<https://www.tutorialspoint.com/java/java_unicode_system.htm>

<https://javaconceptoftheday.com/>

<https://www.programiz.com/java-programming/operators>

**Java - Classes and Objects:**

Class : Blue print for creating the objects. It is logically exists.

Object: instance of a class. It is physically exists.

Blue print: its template. Eg: House blue print. It is logical.



Is it real? Is it existing? Is physical or logical?

There is a House.

Animal. 🡪 4 legs, 1 tail, 2 eyes, white teeth.

Running(), walkng(), eating(), makeSounds(), sleeping()….

All these are ogically exists. what

Dog, cat, Tiger, lion..etc 🡪 4 legs, 1 tail, 2 eyes, white teeeth.

Running(), walkng(), eating(), makeSounds(), sleeping()….

All these are physically existing. Whatever physically exists in the world is called as Object.

What ever you see with your eyes is called as Object.

**Object: every object should contain below properties.**

1. State 🡪 means : **properties / fields/ variables**.
2. Behavior: **methods / functions / actions**
3. Identity. -OPTIONAL.: object name / object reference name.

**State:**

Every object has some **properties / fields/ variables**.

Those properties are called as state.

For example:

Book properties: width, size, weight, price, no:of pages, author name.

Bike properties: color, weight, price, speed, milage, lights, bikeCompanyName, wheels type, etc…

Etc..examples.

**Behavior:**

Every object performs some actions; those actions are called as **behavior / methods / functions / actions**.

For example:

Book actions: reading(), writing(), playing(), beating(),

Bike actions: riding(), engineBeating(), spped(), stunts(), etc….

**Identity:**

Means giving a name to the object. It is optional. If an object doesn’t have identity means we can not say it is not an object. Even though it is not having identity but still it is having STATE and BEHAVIOR.

Class and objects both have state, behavior and identity. A class have logical state, behavior and identity.

But an object have physical state, behavior and identity.

Now our target is to convert every object state and behavior as program.

Without class there is No object. Means first we need to create class, then after using that class we need to create object.

<https://www.tutorialspoint.com/java/java_oops_concepts.htm>

OOP’s concepts:



Class

Object

**P**olymorphism.

**I**nheritance.

**E**ncapsulation. (**PIE**).

Abstraction.

What a class contains?

1. Constructors.
2. Properties / Fields / Variables.
3. Methods – Abstract methods, concrete methods, static methods.
4. Enums.

What types of Classes we have?

1. Classes with business logic.
2. Class for storing the data and transferring the data. (**POJO**-Plain Old Java Object **class**)

**Creating MAVEN project in STS / Eclipse:**

STS 🡪 File 🡪 new 🡪 Maven project🡪 next 🡪 type “**maven-archetype-quickstart**” 🡪 click next button 🡪



Next fill the required details.



Here “**Group Id**” means package name.

“**artifcatId**” means project name.

And click on finish button. Your Maven project will be created.

Below is the maven project structure created.



Here

**src/main/java** 🡪 this the place where we are going to write java code under given package. (com.skh is package).

src/test/java 🡪 this is the place where we are going to write test cases.

Pom.xml file 🡪 it is MAVEN related file. 🡪 which is used for downloading external jars.

**Jar?**

Whenever any developer develops the project. That project will be released in different formats..exe, .bat, ;.pdf, .text, .apk,

In java, developed projects are released in below different formats.

1. .**Jar** (Java Archives). 🡪 99.99%
2. .war (Web Archive). 🡪 0.1%
3. .ear (Enterprise Archive) 🡪 0.0%

Developed software will be release in above formats.

Above are compressed files. They will club/ combine all file together in compressed file.

**How to create our project .jar file in STS / Eclipse?**

**OR**

**How to run maven build in STS?**

Go to project in STS 🡪 right click on project 🡪 Select “Run AS” 🡪 Maven build 🡪 it will open on popup 🡪 In pop up go to “Goals” option 🡪 type there “**clean install**” 🡪 Apply 🡪 Run.



Once the build is completed, on STS console we can see where our project .jar file is generated.

Generally jar file will be created in our “project root folder” inside “**target**” folder.



We can see there is a .jar file created.



**Question?**

**I don’t know where is my project is created by using STS, how to find project root folder in STS?**

Right click on the project 🡪 properties🡪 click on “resource” option 🡪



**Question?**

**What is meant by build? 🡪 [target folder creation + code compile + by combining all .class files creating .jar file]**

When we run build, all java classes in project are compiled and generated all .class files kept inside **target** folder of project root folder.

We no need to worry about **target** folder, when run maven build, automatically **target** folder will be created.

We can delete “target” folder, why because when run maven build, automatically **target** folder will be created again and again.

