Java Variables:-

- ✓ *Variables are used to store the constant values by using these values we are achieving project requirements.*
- ✓ *Variables are also known as **fields** of a class or **properties** of a class.*
- ✓ *All variables must have a type. You can use primitive types such as int, float, boolean, etc. Or you can use reference types, such as strings, arrays, or objects.*
- ✓ *Variable declaration is composed of three components in order,*
  - *Zero or more modifiers.*
  - *The variable type.*
  - *The variable name.*

  Example : public final int x=100;

public int a=10;

public *----> modifier (specify permission)*
int *----> data type (represent type of the variable)*
a *----> variable name*
10 ----> constant value or literal;
; ----> statement terminator

There are three types of variables in java

1. ***Local variables.***
2. ***Instance variables.***
3. ***Static variables.***

Local variables:-

- ❖ *The variables which are declare inside a **method or constructor or blocks** those variables are called local variables.*

```
class Test
{       public static void main(String[] args)      //execution starts from main method
        {       int a=10;                   //local variables
                int b=20;
                System.out.println(a);
                System.out.println(b);
        }
}
```

- ❖ *It is possible to access local variables only inside the method or constructor or blocks only, it is not possible to access outside of method or constructor or blocks.*

```
void add()
{       int a=10;           //local variable
        System.out.println(a);    //possible
}
void mul()
{         System.out.println(a);  //not-possible
}
```

- ❖ *For the local variables memory allocated when method starts and memory released when method completed.*

Instance variables (non-static variables):-

- ✓ *The variables which are declare inside a class but outside of methods those variables are called instance variables.*
- ✓ *The scope (permission) of instance variable is inside the class having global visibility.*
- ✓ *For the instance variables memory allocated during object creation & memory released when object is destroyed.*
- ✓ *Instance variables are stored in heap memory.*

Areas of java language:-

There are two types areas in java.

1) *Instance Area.*
2) *Static Area.*

Instance Area:-

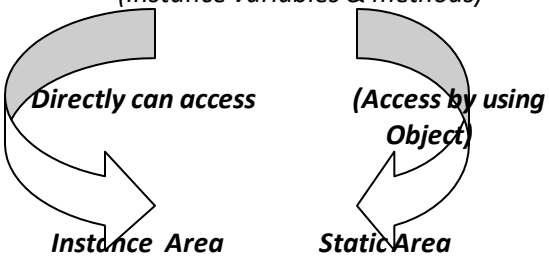
```
void m1() //instance method
{       Logics here //instance area
}
```

Static Area:-

```
Static void m1() //static method
{       Logics here //static area
}
```

Instance variable accessing:-

(Instance variables & methods)



Example:-

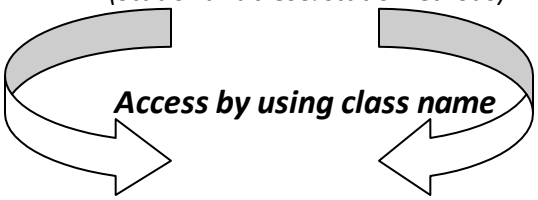
```
class Test
{       //instance variables
        int a=10;
        int b=20;
        //static method
        public static void main(String[] args)
        {       //Static Area
                Test t=new Test();
                System.out.println(t.a);
                System.out.println(t.b);
                t.m1();     //instance method calling

        }
        // instance method
        void m1()      //user defined  method must called by user inside main method
        {       //instance area
                System.out.println(a);
                System.out.println(b);
        }//main ends
};//class ends
```

Static variables (class variables):-

- *The variables which are declared inside the class but outside of the methods with static modifier those variables are called static variables.*
- *Scope of the static variables with in the class global visibility.*
- *Static variables memory allocated during .class file loading and memory released at .class file unloading time.*
- *Static variables are stored in non-heap memory.*

Static variables & methods accessing:-

(Static variables& static methods)




```
class Test
{       //static variables
        static int a=1000;
        static int b=2000;
        public static void main(String[] args)      //static  method
        {       System.out.println(Test.a);
                System.out.println(Test.b);
                Test t = new Test();
                t.m1();             //instance method calling
        }
        //instance method
        void m1()           //user defined  method called by user inside main method
        {       System.out.println(Test.a);
                System.out.println(Test.b);
        }
};
```

Static variables calling: - We are able to access the static members inside the static area in three ways.

- ✓ *Direct accessing.*
- ✓ *By using class name.*
- ✓ *By using reference variable.*

In above three approaches second approach is best approach .

```
class Test
{       static int x=100;           //static variable
        public static void main(String[] args)
        {       System.out.println(a);              //1-way(directly possible)
                System.out.println(Test.a);         //2-way(By using class name)
                Test t=new Test();
                System.out.println(t.a);            //3-way(By using reference variable)
        }
};
```

Example: - When we create object inside method that object is destroyed when method completed, if any other method required object then create the object inside that method.

```
class Test
{       //instance variable
        int a=10;
        int b=20;
        static void m1()
        {       Test t = new Test();
                System.out.println(t.a);
                System.out.println(t.b);
        }
        static void m2()
        {       Test t = new Test();
                System.out.println(t.a);
                System.out.println(t.b);
        }
        public static void main(String[] args)
        {       Test.m1();      //static method calling
                Test.m2();      //static method calling
        }
};
```

