hello java example

```
 public class Simple
{
   public static void main(String args[])
  {
       System.out.println("Hello Java");
   }
 }
```

save this file as Simple.java

To compile: javac Simple.java

To execute: java Simple

How many ways can we write a java program

1. By changing sequence of the modifiers, method prototype is not changed :

Let's see the simple code of main method.

```
static public void main(String args[])
```

Java Basic

2. Subscript notation in java array can be used after type, before variable or after variable :

Let's see the different codes to write the main method.

```
public static void main(String[] args)

public static void main(String []args)

public static void main(String args[])
```

Java Basic

3. You can provide var-args support to main method by passing 3 ellipses (dots)

Let's see the simple code of using var-args in main method. We will learn about var-args later in Java New Features chapter.

```
    public static void main(String... args)
```

Java Basic

4. Having semicolon at the end of class in java is optional
:

Let's see the simple code.

```
  class A
 {
  static public void main(String... args){
  System.out.println("hello java");
  }
  };
```

Java Basic

Valid java main method signature :

```
public static void main(String[] args)
public static void main(String []args)
public static void main(String args[])
public static void main(String... args)
static public void main(String[] args)
public static final void main(String[] args)
final public static void main(String[] args)
final strictfp public static void main(String[] args)
```

Java Basic

Invalid java main method signature :

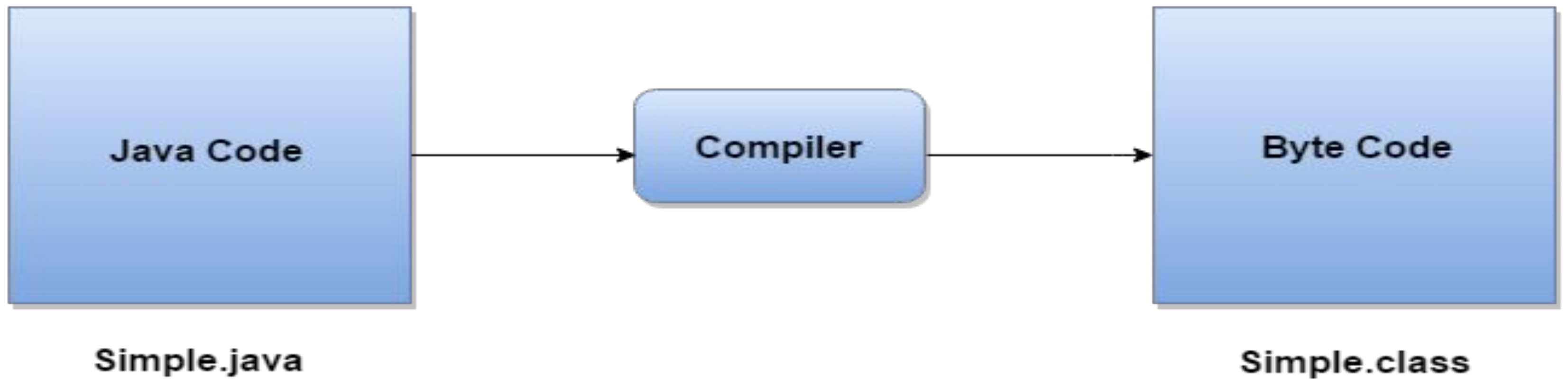
```
public void main(String[] args)
static void main(String[] args)
public void static main(String[] args)
abstract public static void main(String[] args)
```

Java Basic

Internal Details of Hello Java Program

What happens at compile time?

