();
	}
}
```

Task

1. Write program as a student class that has instance variables name and address. Create a constructor that will initialize those variables. Print name & address of given student by the displayInfo method.

2. Create a class in which you will have overloaded constructors. Create and instance of the class that will execute both constructors.

Super Keyword

super keyword in java is a reference variable which is used to refer immediate parent class object.

Usage of Java super Keyword:

- ***super*** can be used to refer immediate parent class instance variable.
- ***super*** can be used to invoke immediate parent class method.
- ***super()*** can be used to invoke immediate parent class constructor.

Super Keyword

super is used to refer immediate parent class instance variable.

We can use ***super*** keyword to access the data member or field of parent class if parent class and child class have members with same name.

Super Keyword

```
class Furniture{
        String color="Red";
}

class Chair extends Furniture{
        String color="black";
        void printColor(){
        System.out.println(color);//prints color of Chair class
        System.out.println(super.color);//prints color of Furniture class
        }
}
class MainClass{
        public static void main(String args[]){
        Chair d=new Chair();
        d.printColor();
        }
}
```

Super Keyword

super can be used to invoke parent class method

The ***super*** keyword also used to access the method of parent class when child class has overridden that method.

Super Keyword

```
class Furniture{
    void color(){
        System.out.println(" Furniture color...");
        }
}
class Chair extends Furniture{
    void color(){
        System.out.println(" Chair color...");
        }

    void height(){
        System.out.println(" 5 ft  ...");
        }

    void work(){
      super.color();
      height();
    }
}

class MainClass{
    public static void main(String args[]){
    Chair obj=new Chair();
    obj.work();
    }
}
```

Super Keyword

super is used to invoke parent class constructor.

```
class Furniture{
      Furniture(){
            System.out.println("Furniture class Constructor");
            }
}

class Chair extends Furniture{
      Chair(){
      super();
      System.out.println("Chair class Constructor");
      }
}

class MainClass{
      public static void main(String args[]){
      Chair d=new Chair();
      }
}
```

Task

1. Write program: user class has a constructor that takes name and mobile number. Extend this class by userInfo that will have user address variable. Print users name, mobile number and address in userDetails method. Test your code.

2. Write program: Shape class has a constructor that takes the radius and extend circle class. Print area of circle while creating and Object of a child class.

this vs super

this is used to refer **current-class** instance as well as static members

super is used to refer **super-class's** instance as well as static members.

NOTE: We can use this as well as super anywhere except static area. Example of this is already shown above where we use this as well as super inside public static void main(String[]args) hence we get Compile Time Error since cannot use them inside static area.

this vs this()

THIS	THIS()
Used with variables and methods	Used with constructors only
One of the uses is differentiate between local and instance variables in a method call	Used to call one constructor from another belonging to the same class

super vs super()

SUPER	SUPER()
super is used to call super class variables and methods by the subclass object when they are overridden by subclass.	super() is used to call super class constructor from subclass constructor.

JAVA

Class 27

Agenda

Abstraction in Java
Abstract Class in Java

Abstraction

Abstraction is the process of separating ideas from their action.

Abstraction is a process **of hiding the implementation details and showing only functionality to the user**.

For example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction

Real Life Example of Abstraction

Abstraction shows only important things to the user and hides the internal details for example when we ride a bike, we only know about how to ride bike but can not know about how it work ? and also we do not know internal functionality of bike.

Real Life Example of Abstraction

Abstraction

Likewise in Object-oriented programming, **abstraction is a process of hiding the implementation details from the user**, only the functionality will be visible to the user.

Abstraction lets you focus on what the object does instead of how it does it i.e the user will have the information on what the object does instead of how it does it.

Note: Data abstraction can be used to provide security for the data from the unauthorized methods.

Abstraction

Abstraction is nothing but the process of giving an set of guidelines to the user, who are going to implement these guidelines based on their needs.

Abstraction is the concept of exposing only the required essential characteristics and behavior with respect to a context.

Hiding of data is known as **data abstraction**. In object oriented programming language this is implemented automatically while writing the code in the form of class and object.

How to achieve Abstraction ?

There are two ways to achieve abstraction in java:

- Abstract class (0 to 100% or partial abstraction)
- Interface (Achieve 100% abstraction or full abstraction)

Class Type

In java programming we have two types of classes:

Abstract class

Concrete class

An **abstract class** is one which is declared with abstract keyword and can contain defined methods and undefined method.

Abstract Class

Example of abstract class

