Constructor, finalize () & Constructor overlo· overloading

Unit - 1

Q) What r constructor & finalize ()

A) • Constructor r special fun wh. runs auto'y when an obj is created.

• Const. can be used to initialize an obj.

• finalize () is similar to destructor
There r no destructor in java.

• It is always named as finalize & not as class name.

• Finalize is called when ~~a~~ the obj is destroyed by GC.

• Since GC is unpredictable, so finalize () method also runs unpredictably.

• Declaration of finalize

$$\begin{tikzcd}
\text{protected} \arrow[d] & \text{void} \arrow[d] & \text{finalize} \arrow[d] & \text{( )} \arrow[d] \\
\text{access specifier} & \text{return type} & \text{name} & \text{No param.}
\end{tikzcd}$$

Eg Create const. & finalize() in box.

```
class box {
    ~~prot~~ private int l, w, h;
    public void set(int x, int y, int z) {
            l = x;
            w = y;
            h = z;
    }

    public void get() {
            Sopln(l);
             Sopln(w);
              Sopln(h);
    }
    box(int x, int y, int z) { // box constr.
            l = x;
            w = y;
            h = z;
    }
    protected void finalize () {
            Sopln(" finalize called");
      //This will run when obj is destroyed.
    }
}
```

```
class demo{
    public static void main(String args[]){
        box obj = new box(1,2,3); //must pass values to const
        obj.set(5,6,7);
        obj.get();
    }
}.
```

Feat. of finalize

1) Declared as
   protected void finalize();
2) Runs when GC releases the m/m of obj.
   So it is unpredictable
3) It is called exactly once for each obj.
4) Used to perform clean up task like
   close file stream, n/w connection etc.

Feat. of a Const

1) Name sud be same as class name.
2) Return type → no return type.
3) Keyword that can be used with const.
   - abstract, static, final, synchronized.
4) access specifier that can be used -
   private, public, protected, ~~void~~ default.
5) Can't be inherited, but can be invoked using super().

Constructor	Other member fun.
• Use - to initialize an obj	- Can perform any task.
• No return type	- must have
• Called auto^y when obj is created	- must be called by obj eg obj.set().
• name - same as class name	- any name.
• Can be called exactly once	- any no. of times
• Can be called only at the time of obj creation	- any time.

Types of const.

1) <u>Def. const.</u>

- When no const. is created by user, then java p/v a ~~def~~ def. const. It is non parameterized & do not perform any task.

2) <u>Non parameterized const</u>

- It is user defined const. wh. has no param. It can perform some task. Def const. doesn't perform any task.

3) <u>Parameterized const</u>

- User defined const. with param.

4) <u>Copy const.</u>

- ~~Const.~~ User defined const. wh. copies one obj to another. It takes obj as paramter.

Constructor overloading

- Const O/L'g means to create mul const with diff parameters.
- Eg Box class can have following const.

```
box( ); // Non parameterized Const.
box(int x); // 1 param.
box(int x, int y, int z) // 3 param.
box(obj); // obj as param.
```

$\underbrace{\text{(obj)}}_{\text{copy. const.}}$

```
Class box {
    private int l, w, h;
    public void set (int x, int y, int z) {
              l=x;
              w=y;
              h=z;
    }
    public void get () {
              sopln(l);
              sopln(w);
              sopln(h);
    }
```

~~public void~~

```
box() { //Non param const.
        Sopln("Non param Const called");
}
box(int x) { // 1 param const.
        l=x;
        w=x;
        h=x;
}
box(int x, int y, int z) { //3 param.
          l=x;
          w=y;
          h=z;
}
box(box obj) { //Copy const.
    //Copy obj (passed obj) to the calling obj.
    //var of calling obj r l, w, h.
    //"  " passed obj r obj.l, obj.w, obj.h;

        l = obj.l;                  //~~Calling~~ passed obj copied
        w = obj.w;                    to ~~passing obj~~ calling obj.
        h = obj.h;

   Calling obj      passed obj
}
}.//box class.
```

```
Class demo {
    public static void main (String args[]) {
        box obj1 = new box(10); //const w/ 1 param
        box obj2 = new box(1,2,3); // 3 param
        box obj3 = new box(); // No param.
        box obj4 = new box(obj1); // copy const.

                 obj1 copied to obj4.
                  ↓              ↓
             passed obj     calling obj.

    }
3.
```