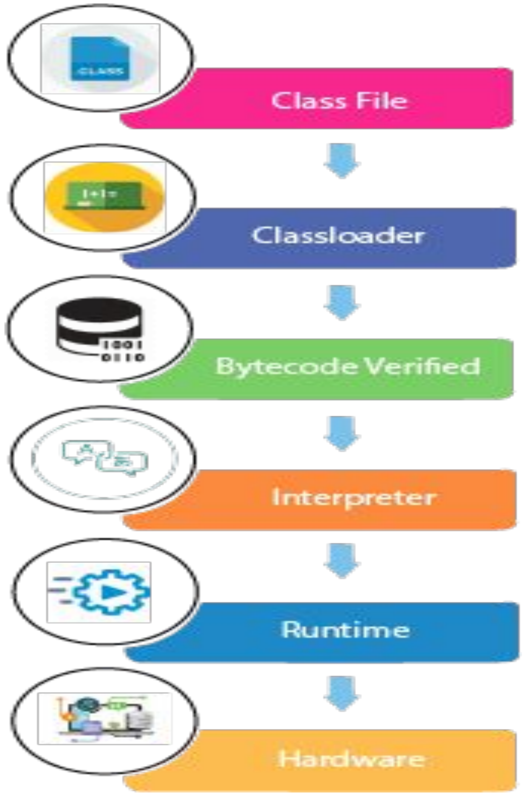
At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.



Java Basic

What happens at runtime?

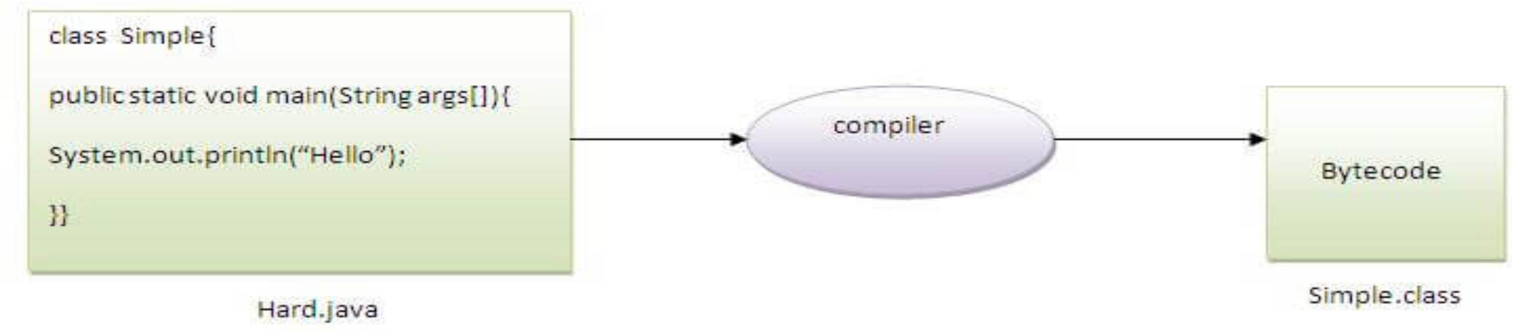
At runtime, following steps are performed:



Java Basic

Can you save a java source file by other name than the class name?

Yes, if the class is not public. It is explained in the figure given below:

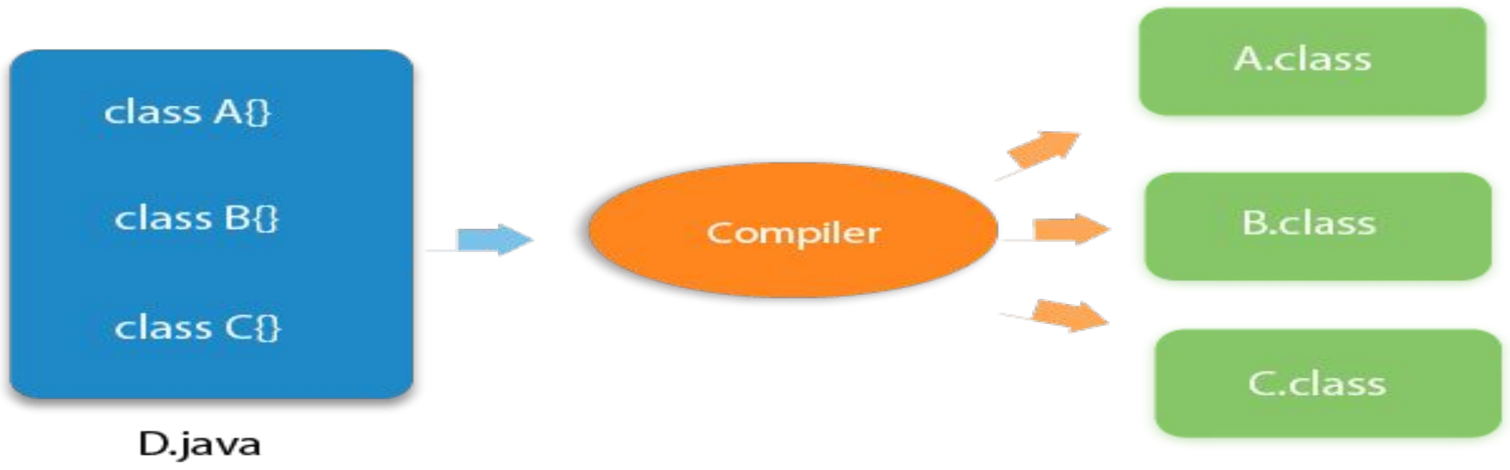


```
class Simple{
public static void main(String args[]){
System.out.println("Hello");
}}
```

