Static var, fun, ~~class~~ block ***.

Unit-2

- static kw can be used with var, fun, ~~class~~, block, class & import

Static Var → All obj will share same copy of the static var

static fun → st. fun can be accessed w/o creating an obj. It can be called directly using class name.

Static block → St. block is similar to a constr.
- It runs autoly when a class is first used.
- It can be used to initialize the var.

Q) Why is st. block used, if it is similar to constr?

A) When st. kw is used, then obj is not created. So constr will not be called. ∴ St. block is used.

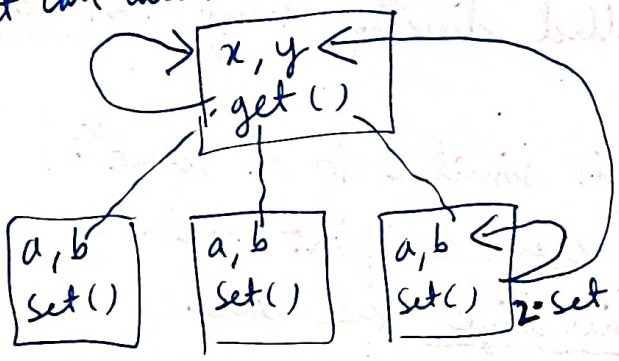
Rule w/ st. fun

- St. fun can call only other st. var & fun.
- Non st. fun can access all " var & fun.

eg var x, y r static
" a, b r normal.

fun set is normal - it can access all var a, b, x, y
" get " static - " " " only static var x & y.

1. get can access only static var



2. set can access st. & non st. variables

obj1 obj2 obj2.

// all obj have same copy of x, y (static)
" " " obj " " a, b (Normal).

```
class box {
    protected static int l, w, h; //st. var.
    static {
        l=0; w=0; h=0;  //static block.
    }
    public static void set(int x, int y, int z) {
        l=x; w=y; h=z;   //st. fun
    }
    public static void get() {
        Sop1n(l); Sop1n(w); Sop1n(h); //st. fun.
    }
}
C D { P S V M (-.) {
    //No need to create obj of box.
    //static fun can be called using class name
    box.set(1, 2, 3);
    box.get();
  }
}
```

Nested Classes

- A class created / defined inside another class is called a nested class.
- Inner class can directly access var of outerclass but outerclass can't " " " " innerclass.
- Outerclass need to create obj of innerclass to access its variables.

```
outerclass {
        PV outerfun() {
PV -> public void      // define
        }

    class innerclass {
            PV innerfun() {
                // define
                outerfun();  --> innerclass can directly call outerfun.
            }
    }

    innerclass obj = new innerclass();
    obj.innerfun();  --> outerclass needs to create obj of innerclass to call innerfun.
}
```

Can't call innerfun directly.

Creating obj of innerclass in demo class

- Obj of a simple class is created inside main()

  eg box obj = new box();

- However, obj of an innerclass can't be created directly
- 1st we need to create obj of outer class.
  Then we can create obj of innerclass using outer class.

```
eg  class outerclass {
        pv outerfun() {
            sopln("outerfun");
        }
PV→ public void
        class innerclass {
            pv innerfun() {
                sopln("innerfun");
            }
        }
    }
    cl{ psvm(...){
```

// to create obj of innerclass, 1st we need to create obj of outer class.

```
outerclass obj1 = new outerclass();
outerclass.innerclass obj2 = obj1.new innerclass();
```

- obj1 ← create outer obj
- outerclass.innerclass ← create inner obj; class name
- obj2 ← inner obj
- obj1 ← outer obj
- new innerclass(); ← innerclass.

```
obj1 → outer obj
obj2 → inner "
// call fun using inner obj
   obj2.innerfun();
  }
} // demo ends.
```

Static Class

- An innerclass can be declared as static
- The purpose of static innerclass is that its obj can be created w/o creating obj of outerclass.

Note: outer class can't be st.
Only inner class can be static.

```
outerClass {
        PV outerfun() {
            // define
        }
        class static innerclass {
                PV innerfun() {
                    //define
                }
        }
}
```

CD { PSVM (---) } → class demo {

Pub st. void main(..) {

//No need to create obj of outerclass.

// obj of innerclass can be created directly

outerclass.innerclass obj = new outerclass.innerclass();

class name → innerclass obj

class name

***

//Here innerclass is static, so its obj can be created directly w/o creating obj of outerclass.

Import & Static Import

Unit-2

- We can import any class & use it.

eg import java.lang.System;

```
class D {psvm() {
        System.out.println();
    use System class after importing
    }
}
```

Note:- System class is auto imported, but it can be manually imported too.

What does System.out.println() mean

```
class System{
static PrintStream out = new PrintStream();

}

//System is a class
//out is an obj inside System.
//out is not an obj of System, but out is an
    obj of PrintStream class.
//out is static, so, it can be accessed by using
    class name
        System.out....
```

What is static import

- Another way to implement a class through static import

How to statically import

1) Use static kw

import static java.lang.System.*;

2) When statically importing, use * after class name wh. means all members of a class are imported & not just a particular class.

What happens when u statically import a class? Benefit?

- We can access members w/o using class name

eg System.out.println(); //Normal import
out.println(); //with static import, class name is not needed.

- So, it reduces effort of typing classname everytime.

import	vs	Static import
• Syntax: & import <class>		import static <class>.* eg:- import static java.lang.System.*; * means all members of class.
eg import java.lang.System;		
• class name must be used System.out.println();		• members can be accessed w/o class name out.println();

Note :-

- St. import doesn't make a class static or it doesn't make the members static.
- The class members r already static.
- It simply allows to use the members directly w/o using class name.