**2.****Class and objects**

In object-oriented programming technique, we design a program using objects and classes.

Object is the physical as well as logical entity whereas class is the logical entity only.

### Object in Java

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tangible and intangible). The example of intangible object is banking system.

**Object Definitions:**

* Object is *a real world entity*.
* Object is *a run time entity*.
* Object is *an entity which has state and behavior*.
* Object is *an instance of a class*.

**Class in Java**

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**
* **nested class and interface**

**Syntax to declare a class:**

class <class\_name>  
{  
    field;    
    method;

}

# Java - Methods

A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

## Creating Method

Considering the following example to explain the syntax of a method −

**Syntax**

public static int methodName(int a, int b)   
{

// body

}

Here,

* **public static** − modifier
* **int** − return type
* **methodName** − name of the method
* **a, b** − formal parameters
* **int a, int b** − list of parameters

Method definition consists of a method header and a method body.

## The Constructors

A constructor initializes an object when it is created. It has the same name as its class and is syntactically similar to a method. However, constructors have no explicit return type.

All classes have constructors, whether you define one or not, because Java automatically provides a default constructor that initializes all member variables to zero. However, once you define your own constructor, the default constructor is no longer used.

**Example**

Here is a simple example that uses a constructor without parameters −

// A simple constructor.

class MyClass {

int x;

// Following is the constructor

MyClass() {

x = 10;

}

}

# Method Overriding in Java

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

In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

### Usage of Java Method Overriding

* Method overriding is used to provide specific implementation of a method that is already provided by its super class.
* Method overriding is used for runtime polymorphism

#### Rules for Java Method Overriding

1. method must have same name as in the parent class
2. method must have same parameter as in the parent class.
3. must be IS-A relationship (inheritance).

# 

Output:Bike is running safely

# Inheritance in Java

**Inheritance in java** is a mechanism in which one object acquires all the properties and behaviors of parent object.  
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 parent class, and you can add new methods and fields also.

Inheritance represents the **IS-A relationship**, also known as parent-child relationship.

**Syntax of Java Inheritance**

class Subclass-name extends Superclass-name

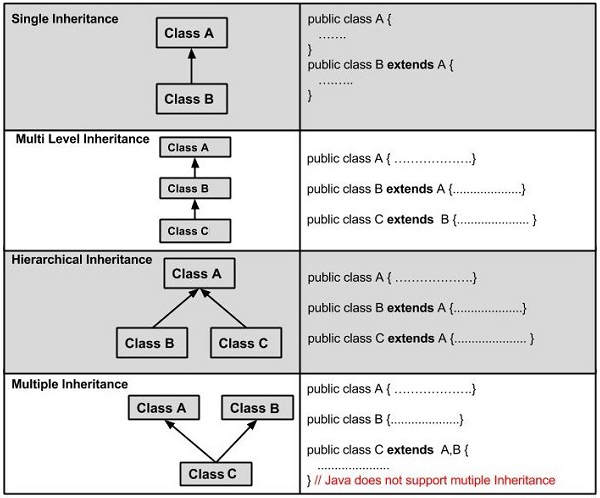
{

   //methods and fields

}

## Types of Inheritance

There are various types of inheritance as demonstrated below.



A very important fact to remember is that Java does not support multiple inheritance. This means that a class cannot extend more than one class. Therefore following is illegal −

**Example**

public class extends Animal, Mammal  
{  
}

However, a class can implement one or more interfaces, which has helped Java get rid of the impossibility of multiple inheritance.