To compile: javac Hard.java
To execute: java Simple

Can you have multiple classes in a java source file?

Yes, like the figure given below illustrates:



Java OOPs Concepts

Object Oriented Programming is a paradigm that provides many concepts such as inheritance, data binding, polymorphism etc.

Simula is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.

Smalltalk is considered as the first truly object-oriented programming language.
The popular object-oriented languages are Java, C#, PHP, Python, C++ etc.

Java Basic

Java OOPs Concepts

Object means a real word entity such as pen, chair, table etc.
Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object :

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing details of each other's data or code, the only necessary thing is that the type of message accepted and type of response returned by the objects.
Example: A dog is an object because it has states i.e. color, name, breed etc. as well as behaviors i.e. wagging the tail, barking, eating etc.

Java Basic

Class :

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn?t store any space.

Inheritance :

When one object acquires all the properties and behaviours of parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Java Basic

Polymorphism :

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

Java Basic

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

Java Basic

Encapsulation:

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Java Basic

Advantage of OOPs over Procedure-oriented programming language :

1) OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

2) OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.

Java Basic

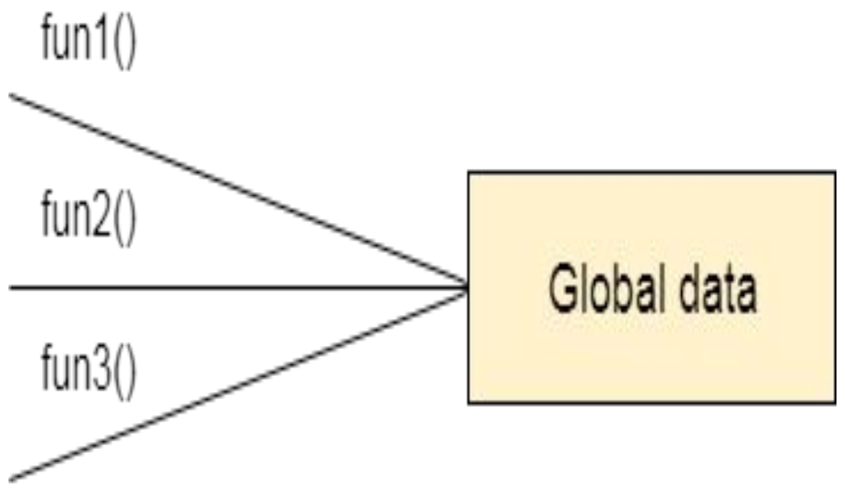


Figure: Data Representation in Procedure-Oriented Programming

Java Basic

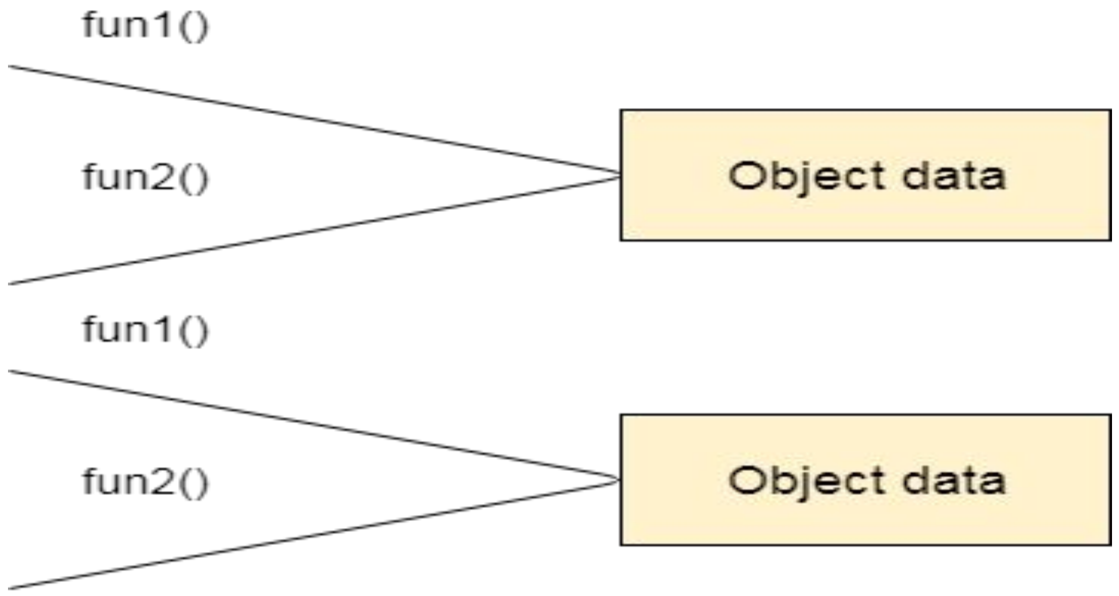


Figure: Data Representation in Object-Oriented Programming

Java Basic

3) OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

Java Basic

What is difference between object-oriented programming language and object-based programming language?

Object based programming language follows all the features of OOPs except Inheritance. JavaScript and VBScript are examples of object based programming languages.

Java Basic

Java Naming conventions :

Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method etc.

But, it is not forced to follow. So, it is known as convention not rule.

