**OOPs (Object-Oriented Programming System)**

* **Object** means a real-world entity such as a pen, chair, table, computer, watch, etc.
* **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.
* The main aim of object-oriented programming is to implement real-world entities for example object, classes, abstraction, inheritance, polymorphism, etc.
* It simplifies the software development and maintenance.

**Note:**

* **Simula** is considered the first object-oriented programming language. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language.
* **Smalltalk** is considered the first truly object-oriented programming language.  The popular object-oriented languages are [Java,](https://www.javatpoint.com/java-tutorial) [C#,](https://www.javatpoint.com/c-sharp-tutorial) [PHP,](https://www.javatpoint.com/php-tutorial) [Python,](https://www.javatpoint.com/python-tutorial) [C++,](https://www.javatpoint.com/cpp-tutorial) etc.

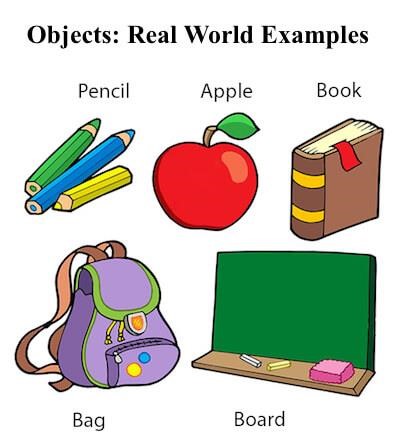
**Object:**

**An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

**Object Definitions:**

* An object is *a real-world entity*. o An object is *a runtime entity*. o The object is *an entity, which has state and behavior*.
* The object is *an instance of a class*.
* Object is the basic unit of object-oriented programming. o An object is a variable of type class. o Stores data and class methods.
* Example: Alto, Zen Estilo, Swift etc are the objects of class Car.

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 an intangible object is the banking system.

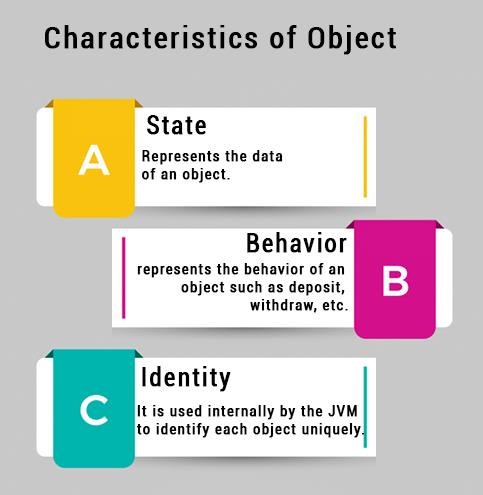


An object has three characteristics:

* **State:** represents the data (value) of an object.

o **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.

* **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.



**Creation of Objects:**

* Once the class is created, one or more objects can be created from the class as objects are instance of the class.
* Objects are declared as:

Syntax: class-name object-name= new class-name();

Example: Student S1=new Student();

|  |  |  |
| --- | --- | --- |
| The | **new** | keyword is used to allocate memory at runtime. All objects get memory in Heap memory |
| area. | | |

**Class:**

* Class is a collection of data members and member function.
  + Data, representing an object or "thing".
  + Operations on that data.
* Class is a user defined data type.
* Class implements encapsulation and abstraction.
* Member data specifies the type of data to be stored.
* Member function acts as a mediator between user and data. They manipulate data.
* 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

**Class Declaration/ Class prototyping:**

* A class is specified with a name after the keyword class.
* The body of class begins with curly bracket ‘{‘ and is followed by the code containing the data members and functions. Then the class is closed with curly bracket ‘}’ and concluded with a semi-colon (;)- Optional.
* There are different access specifiers for defining the data and functions present inside a class.
* In Java the class can be declared as follows:
* Syntax: class classname

{

list of data members; constructors list of member function; etc..

