Class Method Object

- Class is blueprint of object
- object is instance of class
- With inclass we will write methods, variable...etc
- These are called as class members
- We can reuse class members by creating instance
- instance is also called as Object/Reference

- syntax of class

```
class classNAme {

}
```

- syntax of class with reusable method

```
class classNAme {

        public void demo(){

                method body;

        }

}
```

- A class with main method reusable and executable
- A class without main method is only reusable
- **Accessing Class Members**
- To access class members we have to create instance
- syntax for creating instance
  - classNAme variable = new className();
  - variable.classmember
- the variable is called reference variable
- because it stores the reference of the class
- We can create multiple instances for a class
  - classNAme variable1 = new className();
  - classNAme variable2 = new className();

Methods

- A method is reusable code block
- we can write some code with in a method and assign a name to it
- We can execute all the code of method by calling that method name in any other classes or in same class
- We can have parameters for a method

- If we create parameters we must pass the values while calling that method
- In this way we can make methods which works with dynamic values
- We can also create a constructor method which works like prerequisite for the class
- This method will be executed along with instance creation
- We don't need to call constructor method
- This should have same class name
- constructors doesn't return values
- We can pass values to a method. But the method must have been configured with parameters to accept the values. How many parameters created in that method definition those many values we can pass.
- We can pass any number values by passing an array if we create method parameters as array.
- There is one more facility to pass any number of values while calling a method. We can create method parameter as datatype...
- This will accept any number of values or an array while calling that method

```
// write a method for adding numbers of array
public void demoAddArray(int[] numbers) {

   int val = 0;
   for (int i = 0; i < numbers.length; i++) {
         val = val + numbers[i];
   }
   System.out.println(val);
}
```

- the above method accept array as the parameter

```
// write a method which will accept unlimited values, and add them
public void demoAddAllNumbers(int... numbers) {

      int val = 0;
      for (int i = 0; i < numbers.length; i++) {
            val = val + numbers[i];
      }
      System.out.println(val);
}
```

- the above method accept array as the parameter or any number of integer values as parameters
- printnumbers(10,20,30,40,11,23);
- **Returning values from method**
- void means method does not return any value
- replace void with any datatype to return that type of data
- we can use return keyword to return data
- return statement will become last executable statement in a method
- we can return multiple values by returning an array

```
public boolean isOdd(int n) {
      if(n%2==1) {
            return true;
      }else {
            return false;
      }
}

public boolean isPrime(int n) {
      for(int i=2;i<n/2;i++) {
            if(n%i==0) {
                  return false;
            }
      }

      return true;
}

public int[] getNumbers() {

      int[] arr = {10,20,30,40};
      return arr;
}
```

Static And Non Static (Instance)

- When a method doesn't have dependency on constructor or any other methods then we can create that method as static. Because accessibility becomes easier.
- Feature of static
  - Static class members can be accessed with or without creating instance.
  - We can access static class members using classname or instance.
    - Syntax: className.StaticClassMember
  - Non static class members should be accessed only with instance. That's why we call them as instance variables or methods
- functionality of static
  - static variables maintain single memory across all instances
  - You can update the variable value from one instance and take it from other instance. It will have the same value. What ever the latest value updated in any instance or by using class name that value will be accessed from all instances
  - Instance variables can maintain different values for different instances. These will have different memory locations for each instance.
  - We can use static variables as global variables across all instances
- Restriction of static
  - With in class we can access any class member from any non static method directly.
  - But from static method we can access only static class members directly. To access non static members we have to create instance and access them.
  - directly means without creating instance or using class name;