OOP-3

Static and non-static members

Java static keyword

The **static keyword** in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than an instance of the class.

JAVA Static variable:

If you declare any variable as static, it is known as a static variable.

The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.

It makes your program **memory efficient** (i.e., it saves memory).

Static variables

```
class Student{
    int rollno;
    String name;
    String college="ITS";
}
```

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all objects. If we make it static, this field will get the memory only once.

Static variables

```
//Java Program to demonstrate the use of static variable
class Student{
   int rollno;//instance variable
   String name;
   static String college ="ITS";//static variable
   //constructor
   Student(int r, String n){
   rollno = r;
   name = n;
   }
   //method to display the values
   void display (){System.out.println(rollno+" "+name+" "+college);}
}
```

Output:

```
111 Karan ITS
222 Aryan ITS
```

Static variables

Program of the counter without static variable

```
class Counter{
int count=0;//will get memory each time when the instance is created

Counter(){
count++;//incrementing value
System.out.println(count);
}

public static void main(String args[]){
//Creating objects
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
}
}
```

Output:

```
1
1
1
```

Static variables

Program of counter by static variable

```
class Counter2{
static int count=0;//will get memory only once and retain its value

Counter2(){
count++;//incrementing the value of static variable
System.out.println(count);
}

public static void main(String args[]){
//creating objects
Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();
}
}
```

Output:

```
1
2
3
```

Static vs non static variables

STATIC VARIABLE	NON STATIC VARIABLE
Static variables can be accessed using class name	Non static variables can be accessed using instance of a class
Static variables can be accessed by static and non static methods	Non static variables cannot be accessed inside a static method.
Static variables reduce the amount of memory used by a program.	Non static variables do not reduce the amount of memory used by a program
Static variables are shared among all instances of a class.	Non static variables are specific to that instance of a class.
Static variable is like a global variable and is available to all methods.	Non static variable is like a local variable and they can be accessed through only instance of a class.

Static methods

If we apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

Java program to call a static method

```
class Test {
   // static method
   public static int sum(int a, int b)
   {
      return a + b;
   }
}
```

Static methods

```
class Main {
   public static void main(String[] args)
   {
      int n = 3, m = 6;

      // call the static method
      int s = Test.sum(n, m);

      System.out.print("sum is = " + s);
   }
}
```

Output:
sum is = 9

Static and non static methods

Java program to call a non-static method

```
class Test{
        // static method
        public int sum(int a, int b)
        {
                return a + b;
        }
}

class Main {
        public static void main(String[] args)
        {
                int n = 3, m = 6;
                Test g = new Test();
                int s = g.sum(n, m);
                // call the non-static method
                System.out.print("sum is = " + s);
        }
}
```

Output:

sum is = 9

Static and non static methods

Non-Static methods

- A non-static method does not have the keyword *static* before the name of the method.
- A non-static method belongs to an object of the class and you have to create an instance of the class to access it.
- Non-static methods can access any static method and any static variable without creating an instance of the class.

Array of objects

An array of objects is created just like an array of primitive type data items in the following way.

```
Student[] arr = new Student[7]; //Student is a user-defined class
```

The created array contains seven memory spaces each of size of student class in which the address of seven Student objects can be stored. The Student objects have to be instantiated using the constructor of the Student class and their references should be assigned to the array elements.

Array of objects

Java program to illustrate creating an array of objects

```
class Student
{
    public int roll_no;
    public String name;
    Student(int roll_no, String name)
    {
        this.roll_no = roll_no;
        this.name = name;
    }
}
```

Cont...

```
public Test
{
   public static void main (String[] args) {
      //declaring and allocating memory for 5 objects of type Student.
      Student[] arr= new Student[5];

      // initialize the first elements of the array
      arr[0] = new Student(1,"aman");

      // initialize the second elements of the array
      arr[1] = new Student(2,"vaibhav");
      // so on...
      arr[2] = new Student(3,"shikar");
      arr[3] = new Student(4,"dharmesh");
      arr[4] = new Student(5,"mohit");
      // accessing the elements of the specified array
      for (int i = 0; i < arr.length; i++)
         System.out.println("Element at " + i + " : " +
                  arr[i].roll_no +" "+ arr[i].name);
   }
}
```

Cont…

Output:

Element at 0 : 1 aman
Element at 1 : 2 vaibhav
Element at 2 : 3 shikar
Element at 3 : 4 dharmesh
Element at 4 : 5 mohit