} ; (optional)

**Methods or Functions**

* A Method/function is a group of statements that performs any coherent task.
* A function contains set of statements that are bundled within a set of curly brackets and is identified by a name called function name.
* It can be classified into two categories:
  + Predefined or In-Built functions
  + User-defined function.
* The following three things have to be done to use a function

o **Function Prototyping** Syntax:

Return type function name (list of formal arguments);

* + **Function Definition** Syntax:

Return type function name (list of formal arguments)

{

Set of statements;

}

* + **Function Calling** Syntax:

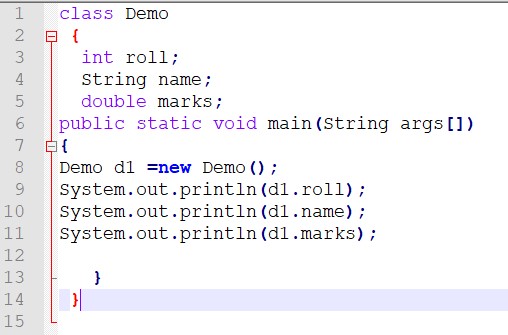
Object.function name (list of actual arguments);

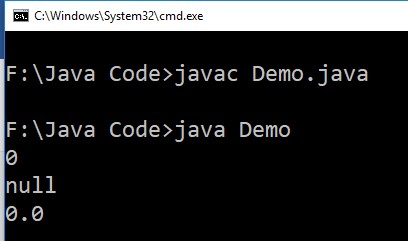
int addition(int x, int y) –takes something, returns something void addition(int x,int y)- takes something returns nothing int addition( )- takes nothing returns something void addition() - takes noting, return nothing

**Access specifiers:**

* Access specifiers are used to identify access rights for the data and member functions of the class. There are four types of access specifiers in Java programming language:
* Private (Access within class)
* protected (Access within class and derived class) o
* Package (Access within directory)it is by default in Java o
* Public (Access from anywhere)

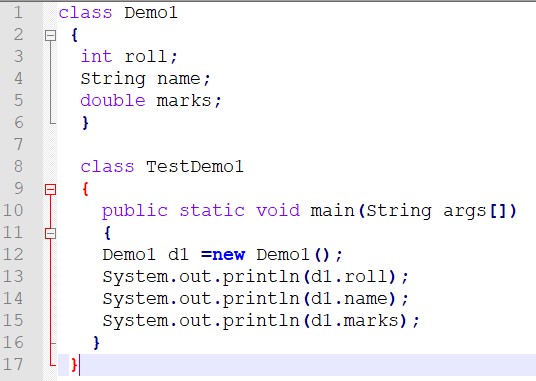
Java Program to illustrate how to define a class and fields.



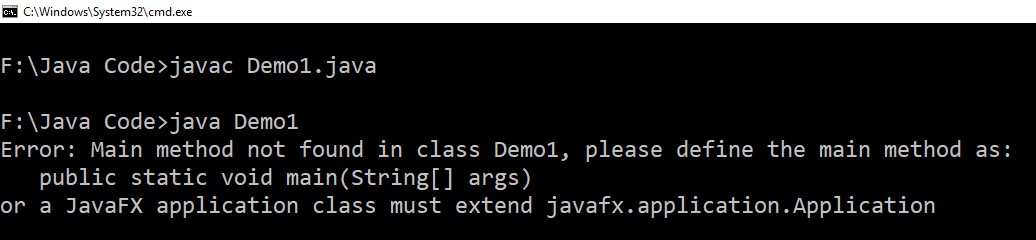


**Note:** In real time development, we create classes and use it from another class. It is a better approach than previous one. Let's see a simple example, where we are having main() method in another class.

# main outside the class

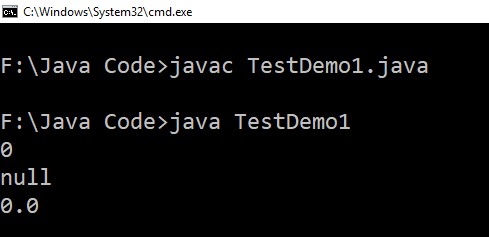


If we save this program as Demo1.java,it will show compile time error.



