**Introduction to Java:**

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Object Oriented − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

Platform Independent − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

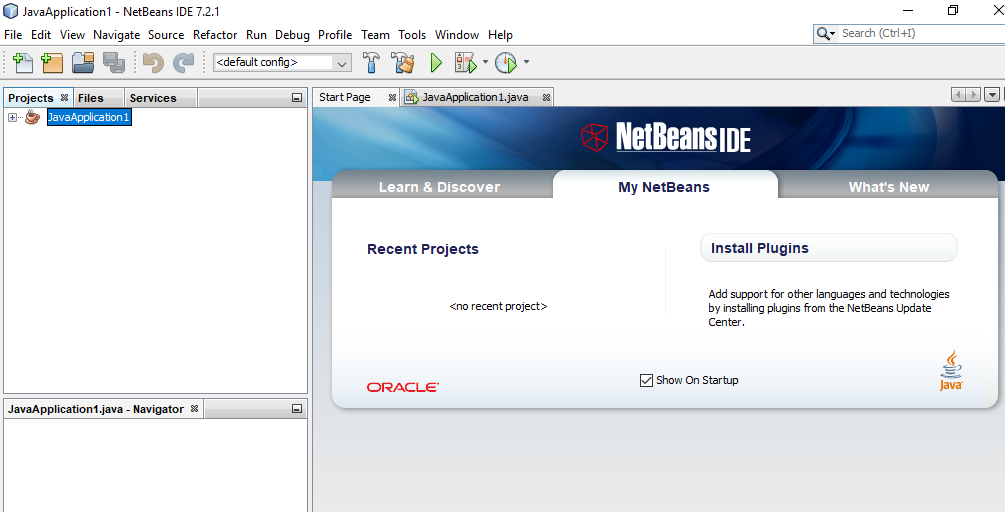
Simple − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

**Installation Link:**

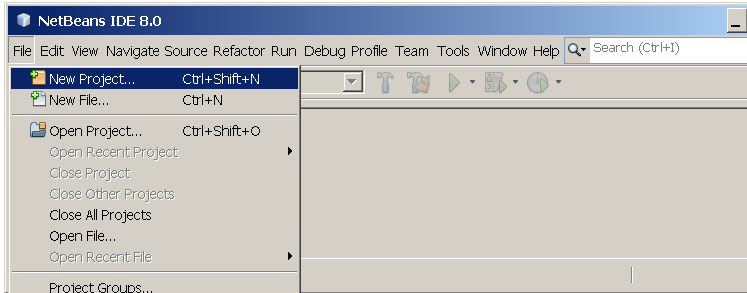
<https://drive.google.com/file/d/1QwcUio57hJoFiIRaWdOLaPbjpzKifgtd/view?usp=sharing>

**Creating a java Project:**

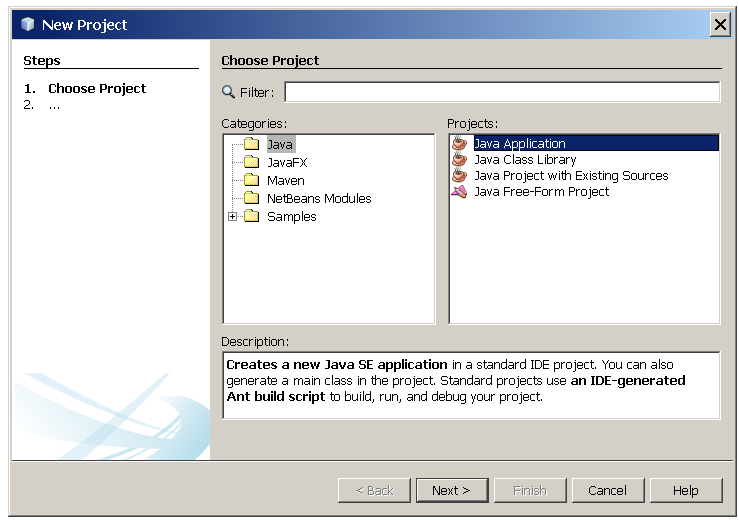
Open NetBeans:



In the NetBeans IDE, choose **File** | **New Project...**.

