Top of Form

##### **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.

A class in Java can contain:

* Fields
* Methods
* Constructors
* Blocks (Instance Initialization Block (IIB), Static Initialization Block (SIB))
* Nested class and interface

**Syntax to declare a class:**

**class <class\_name>{**

**field;**

**method;**

**}**

##### **Static Class:**

You cannot use the static keyword with a class unless it is an inner class. A static inner class is a nested class which is a static member of the outer class. It can be accessed without instantiating the outer class, using other static members. Just like static members, a static nested class has not have access to the instance variables and methods of the outer class.

###### Syntax

**class MyOuter {**

**static class Nested\_Demo {**

**}**

**}**

Instantiating a static nested class is a bit different from instantiating an inner class. The following program shows how to use a static nested class.

**EXAMPLE:**

**public class Outer {**

**Java Arrays with Answers**

**static class Nested\_Demo {**

**public void my\_method() {**

**System.out.println("This is my nested class");**

**}**

**}**

**public static void main(String args[]) {**

**Outer.Nested\_Demo nested = new Outer.Nested\_Demo();**

**nested.my\_method();**

**}**

**}**

**Can a class be static in Java?**

The answer is Yes, some classes can be made static in Java. Java supports

* Static Instance Variables
* Static Methods,
* Static Block
* Static Classes

Java allows a class to be defined within another class. These are called **Nested Classes**. The class in which the nested class is defined is known as the **Outer Class**. Unlike top-level classes, **Inner classes can be Static**. Non-static nested classes are also known as **Inner classes**.

An instance of an inner class cannot be created without an instance of the outer class. Therefore, an inner class instance can access all of the members of its outer class, without using a reference to the outer class instance. For this reason, inner classes can help make programs simple and concise.

##### **Java Inner Classes:**

In Java, it is also possible to nest classes (a class within a class). The purpose of nested classes is to group classes that belong together, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class.

Example:

**class OuterClass {**

**int x = 10;**

**class InnerClass {**

**int y = 5;**

**}**

**}**

**public class MyMainClass {**

**public static void main(String[] args) {**

**OuterClass myOuter = new OuterClass();**

**OuterClass.InnerClass myInner = myOuter.new InnerClass();**

**System.out.println(myInner.y + myOuter.x);**

###### }

**}**

###### Private Inner Class:

Unlike a "regular" class, an inner class can be private or protected. If you don't want outside objects to access the inner class, declare the class as private:

###### Example:

**class OuterClass {**

**int x = 10;**

**private class InnerClass {**

**int y = 5;**

**}**

**}**

**public class MyMainClass {**

**public static void main(String[] args) {**

**OuterClass myOuter = new OuterClass();**

**OuterClass.InnerClass myInner = myOuter.new InnerClass();**

**System.out.println(myInner.y + myOuter.x);**

**}**

**}**

###### Static Inner Class:

An inner class can also be static, which means that you can access it without creating an object of the outer class:

###### Example:

**class OuterClass {**

**int x = 10;**

**static class InnerClass {**

**int y = 5;**

**}**

**}**

**public class MyMainClass {**

**public static void main(String[] args) {**

**OuterClass.InnerClass myInner = new OuterClass.InnerClass();**

**System.out.println(myInner.y);**

**}**

**}**

###### Access Outer Class From Inner Class:

One advantage of inner classes, is that they can access attributes and methods of the outer class:

###### Example

**class OuterClass {**

**int x = 10;**

**class InnerClass {**

**public int myInnerMethod() {**

**return x;**

**}**

**}**

**}**

**public class MyMainClass {**

**public static void main(String[] args) {**

**OuterClass myOuter = new OuterClass();**

**OuterClass.InnerClass myInner = myOuter.new InnerClass();**

**System.out.println(myInner.myInnerMethod());**

 }

}

Bottom of Form

Top of Form

##### **Inheritance in Java**

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship.

###### Why use inheritance in java:

* For Method Overriding (so runtime polymorphism can be achieved).
* For Code Reusability.