```
abstract class Vehicle {
    abstract void speed();  // abstract method
}

class Bike extends Vehicle {
    void speed(){
    SOP("Speed limit is 40 km/hr..");
 }
 public static void main(String args[]) {
   Vehicle obj = new Bike(); //indirect obj creation
    obj.speed();
  }
}
```

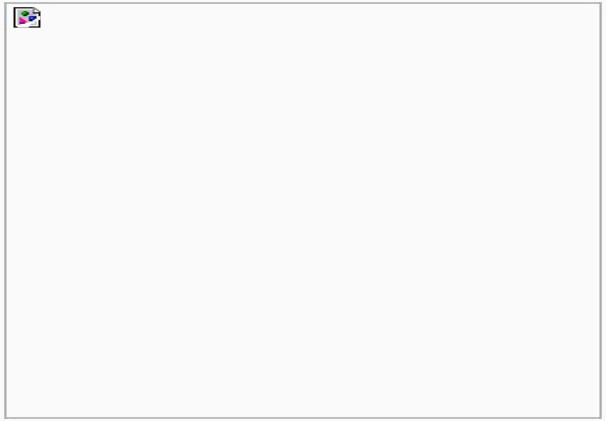
Abstract class in Java

1. Class that is declared with abstract keyword, is known as abstract class. An abstract class is one which is containing some defined method and some undefined method.
2. Class is declared abstract, it cannot be instantiated/ cannot create an object of it.
3. In Java programming undefined methods are known as un-Implemented or abstract method.
4. To utilize an abstract class, we have to inherit it from another class and provide implementations for the abstract methods in it. (It needs to be extended and its method implemented.)
5. If we inherit an abstract class, we have to provide implementations for all the abstract methods in it.

Abstract class in Java

Syntax:	Example:	If any class have any abstract method then that class become an abstract class.

abstract class
 	
className {
.....
}
	abstract class A {
	.....
	}
		Example:
		class Vehicle {
		abstract void Bike();
		}



Make method as abstract method

An abstract method is one which contains only declaration or prototype but it never contains body or definition. In order to make any undefined method as abstract whose declaration is must be predefined by abstract keyword.

Syntax

abstract ReturnType methodName(List of formal parameter)

Example

```
abstract   void  sum();
abstract  void  diff(int, int);
```

Concrete Class

A concrete class is a class that has an implementation for all of its methods that were inherited from abstract or implemented via interfaces. It also does not define any abstract methods of its own.

```
public abstract class AbstractClass {
	public abstract void abstractMethod();
}

class ConcreteClass extends AbstractClass {

	@Override
	public void abstractMethod() {
		System.out.println("Providing an implementation
of for an abstract method");
	}
}
```

Abstract Class

Create an Object of abstract class

An object of abstract class can not be created directly but it can be created indirectly. It means you can create an object of abstract derived class. You can see in above example Example

Vehicle obj = new Bike(); //indirect object creation

If you are extending any abstract class that have abstract method, you must either provide the implementation of the method or make this class abstract.

Advantage of Abstract class

Abstraction is one of the fundamental principles of Object Oriented Programming languages.

Abstraction helps to reduce the complexity and also improves the maintainability of the system.

Combining abstraction with the concepts of the Encapsulation and Polymorphism, it gives more power to the Object oriented programming languages.

Important Points about abstract class

We use abstract classes when we know the methods, but we don’t know how they would be implemented.

Every abstract class participate in inheritance.

Abstract class of java always contains common features.

Abstract classes definitions should not be made as final because abstract classes always participate in inheritance classes.

An object of abstract class can not be created directly but it can be created indirectly.

Implementation of Abstract Class

```
abstract class Vehicle {
    abstract void speed();
    void mileage() {
          SOP("Mileage is 60 km");
     }
}

class Bike extends Vehicle {
      void speed() {
            SOP("Speed limit is 40 km/hr..");
       }

 public static void main(String args[]) {
     Vehicle obj = new Bike();
     obj.speed();
     obj.mileage();
 }
}
```

Output

```
Speed limit is 40 km/hr..
Mileage is 60 km/ltr..
```