All the classes, interfaces, packages, methods and fields of java programming language are given according to java naming convention.

Java Basic

Advantage of naming conventions in java :

By using standard Java naming conventions, you make your code easier to read for yourself and for other programmers. Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.

Java Basic

Advantage of naming conventions in java :

Name	Convention
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Java Basic

CamelCase in java naming conventions :

Java follows camelcase syntax for naming the class, interface, method and variable.

If name is combined with two words, second word will start with uppercase letter always e.g. actionPerformed(), firstName, ActionEvent, ActionListener etc.

Java Basic

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.

An object has three characteristics:

state: represents data (value) of an object.
behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

Java Basic

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

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.

Java Basic

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

Java Basic

Syntax to declare a class:

```
class <class_name>
{
    field;
    method;
}
```

Instance variable in Java :

A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object(instance) is created. That is why, it is known as instance variable.

Java Basic

Method in Java :

In java, a method is like function i.e. used to expose behavior of an object.
Advantage of Method

- Code Reusability
- Code Optimization

new keyword in Java :

The new keyword is used to allocate memory at run time. All objects get memory in Heap memory area.

Object and Class Example: main within class

```
class Student
{
 int id;//field or data member or instance variable
 String name;

 public static void main(String args[])
 {
  Student s1=new Student();//creating an object of Student
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

Object and Class Example: main outside class

In real time development, we create classes and use it from another class. It is a better approach than previous one. Let's see a simple example, where we are having main() method in another class.

We can have multiple classes in different java files or single java file. If you define multiple classes in a single java source file, it is a good idea to save the file name with the class name which has main() method.

Object and Class Example: main outside class

