DAY-8

Garbage Collection: (JVM Inside)

Automatic process of eliminating unwanted obj from memory.

C/C++ manually delete : Java automatically delete

Making Ref NULL

```
Car car1 = new Car();
car1 = null;
```

Automatically del unused variable or value

Assign ref to other

```
Car car1 = new Car();
Car car2 = new Car();
car1 = car2;
```

Anonymous obj

```
myMethod(new Car());
```

Finalize():

It does not return, Protected define.

Before del obj → Give several task.

```
void protected finalize() {
    // code
}
```

when obj doesn't deallocate finalize doesn't work

Packages: Store in Hierarchical Manner → Folder inside Folder inside Folder.

Packages are containers for the classes. Used to keep the class name in compartment.

Packages just a folder.

Cannot have two file name with something.

Eg:

```
package com.Priya.packages.a;
public class Greeting{
    public static void main(String[] args){
        System.out.println("Hello");
    }
}
```

instead of a we can create b to use same class name.

Compartment for folders.

import:

import the path of folder from one to another.

Eg:

```
package com.priya.package.a;
import static com.priya.
        packages.b.Msg.message;
public class Greeting {
    public static void main
            (String[] args) {
        System.out.println("Hello");
        message();
    }
}
```

This msg is imported Here

```
package com.priya.packages.b;
public class Msg {
    public static void main
            (String[] args) {
    }
    public static void message() {
        System.out.println("This
            is amazing");
    }
}
```

Private file we cannot call this packages in a

O/p:

Hello
This is amazing

Static:

static variable is variable that belongs to the class itself.

Eg:

```
package com.kunal.staticeg;
public class Human {
    int age;
    String name;
    static long population;
    public Human(int age, String
            name) {
        this.age = age;
        this.name = name;
        Human.population += 1;
    }
}
```

Static long should be used

To call static we couldn't use this.

Instead we use class name

Eg:

```
package com.kunal.staticeg;
public class Main {
    public static void main
            (String[] args) {
        Human kunal = new Human
            (age:22, name:"Kunal");
        Human priya = new Human
            (age:21, name:"Priya");
        System.out.println(kunal.
            population);
        System.out.println(priya.population);
    }
}
```

While using this we can get (1) as a O/p. long variable Instead of static we get 1 O/p

In this while assigning Human.population we can add & get the value.

Static population initial value = 0

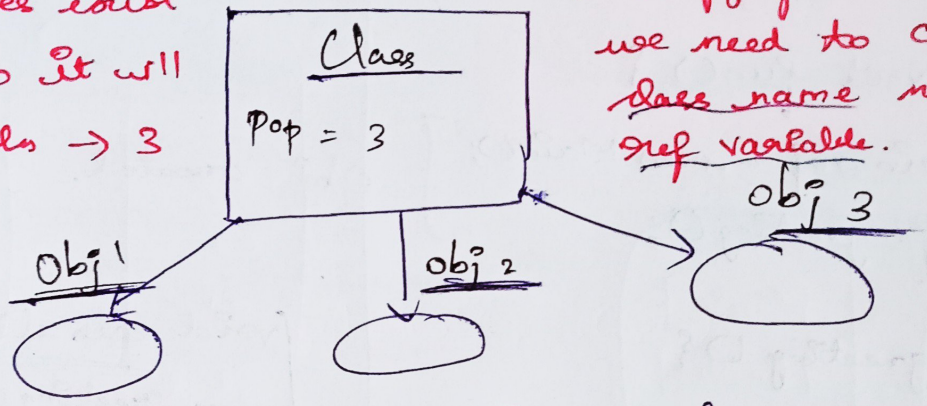
o/p : 2
2

this.population = 2

⇒ Whenever we create with class name it will directly update the class.

NOTE:
Whenever accessing or modifying static variable we need to call by class name not with ref variable.

Population does exist in Human. So it will print variables → 3



System.out.println(Human.population); Should be assigned.

Static does not depend on obj. We can access static variable. Static belong to class

Non-static member inside a static:

```
public class Main {
    public static void main(String[] args){
        greeting();  → We cannot use
    }                  Only access static data
    void greeting() {
        ---
    }
```

Belongs to obj
Not static

public static void main
⇒ Use main fun with obj → class
Main → mandatory
Static only access static data
Static not access non-static data
Non-static belongs to instance

```
    static void fun(){
        greeting();
    }
```

Require obj

Not dependent on obj

But function does not depend on instance or obj

Static does not allow to do it.

Error

Static member inside non-static:

Does not limit itself

```
void greeting(){
    fun();
    System.out.println("Hello");
}
```

Not static obj will be required.

We cannot access non-static variable without refering their instance

We make sure to assign with obj to access non-static.

Static part of class
not part of obj

⇒
```
static void fun(){
    Main obj = new Main();
    obj.greeting();
}
void greeting(){
    System.out.println("Hello");
}
```

obj created.

Static doesn't need an obj to run
So we are creating obj in it.

Fun doesn't need to create obj.

```
void fun2(){
    greeting();
}
```

For function doesn't need to create obj.

But here fun take care of greeting,
Both need an obj to run.
But doesn't create.

Initialization of static variable:

```
public class StaticBlock {
    static int a=4;
    static int b;
    static {
        System.out.println("I am in Static");
        b = a*5;
    }
    public static void main(String[] args){
        StaticBlock obj = new StaticBlock();
        System.out.println(StaticBlock.a + " " + StaticBlock.b);
        StaticBlock.b += 3;   ⇒ 23
        System.out.println(StaticBlock.a + " " + StaticBlock.b);
```

Print only once.

```
StaticBlock obj2 = new StaticBlock();
System.out.println(StaticBlock.a + " " + StaticBlock.b);
}
}
```

Even we assign another obj & Print again, it will display the same result as before

Op:

I am in Static
A 20
A 23] → Same result
A 23

Inner Class

Only inner classes are static

```
Static Class A{   Error
}
```

Eg:

```
public class Innerclass{
    static class Test{
        static String name;
        public Test(String name){
            this.name = name;
        }
    }
    public static void main(String[] args){
        Test a = new Test(name: "Priya");
        Test b = new Test(name: "Dharshini");
        S.o.p(a.name);
        " (b.name);
    }
}
```

Depend on this class Innerclass

If this is static, no error occurs.

Here

Error

Op: Priya
Dharshini

Static variable will assign same value to all.