Java contents:

1. **Classes and Objects**

A class in Java is a blueprint which includes all your data.  A class contain fields(variables) and methods to describe the behavior of an object

An object is a major element in a class which has a state and behavior. It is an instance of a class which can access your data.

**public** **class** Student {

String name ="Vinay" ; //global variable

**public** **void** display() //method

{

System.***out***.println("Name displayed: "+name);

}

**public** **static** **void** main(String[] args) {

Student student = **new** Student() ; //// Created an object

student.display(); // call method

}

}

Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

### What is Class?

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

* **Class** - Dogs
* **Data members** or **objects**- size, age, color, breed, etc.
* **Methods**- eat, sleep, sit and run.

**public** **class** Dog {

// Instance Variables

String breed;

String size;

**int** age;

String color;

**public** String getDogInfo() {

**return** ("Breed is: "+breed +"\nSize is:"+size+"\n Age is:"+age+"\n color is: "+color);

}

**public** **static** **void** main(String[] args) {

Dog dogObj = **new** Dog();

dogObj.breed="Maltese";

dogObj.size="Small";

dogObj.age=2;

dogObj.color="white";

System.***out***.println(dogObj.getDogInfo());

}

}

**Java Rules to create the Class file :**

* There can be only one public class per source file.
* A source file can have multiple non-public classes.
* The public class name should be the name of the source file as well which should be appended by **.java** at the end. For example: the class name is *public class Employee{}* then the source file should be as Employee.java.
* If the class is defined inside a package, then the package statement should be the first statement in the source file.
* If import statements are present, then they must be written between the package statement and the class declaration. If there are no package statements, then the import statement should be the first line in the source file.
* Import and package statements will imply to all the classes present in the source file. It is not possible to declare different import and/or package statements to different classes in the source file.

**public** **class** ClassRule1 {

**public** **static** **void** main(String[] args) {

System.***out***.println("There can be only one public class per source file");

FourClass anotherClass = **new** FourClass() ;

anotherClass.display();

}

}

/\*public class Anotherclass

{

}

public class ThridClass

{

}

\*/

**public** **class** ClassRule2 {

**public** **static** **void** main(String[] args) {

System.***out***.println("A source file can have multiple non-public classes");

FourClass anotherClass = **new** FourClass() ;

anotherClass.display();

}

}

**class** AnotherClass

{

**public** **void** syso() {

System.***out***.println(getClass().getName());

}

}

**class** ThridClass

{

}

**class** FourClass

{

**public** **void** display() {

System.***out***.println(getClass().getName());

}

}

/\*public class ClassRule4 {

public static void main(String[] args)

{

System.out.println("The public class name should be the name of the source file as well "

+ "which should be appended by .java at the end");

}

}\*/

Save as ClassRule3 : it will allow to save and but not able to compile

Classes have several access levels and there are different types of classes; abstract classes, final classes

# Variables in Java :

## Variables naming convention in java

1) Variables naming cannot contain white spaces, for example: int num ber = 100; is invalid because the variable name has space in it.  
2) Variable name can begin with special characters such as $ and \_  
3) As per the java coding standards the variable name should begin with a lower case letter, for example int number; For lengthy variables names that has more than one words do it like this: int smallNumber; int bigNumber; (start the second word with capital letter).  
4) Variable names are case sensitive in Java.

## Types of Variables in Java

There are **three types of variables** in Java.  
1) Local variable 2) Static (or class) variable 3) Instance variable

## Local Variable:

These variables are declared inside method of the class. Their scope is limited to the method which means that You can’t change their values and access them outside of the method.

**public** **void** display()

{

**int** local = 20 ;

System.***out***.println("Local variable access only inside display method"+ local);

}

**public** **static** **void** main(String[] args) {

LocalVariable lobj = **new** LocalVariable();

lobj.display();

**int** local = 30 ;

System.***out***.println("Local varable access only inside main method "+ local);

}

## Instance variable :

## Each instance(objects) of class has its own copy of instance variable.

**public** **class** InstanceVariable {

String objInstVar = "Instance variable" ;

**public** **static** **void** main(String[] args) {

InstanceVariable obj1 = **new** InstanceVariable() ;

InstanceVariable obj2 = **new** InstanceVariable() ;

InstanceVariable obj3 = **new** InstanceVariable() ;

System.***out***.println(obj1.objInstVar);

System.***out***.println(obj2.objInstVar);

System.***out***.println(obj3.objInstVar);

obj1.objInstVar ="Modified Instance" ;

System.***out***.println(obj1.objInstVar);

System.***out***.println(obj2.objInstVar);

System.***out***.println(obj3.objInstVar);

}

## }

## Static (or class) Variable :

## Static variables are also known as class variable because they are associated with the class and common for all the instances of class.

**public** **static** String *companyName* ="Synechron" ; //class or static variable

**public** **static** **void** main(String[] args) {

StaticClassVariable obj1 = **new** StaticClassVariable() ;

StaticClassVariable obj2 = **new** StaticClassVariable() ;

StaticClassVariable obj3 = **new** StaticClassVariable() ;

System.***out***.println(obj1.*companyName*);

System.***out***.println(obj2.*companyName*);

System.***out***.println(obj3.*companyName*);

obj1.*companyName* ="Synechron-GTP" ;

System.***out***.println(obj1.*companyName*);

System.***out***.println(obj2.*companyName*);

System.***out***.println(obj3.*companyName*);

System.***out***.println("==========Class variable or Static =============");

System.***out***.println(*companyName*);

System.***out***.println(*companyName*);

System.***out***.println(*companyName*);

}

}

# Data Types in Java :

## java we have two categories of data type: 1) Primitive data types 2) Non-primitive data types – Arrays and Strings are non-primitive data types

## 

## 1) Primitive data types:

## we have eight primitive data types: boolean, char, byte, short, int, long, float and double. Java developers included these data types to maintain the portability of java as the size of these primitive data types do not change from one operating system to another:

**public** **class** PrimitiveDataTypes {

**public** **static** **void** main(String[] args) {

System.***out***.println("8 PrimitiveDataTypes ");

**byte** num =123 ; //-127 to -128

**short** val =12345 ; //4 byte

**long** number = 123456789L ; //15 digit will hold

**int** a =10 ;

**float** b = 20.0f ;

**char** c ='A' ;

**double** d = 33.33 ;

**boolean** status = **false** ;

System.***out***.println("Value of Byte:="+ num);

System.***out***.println("Value of short:="+ val);

System.***out***.println("Value of long:="+ number);

System.***out***.println("Value of Integer:="+ a);

System.***out***.println("Value of Float:="+ b);

System.***out***.println("Value of char :="+ c);

System.***out***.println("Value of Double:="+ d);

System.***out***.println("Value of Boolean:="+ status);

}

}

## 

## ****Java Data Operators:****

There are mainly 4 different types of operators, which are listed below:

* **Arithmetic Operator:** Perform arithmetic operations such as addition, subtraction, multiplication, division and modulus.
* **Unary Operator:** Unary operators are used to increment or decrement a particular value. For example: ++ stands for increment, – – stands for decrement.
* **Relational Operator:** It defines some kind of relation between two entities. For example: <, >, <=, >=, !=, ==.
* **Logical Operator:** Logical operators are typically used with boolean (logical) values.