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.

Save it: TestDemo1.java (it contains main() method)

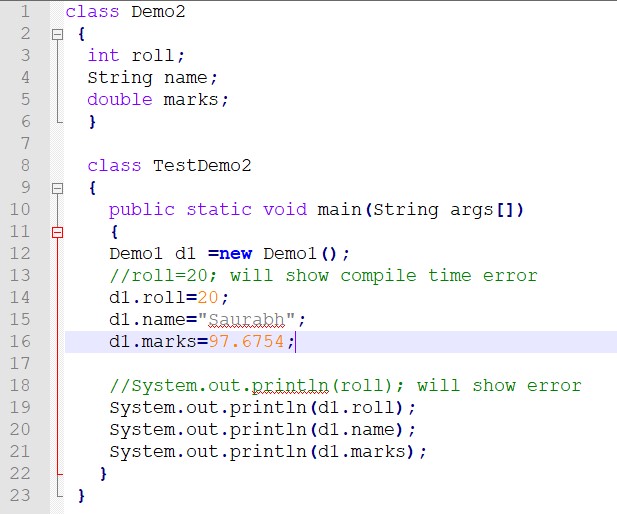


There are 3 ways to initialize object in java.

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

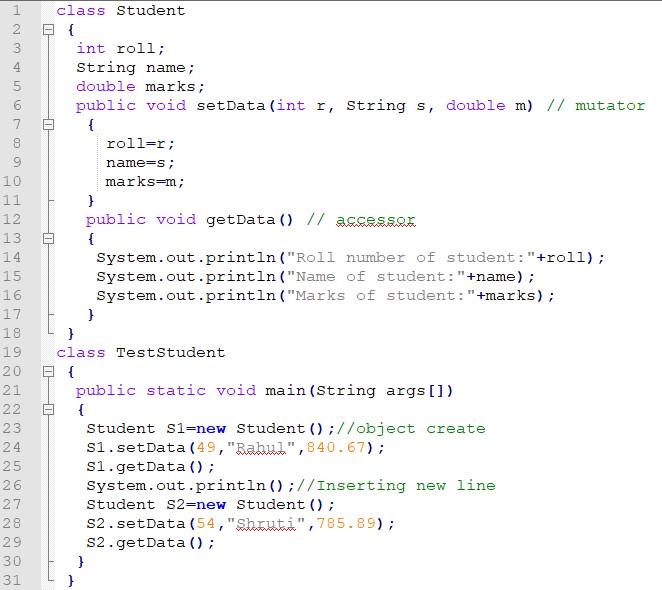
**Initialize object by reference variable**

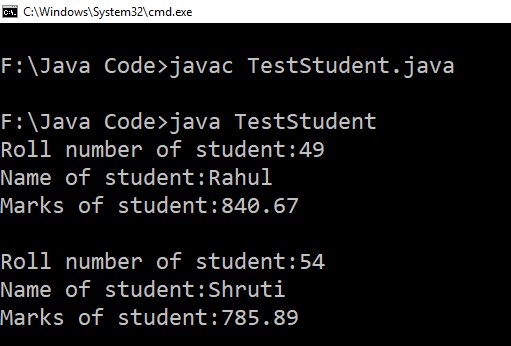
Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.



# Initialization through method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the method.





**Initialization through a constructor:** Will discuss later**.**

Take input from user and print area of rectangle.

import java.util.Scanner;

class Rectangle

{

int l,b,area;

public void insertData()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter length");

l=s.nextInt();

System.out.println("Enter Width");

b=s.nextInt();

}

public void printArea()

{

area=l\*b;

System.out.println("Area is :"+area);

}

}

class TestRectangle

{

public static void main(String args[])