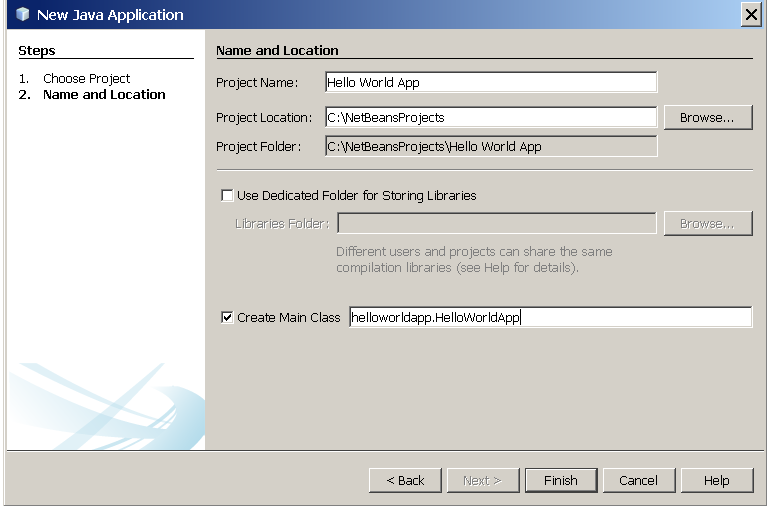


In the **New Project** wizard, expand the **Java** category and select **Java Application** as shown in the following figure:



In the **Name and Location** page of the wizard, do the following (as shown in the figure below):

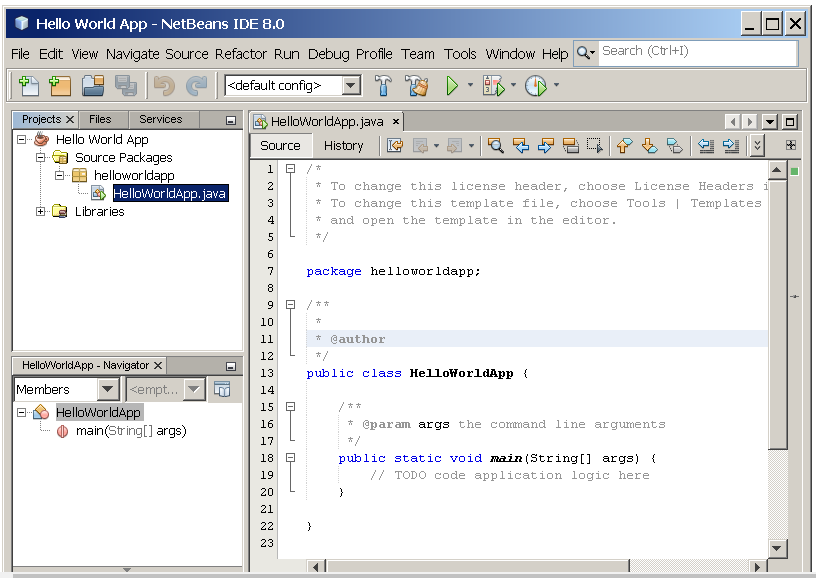
* In the **Project Name** field, type Hello World App.
* In the **Create Main Class** field, type helloworldapp.HelloWorldApp.



Click Finish.

The project is created and opened in the IDE. You should see the following components:

* The **Projects** window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
* The **Source Editor** window with a file called HelloWorldApp.java open.
* The **Navigator** window, which you can use to quickly navigate between elements within the selected class.



**Classes, Member Variables, Methods and Objects**

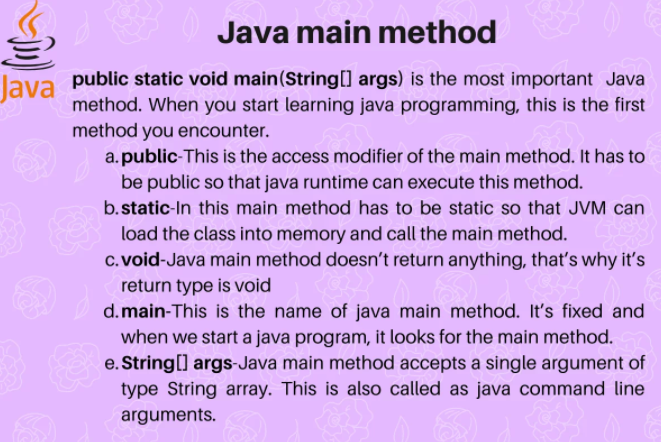
Classes provide us the platform for modular programming by allowing us to encapsulate data and methods in a single unit. Placing data and functions together into a single entity is the central idea of object oriented programming. Class can have public and private members. They also have protected members which we will see some time later. An object has the same relationship to a class that a variable has to a data type. An object is said to be an instance of a class, in same way as 1988 Corolla is an instance of a vehicle.