```
 class Student
 {
  int id;
  String name;
 }
 public class TestStudent1
{
  public static void main(String args[])
  {
   Student s1=new Student();
   System.out.println(s1.id);
   System.out.println(s1.name);
  }
 }
```

Object Initialization

3 Ways to initialize object :

There are 3 ways to initialize object in java.

1. By reference variable
2. By method
3. By constructor

Object Initialization

1) Object and Class Example: Initialization through reference :

```
class Student{
 int id;
 String name;
}
public class TestStudent{
 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Ankit";
  System.out.println(s1.id+" "+s1.name);//printing members
with a white space
 } }
```

Object Initialization

```
class Student{
 int id;
 String name;
}
public class TestStudent{
 public static void main(String args[]){
  //Creating objects
  Student s1=new Student();
  Student s2=new Student();
  //Initializing objects
  s1.id=101;
  s1.name="Ankit";
  s2.id=102;
  s2.name="Amit";
  //Printing data
  System.out.println(s1.id+" "+s1.name);
  System.out.println(s2.id+" "+s2.name);
 }  }
```

Object Initialization

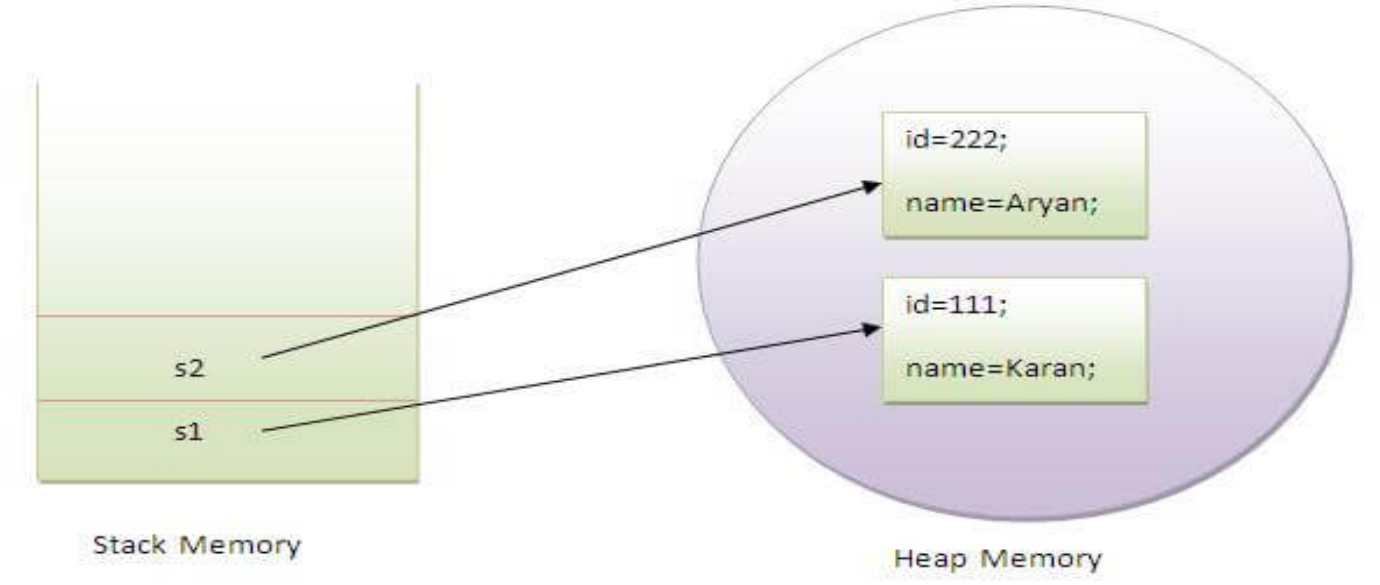
2) Object and Class Example: Initialization through method :

```
class Student
{
 int rollno;
 String name;
 void insertRecord(int r, String n)
{
  rollno=r;
  name=n;
 }
 void displayInformation()
  {System.out.println(rollno+" "+name);}
}
```

Object Initialization

```
public class TestStudent
{
 public static void main(String args[])
 {
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

Object Initialization



Object Initialization

3) Object and Class Example: Initialization through constructor

We will learn about constructors in constructor topic later.