**What .jar file contains?**

.jar files contains all compiled java classes, means a .jar file contains all .class files only.



We are working with Spring boot means we are working with their .jar files.

We are working with servlets means we are working with their .jar files.

We are working with Hibernate means we are working with their .jar files.

**How to download spring boot or servlets or hibernate external jar files / dependencies in our project?**

To download other external jar files in our project **we need to make use of “maven – pom.xml” tool or “gradle – build.gradle” tool.**

In maven/gradle we need to add external projects **dependencies**.

**Currently “gradle” is tending in the market.**

Build tools:

1. Ant
2. Maven
3. Gradle
4. Jenkins – we use in Server.

The main drawbacks of Ant include −

* XML is used as a format to write the build scripts.
* Being hierarchical is not good for procedural programming, and
* XML is relatively unmanageable.

The main drawbacks of Maven include −

* It does not handle the conflicts between versions of the same library.
* Complex customised build scripts are difficult to write in Maven, as compared to writing the build scripts in **ANT**.

<https://www.tutorialspoint.com/gradle/gradle_overview.htm>



<https://mvnrepository.com/>

Go to above website and copy the required dependency and paste it in either maven pom.xml file or gradle build.gradle file. Then it will download all required jar files into your project.

If we add dependencies .jar files will be downloaded.

**How many “public” classes a .java file contains?**

A .java file can contain only one public class. If we write more than one public class, we are going to get compilation error.

**To work with class.**

Create a class, declare variables if required, create constructors if required, create methods. Write the logic inside methods.

-Class name constructor name should be same.

-“void” means NO Return type.

Create another class, create main() method in that class and create object to above class in main() method and call methods. Now run this 2nd class.

**What is the difference between method and constructor?**

-Method have return type where as constructor don’t have return type.

-class name and constructor name always should same, methods name we can give any names as per our choice.

-when ever we created object that time only Constructor will be called and executed, but methods we need to call after creating object.

- constructor will be called only once while creating the object, but methods we can call any number of times using object reference.

<https://www.programiz.com/java-programming/variables-literals>

<https://www.tutorialspoint.com/java/java_date_time.htm>

**Variables: varies/ changeable. Over the time values can be change.**

* + 1. Class level / global level 🡪 we create outside the methods. Scope is complete class. Class level variables we can use in all methods.
    2. Method / construcot level. 🡪 we create variables inside the method. Scope is only that declared method. Variable data we can use only in that particular method.

**How to display the content on console?**

int a = 10;  
int b = 34;  
int c =45;  
float ff = 12.23f;

**1st approach: using “concatenation”.**  
 System.*out*.println("The value of a is " + a+ " "+ b +" "+ c + " "+ ff);  
**2nd approach: using String class format() method.**

System.*out*.println**(String.*format*("**The value of a is **%d %d %d %f", a, b, c, ff**));

**8 Primitive Data Types:**

BSIL 🡪 bite short int long





**Operators in Java can be classified into 5 types:**

1. Arithmetic Operators.
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Unary Operators
6. Bitwise Operators

Only for primitive types we use **==** operator. For objects comparison we always use .equals() method.





Java instanceof Operator.

Java Ternary Operator.

<https://techvidvan.com/tutorials/java-virtual-machine/>

**Static keyword:**

I want to call other class variables or methods... without creating object?  
How to do it? --> we can do it by using "**static**" keyword.  
If a variable or method is static type.. **we can call them by using className, we don’t need to create object to call them**.  
  
eg: Employe.**name**, Student.**id**, Student.**fetchFullName();**  
  
For calling non-static variables or methods. We must create object and call them with object referecne.

|  |  |
| --- | --- |
| public class CoreJavaTest {  int a = 10;  static String *name* = "Azad";  public static void m1(){  System.*out*.println("This is a method..!");  } } | public class App {   public static void main(String[] args) {  System.*out*.println(CoreJavaTest.*name*);  CoreJavaTest.*m1*();  CoreJavaTest javaTest = new CoreJavaTest();  System.*out*.println(javaTest.a);  } } |

**We have 8 primitive data types. [BSIL, float, double, char, boolean]**

**Apart from JDK given primitive data types we can use any kind of class/interface/abstract class as Datatype.**

**Means predefined classes we can use as data types and user defined classes also we can use as datatypes.**

|  |
| --- |
| public **int** *i* = 0; public **String** *name* = null; public **PrintStream** *out* = null; public **Calculator** *cal* = null; |

Data types:

1. Primitive datatype. 🡪 **[BSIL, float, double, char, boolean]**
2. Non-Primitive datatype. 🡪 Wrapper classes + All predefined (classes/interfaces/abstract classes) +

All user defined (classes/interfaces/abstract classes) + arrays….etc.