Example:-

```
class Test
{       int a=10;       int b=20;       // instance variables
        static int c=30;  static int d=40;  //static variables
                void m1() //instance method
        {       System.out.println(a);
                System.out.println(b);
                System.out.println(Test.c);
                System.out.println(Test.d);
        }
        static void m2()  //static method
        {       Test t = new Test();
                System.out.println(t.a);
                System.out.println(t.b);
                System.out.println(Test.c);
                System.out.println(Test.d);
        }
        public static void main(String[] args)
        {       Test t = new Test();
                t.m1();  //instance method calling
                Test.m2();      //static method calling
        }
};
```

Variables VS default values:-

Case 1:- for the instance variables JVM will assign default values.

```
class Test
{       int a;
        boolean b;
        public static void main(String[] args)
        {       //access the instance variables  by using object
                Test t=new Test();
                System.out.println(t.a);
                System.out.println(t.b);
        }
};
```

Case 2:- for the static variables JVM will assign default values.

```
class Test
{       static int a;
         static float b;
        public static void main(String[] args)
        {       //access the static variable  by using class Names
                System.out.println(Test.a);
                System.out.println(Test.b);
        }
};
```

Case 3:-

- *For the instance and static variables JVM will assign default values but for the local variables the JVM won't provide default values.*
- *In java before using local variables must initialize some values to the variables otherwise compiler will raise compilation error "variable a might not have been initialized"***.**

```
class Test
{       public static void main(String[] args)
        {       //local variables (access directly)
                int a;
                int b;
                System.out.println(a);
                System.out.println(b);
        }
};
```

D:\>javac   Test.java

Test.java:6: variable a might not have been initialized

System.out.println(a);

Class Vs Object:-

- ***Class is a logical entity it contains logics where as object is physical entity it is representing memory.***
- *Class is blue print it decides object creation without class we are unable to create object.*
- ***Based on single class (blue print) it is possible to create multiple objects but every object occupies memory.***
- *Civil engineer based on blue print of house it is possible to create multiple houses in different places but every house required some area.*
- ***We are declaring the class by using class keyword but we are creating object by using new keyword.***
- *We are able to create object in different ways like*
    - *By using new operator*
    - *By using clone() method*
    - *By using new Instance()*
    - *By using factory method.*
    - *By using deserialization….etc*

  But we are able to declare the class by using class keyword.
- ***We will discuss object creation in detailed in constructor concept.***

Instance vs. Static variables:-

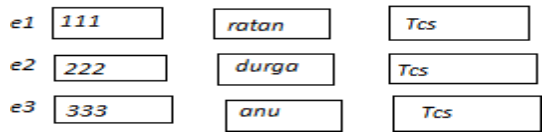
- *For the instance variables the JVM will create separate memory for each and every object it means separate instance variable value for each and every object.*
- *For the static variables irrespective of object creation per class single memory is allocated, here all objects of that class using single copy.*

Example :-

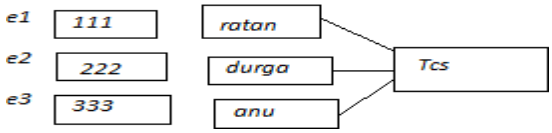
```
class Test
{       int a=10;           //instance variable
        static int b=20;  //static variable
        public static void main(String[] args)
        {       Test t = new Test();
                System.out.println(t.a); //10
                System.out.println(t.b); //20
                t.a=111;          t.b=222;
                System.out.println(t.a);  //111
                System.out.println(t.b);  //222
                Test t1 = new Test();      //10 222
                System.out.println(t1.a);          //10
                System.out.println(t1.b);          //222
                t1.b=444;
                Test t2 = new Test();      //10 444
                System.out.println(t2.b);          //444
        }
}
```

Instance variable vs static variable :-

```
class Emp
{       //instance variable
        int eid;
        String ename;
        String company;
}
```




```
class Emp
{       //instance variable
        int eid;
        String ename;
        //static variable
        static String company;
}
```



Different ways to initialize the variables :-

Summary of variables:-

Characteristic	***Local variable***	***instance variable***	***static***
variables where declared	*inside method or Constructor or block.*	*inside the class outside Of methods*	*inside the class outside of methods .*
Usage	*within the method*	*inside the class.*	*inside the class all*
When memory allocated	*when method starts*	*when object created*	*when .class file loading*
When memory destroyed	*when method ends.*	*When object destroyed*	*when .class unloading.*
Initial values	none, must initialize the value *before first use.*	default values are *Assigned by JVM.*	*default values are Assigned by JVM.*
Relation with Object	*no way related to object.*	*for every object one copy Of instance variable created means memory.*	*for all objects one copy is created. It Single memory.*
Accessing	*directly possible.*	*By using object name.* ***Test t = new Test();*** ***ut.println(Test.a); System.out.println(t.a);***	*by using class name.* ***System.o***
Memory	*stored in stack memory.*	*Stored in heap memory*	*non-heap memory.*