Object Initialization

Object and Class Example: Employee

```
class Employee{
    int id;
    String name;
    float salary;
    void insert(int i, String n, float s) {
        id=i;
        name=n;
        salary=s;
    }
    void display(){System.out.println(id+" "+name+"
"+salary);}
}
```

Object Initialization

```
public class TestEmployee {
public static void main(String[] args) {
    Employee e1=new Employee();
    Employee e2=new Employee();
    Employee e3=new Employee();
    e1.insert(101,"ajeet",45000);
    e2.insert(102,"irfan",25000);
    e3.insert(103,"nakul",55000);
    e1.display();
    e2.display();
    e3.display();
}
}
```

Object Initialization

Object and Class Example: Rectangle

```
class Rectangle{
 int length;
 int width;
 void insert(int l, int w){
  length=l;
  width=w;
 }
 void calculateArea(){System.out.println(length*width);}
}
```

Object Initialization

```
public class TestRectangle1{
 public static void main(String args[]){
  Rectangle r1=new Rectangle();
  Rectangle r2=new Rectangle();
  r1.insert(11,5);
  r2.insert(3,15);
  r1.calculateArea();
  r2.calculateArea();
}
}
```

Object Initialization

What are the different ways to create an object in Java?

There are many ways to create an object in java. They are:

- **By new keyword**
- **By newInstance() method**
- **By clone() method**
- **By deserialization**
- **By factory method**

We will learn these ways to create object later.

Object

Anonymous object :

Anonymous simply means nameless. An object which has no reference is known as anonymous object. It can be used at the time of object creation only.

If you have to use an object only once, anonymous object is a good approach. For example:

```
new Calculation();//anonymous object
```

Object

Anonymous object :

Calling method through reference:

```
Calculation c=new Calculation();
c.fact(5);
```

Calling method through anonymous object:

```
new Calculation().fact(5);
```

Object

Anonymous object :

```
class Calculation{
 void fact(int  n){
  int fact=1;
  for(int i=1;i<=n;i++){
   fact=fact*i;
  }
 System.out.println("factorial is "+fact);
}
public static void main(String args[]){
 new Calculation().fact(5);//calling method with
anonymous object
} }
```

Object

Creating multiple objects by one type only :

We can create multiple objects by one type only as we do in case of primitives.

Initialization of primitive variables:

```
    int a=10, b=20;
```

Initialization of refernce variables:

```
Rectangle r1=new Rectangle(), r2=new Rectangle();
//creating two objects
```

Object

```
class Rectangle
{
 int length;
 int width;
 void insert(int l,int w)
 {
  length=l;
  width=w;
 }
 void calculateArea()
 {
    System.out.println(length*width);
 }
}
```

Object

```
public class TestRectangle
{
     public static void main(String args[])
     {
      Rectangle r1=new Rectangle(),r2=new Rectangle();
      //creating two objects
      r1.insert(11,5);
      r2.insert(3,15);
      r1.calculateArea();
      r2.calculateArea();
    }
 }
```

Object

Real World Example: Account

```
class Account{
int acc_no;
String name;
float amount;
void insert(int a,String n,float amt){
acc_no=a;
name=n;
amount=amt;
}
```

Object

```
void deposit(float amt){
amount=amount+amt;
System.out.println(amt+" deposited");  }
void withdraw(float amt){
if(amount<amt){
System.out.println("Insufficient Balance");
}else{
amount=amount - amt;
System.out.println(amt+" withdrawn");
} }
void checkBalance(){System.out.println("Balance is:
"+amount);}
void display(){System.out.println(acc_no+" "+name+"
"+amount);}
}
```

Object

```
public class TestAccount
{
public static void main(String[] args)
{
Account a1=new Account();
a1.insert(832345,"Ankit",1000);
a1.display();
a1.checkBalance();
a1.deposit(40000);
a1.checkBalance();
a1.withdraw(15000);
a1.checkBalance();
}
}
```