# Interface in Java

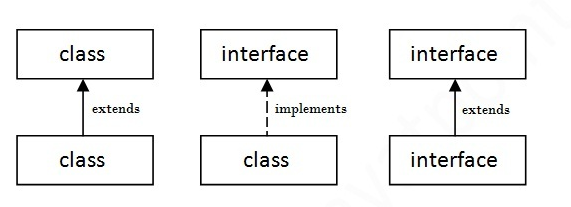
An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

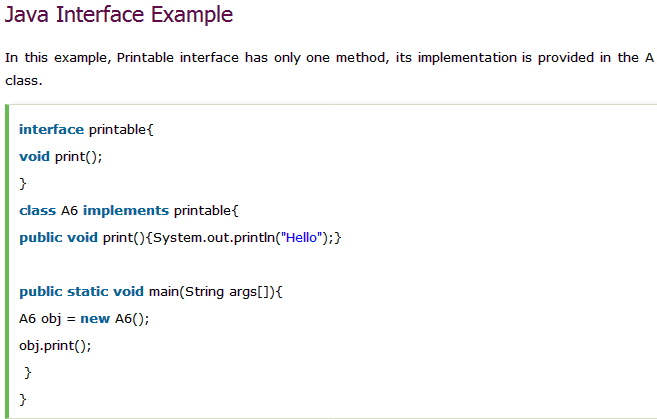
The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Java Interface also **represents IS-A relationship**.

#### Understanding relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.





Output:

Hello

## Extending Interfaces

An interface can extend another interface in the same way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

# super keyword in java

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

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

# Abstract class in Java

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

**Abstract class in Java**

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

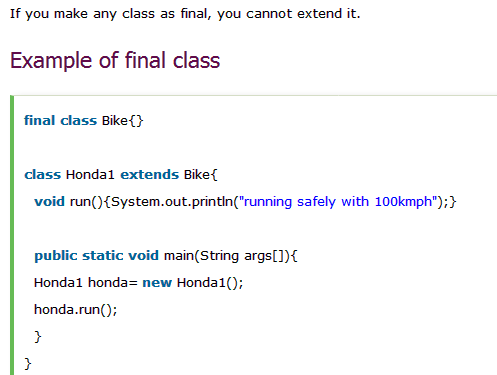
**Example abstract class**

abstract class A  
{  
}

# Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

Output:Compile Time Error

# Java Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

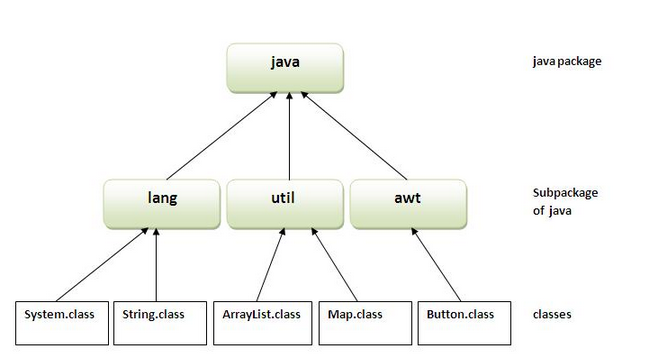
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

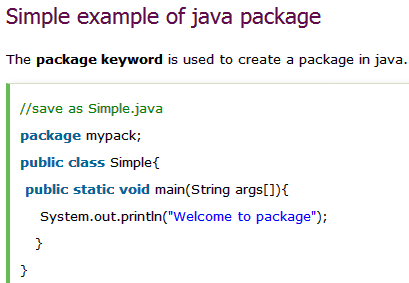
## Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.



****

**Access protection**

Java provides a number of access modifiers to set access levels for classes, variables, methods, and constructors. The four access levels are −

* Visible to the package, the default. No modifiers are needed.
* Visible to the class only (private).
* Visible to the world (public).
* Visible to the package and all subclasses (protected)

## Default Access Modifier - No Keyword

Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc.

## Private Access Modifier - Private

Methods, variables, and constructors that are declared private can only be accessed within the declared class itself.

## Public Access Modifier - Public

A class, method, constructor, interface, etc. declared public can be accessed from any other class. Therefore, fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.

Ex: public static void main(String[] arguments)   
{

// ...

}

## Protected Access Modifier - Protected

Variables, methods, and constructors, which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.