There are several kinds of **variables**:

* Member variables in a class—these are called *fields*.
* Variables in a method or block of code—these are called *local variables*.
* Variables in method declarations—these are called *parameters*.

You've seen classes defined in the following way:

|  |
| --- |
| public class MyFirstJavaProgram {  /\* This is my first java program.  \* This will print 'Hello World' as the output  \*/  public static void main(String []args) {  System.out.println("Hello World"); // prints Hello World  }  } |



This is a *class declaration*. The *class body* (the area between the braces) contains all the code that provides for the life cycle of the objects created from the class: constructors for initializing new objects, declarations for the fields that provide the state of the class and its objects, and methods to implement the behavior of the class and its objects.

A class defines a new type of data. Box is a new data type in this case.

|  |
| --- |
| class Box  {  private int lenght;  private int width;  private int height;  } |

Class declaration only creates a template. Not actual object. To actually create a Box object:

|  |
| --- |
| Box mybox=new my Box(); |

Now mybox is an instance of Box.

You can create multiple objects of one class. Consider the following example:

|  |
| --- |
| class Car  {  public String brand;  public String model;  public int year;  }  public class JavaApplication1  {  public static void main(String[] args)  {  Car carObj1=new Car();  carObj1.brand = "BMW";  carObj1.model = "X5";  carObj1.year = 1999;  // Create another object of Car  Car carObj2=new Car();;  carObj2.brand = "Ford";  carObj2.model = "Mustang";  carObj2.year = 1969;  System.out.println(carObj1.brand + " "+ carObj1.model+ " "+carObj1.year);  System.out.println(carObj2.brand + " "+ carObj2.model+ " "+carObj2.year);  }  } |

## Class Methods

Methods are **functions** that belongs to the class. Inside class definition

In the following example, we define a function inside the class, and we name it "myMethod".

**Note:** You access methods just like you access attributes; by creating an object of the class and by using the dot syntax (.).

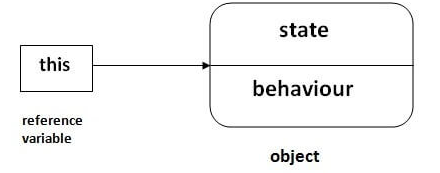
|  |
| --- |
| class MyClass  {  public void myMethod()  {  System.out.println("Hello World!\n");  }  }  public class JavaApplication1  {  public static void main(String[] args)  {  MyClass myObj=new MyClass(); // Create an object of MyClass  myObj.myMethod();  }  } |

## Parameters

You can also add parameters:

|  |
| --- |
| class Car {  public int speed(int maxSpeed)  {  return maxSpeed+70;  }  }  public class JavaApplication1  {  public static void main(String[] args)  {  Car myObj=new Car(); // Create an object of Car  System.out.println( myObj.speed(200)) ; // Call the method with an argument    }  } |

**Using the “this” Keyword：**



1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

|  |
| --- |
| 1. **class** Student{ 2. **int** rollno; 3. String name; 4. **float** fee; 5. Student(**int** rollno,String name,**float** fee){ 6. **this**.rollno=rollno; 7. **this**.name=name; 8. **this**.fee=fee; 9. } 10. **void** display(){System.out.println(rollno+" "+name+" "+fee);} 11. } 13. **class** TestThis2{ 14. **public** **static** **void** main(String args[]){ 15. Student s1=**new** Student(111,"ankit",5000f); 16. Student s2=**new** Student(112,"sumit",6000f); 17. s1.display(); 18. s2.display(); 19. }} |