###### Terms used in Inheritance:

* **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
* **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
* **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
* **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

###### The syntax of Java Inheritance:

**class Subclass-name extends Superclass-name**

**{**

**//methods and fields**

**}**

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

**Types of inheritance in java:**

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.

#### ****Single Inheritance Example:****

File: TestInheritance.java

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();  }}

#### ****Multilevel Inheritance Example****

File: TestInheritance2.java

 class Animal{

 void eat(){System.out.println("eating...");}

  }

  class Dog extends Animal{

 void bark(){System.out.println("barking...");}

 }

 class BabyDog extends Dog{

 void weep(){System.out.println("weeping...");}

  }

class TestInheritance2{

public static void main(String args[]){

BabyDog d=new BabyDog();

d.weep();

d.bark();

d.eat();

}}

#### ****Hierarchical Inheritance Example****

File: TestInheritance3.java

 class Animal{

 void eat(){System.out.println("eating...");}

  }

 class Dog extends Animal{

 void bark(){System.out.println("barking...");}

  }

 class Cat extends Animal{

 void meow(){System.out.println("meowing...");}

  }

class TestInheritance3{

public static void main(String args[]){

Cat c=new Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}}

#### ****Why multiple inheritance is not supported in java?****

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

class A{

 void msg(){System.out.println("Hello");}

 }

class B{

 void msg(){System.out.println("Welcome");}

  }

 class C extends A,B{

  public static void main(String args[]){

   C obj=new C();

   obj.msg();//Now which msg() method would be invoked?

}

}

Bottom of Form

Top of Form

#### ****Abstraction in Java****

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

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

#### ****Ways to achieve Abstraction****

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

#### ****Abstract Classes****

A class that is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

#### ****Abstract class in Java****

A class that is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

#### ****Points to Remember****

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have constructors and static methods also.
* It can have final methods that will force the subclass not to change the body of the method.

#### ****Example of Abstract class that has an abstract method****

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

1. **abstract** **class** Bike{
2. **abstract** **void** run();
3. }
4. **class** Honda4 **extends** Bike{
5. **void** run(){System.out.println("running safely");}
6. **public** **static** **void** main(String args[]){
7. Bike obj = **new** Honda4();
8. obj.run();
9. }
10. }

#### ****An abstract class having constructor, data member and methods****

An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

File: TestAbstraction2.java

1. //Example of an abstract class that has abstract and non-abstract methods
2. **abstract** **class** Bike{
3. Bike(){System.out.println("bike is created");}
4. **abstract** **void** run();
5. **void** changeGear(){System.out.println("gear changed");}
6. }
7. //Creating a Child class which inherits Abstract class
8. **class** Honda **extends** Bike{
9. **void** run(){System.out.println("running safely..");}
10. }
11. //Creating a Test class which calls abstract and non-abstract methods
12. **class** TestAbstraction2{
13. **public** **static** **void** main(String args[]){
14. Bike obj = **new** Honda();
15. obj.run();
16. obj.changeGear();
17. }
18. }

Bottom of Form

Top of Form

Bottom of Form

Top of Form

###### Reference Links:

**ONLINE NOTES LINKS:**

<https://www.w3schools.com/java/java_inner_classes.asp>

<https://www.geeksforgeeks.org/inner-class-java/>

<https://www.tutorialspoint.com/java/java_innerclasses.htm>l

<https://www.w3schools.com/java/java_abstract.asp>

<https://www.javatpoint.com/abstract-class-in-java>

<https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html>

<https://www.w3schools.com/java/java_abstract.asp>

<https://www.geeksforgeeks.org/static-class-in-java/>

**VIDEO LINKS:**

<https://www.youtube.com/watch?v=okruEgWGVGU>

<https://www.youtube.com/watch?v=V7yVbG9_xkM>

<https://www.youtube.com/watch?v=RcIsb9iFKH8>

<https://www.youtube.com/watch?v=p_4Dyfplqkw>

Bottom of Form