{

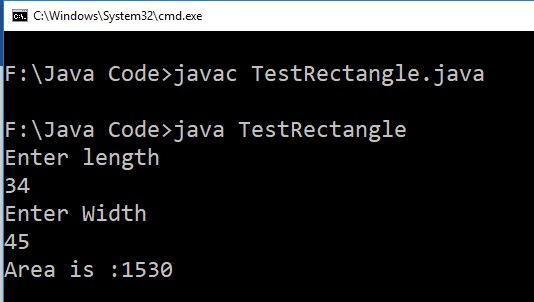
Rectangle r=new Rectangle();

r.insertData();

r.printArea();

}

}



**Java Program to demonstrate the working of a banking system, where we deposit and withdraw amount from our account. Display the transaction details of customer account.**

class Account1{

int acc\_no;

String name;

double amount;

//Method to initialize object

void insert(int a,String n,double amt)

{

acc\_no=a; name=n;

amount=amt;

}

//deposit method

void deposit(double amt)

{

amount=amount+amt;

System.out.println(amt+" deposited");

}

//withdraw method

void withdraw(double amt){

if(amount<amt){

System.out.println("Insufficient Balance");

}else{

amount=amount-amt;

System.out.println(amt+" withdrawn");

}

}

//method to check the balance of the account

void checkBalance()

{

System.out.println("Balance is: "+amount);

}

//method to display the values of an object

void display()

{

System.out.println("Account No is:"+ acc\_no);

System.out.println("Customer Name is:"+name);

}

}

//Creating a test class to deposit and withdraw amount

class Main{

public static void main(String[] args){

Account1 a1=new Account1();

a1.insert(123456,"Saurabh",50000);

a1.display();

a1.checkBalance();

a1.deposit(5000);

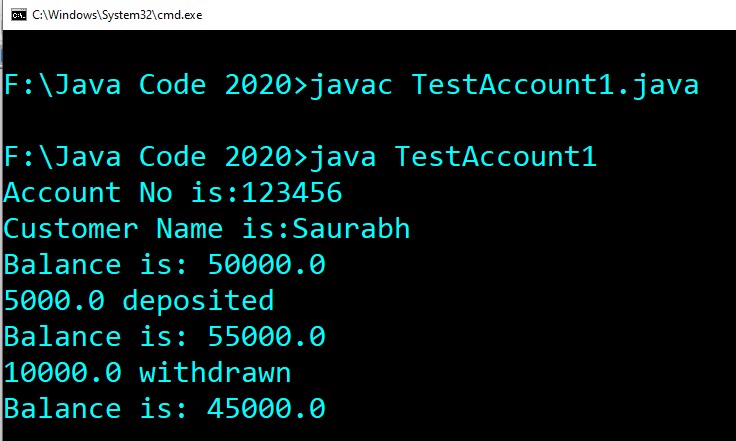
a1.checkBalance();

a1.withdraw(10000);

a1.checkBalance();

}

}



Using Scanner class:

import java.util.\*; class Account{ long a; String n; double amt;

Scanner s=new Scanner(System.in); //Method to initialize object void insert(){

System.out.println("Enter acc\_no"); long a=s.nextLong(); System.out.println("Enter name");

String n =s.next();

System.out.println("Enter amount"); double amt=s.nextDouble();

this.a=a; this.n=n; this.amt=amt;

}

//deposit method void deposit(){

System.out.println("Enter amount to be deposited");

double damount=s.nextDouble(); amt=damount+amt;

System.out.println(damount+" deposited");

}

//withdraw method void withdraw(){

System.out.println("Enter amount to be withdrawn"); double wamount=s.nextDouble(); if(amt<wamount){

System.out.println("Insufficient Balance");

}else{ amt=amt-wamount;

System.out.println(wamount+" withdrawn");

}

}

//method to check the balance of the account

void checkBalance(){System.out.println("Balance is: "+amt);}

//method to display the values of an object void display()

{System.out.println("Account No is:"+ a);

System.out.println("Customer Name is:"+n);

}

}

//Creating a test class to deposit and withdraw amount

class TestAccount{

public static void main(String[] args){ Account a1=new Account();

a1.insert(); a1.display(); a1.checkBalance(); a1.deposit(); a1.checkBalance(); a1.withdraw(); a1.checkBalance();

}}

