Lecture 25

Scope of variable

- Every variable has a scope.
- The scope defines where a certain variable is accessible in a program. and if you try to access a variable outside of its scope, you will get a compilation error.

① Class level Scope:- Variables declared inside a class but outside any method, block or constructor are known as class level variables and these variables are accessible within entire class meaning accessible by all the methods in that class.

. They can be directly accessible anywhere in the class.

e.g:-

```
Class VariableScopeDemo {
    int a;
    int b = a;
    void display() {
        System.out.println(a);
        System.out.println(c);   //no error
    }
    int B = c;   //error
    int c;
}
```

By the time display() method will be involved, all the class level variables are already initialized.

There are 2 type of class level variables:-

① Static
② Instance

Static Variable :- declared with the static keyword inside a class but outside any method, constructor or block.

- accessible by all instances (objects) of the class and remain in memory as long as class is loaded.

Instance Variable :- These are non-static variables declared within a class but outside any method, constructor or block.

- These are accessible throughout the class and belongs to the instance of the class.
- These are created when object is created & destroyed when the object is destroyed.

Static	Instance
• Requires "static" keyword	No "static" keyword
• Value is stored in a special memory called "Method Area" or "class Area" & this area is shared by all the class instances & is allocated when the class is loaded into memory	Value of instance variable is not allocated any memory & has no value until the object is created. Separate memory allocation for each object
• Can be accessed using classname classname.variablename	accessed by object of class. Objectname.variablename
• Belongs to class	belongs to each object

Local Scope :-

Variables declared within a method or block are local variables & these variables are only accessible within that specific method or block. They are destroyed once the block or method ends and they don't retain their value beyond that

e.g:- (i)
```
public class Demo {
    void display()
    {   int a = 5;
        System.out.println(a);
    }
}
```
a can be used inside this method only

if we want to access this a like :-
```
Demo obj = new Demo();
System.out.println(obj.a); //error.
```

e.g:- (ii)
```
public class Demo {
    void display() {
        int a = 5;
        System.out.println(b); // error
        int b = a;
        System.out.println(b);
    }
}
```
Scope of a

Scope of b

So scope of local variables are from the declaration point to the end of block.

Block Scope:- Variables declared within a specific block of code (like if, for, while) have a limited scope to that block only

- These are not accessible outside the block where they are defined.
- In Java local & Block Scope are closely related but not exactly the same.
- A block is defined by a pair of braces { ... }

e.g:-

```
void display() {
    int a = 5;
    if (a == 5) {
        int blockVar = 10;
        System.out.println(blockVar);
    }
    System.out.println(a);
    System.out.println(blockVar); //error
}
```

} Scope of blockVar

(ii)

```
void display() {
    int a = 5;
    {
        int b = 10;
        System.out.println(b);
    }
    System.out.println(b); //error
}
```

(iii)

```
void display() {
    int a = 5; int i = 0;
    for (int i = 0; i <= 5; i++) {

    }
```

error redeclaration is not allowed.

but if write:

```
    for (int j = 0; j <= 5; j++) {
        System.out.println(j);
    }
    System.out.println(j); //error out of scope.
}
```