Lesson:

Constructor

List of Concepts Involved:

- Constructor
- DefaultConstructor
- Usage of Constructor
- Constructor Chaining
- Usage of this() and super()

Constructor

- Object creation is not enough, compulsorily we should perform initialization then only the object is in a position to provide the response properly.
- Whenever we are creating an object some piece of the code will be executed automatically to perform initialization of an object. This piece of code is nothing but a constructor.
- Main objective of the constructor is nothing but initialisation of Object.

Rules for writing a constructor

- Name of the constructor and name of the class must be the same.
- Return type concept not applicable for constructor, even if we provide it won't result in compile time errors, if we do so then the Java language will treat this as "normal method".

Eg

```
class Test{
        void Test(){
                System.out.println("Hello");// It is not a constructor,it is a method.
        }
}
```

- It is not a good practice to take the method name same as that of the classname.
- The modifiers applicable for constructors are private,public,protected,default.
- The other modifiers if we use, it would result in compile time error.

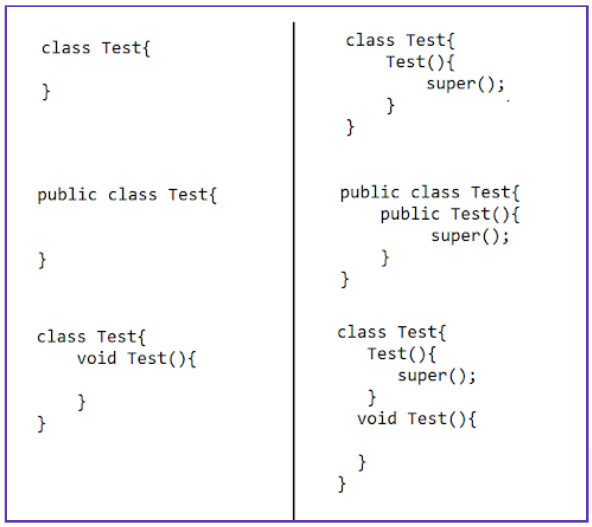
```
class Test{
        static Test(){
}
}
```

Default constructor

- For every java class constructor concept is applicable.
- If we don't write any constructor, then the compiler will generate a default constructor.
- If we write at least one constructor then the compiler won't generate any default constructor, so we say every java class will have a compiler generated default constructor or programmer written constructor but not both simultaneously.

Prototype of default constructor

- There is always no argument constructor.
- The access modifier of the default constructor is the same as the class modifier. [applicable for public and default]
- Default constructor contains one line, super(). It is a call to super class constructor.



```
class Test{

}
```

```
class Test{
    Test(){
        super();
    }
}
```

```
public class Test{

}
```

```
public class Test{
    public Test(){
        super();
    }
}
```

```
class Test{
    void Test(){

    }
}
```

```
class Test{
   Test(){
      super();
   }
  void Test(){

  }
}
```

Constructor Overloading/Constructor Chaining

- A class can contain more than one constructor and all these constructors have the same name they differ only in the type of argument, hence these constructors are considered as "Overloaded constructor".

Eg

```
class Test {
        Test(double d) {
                System.out.println("double argument constructor");
        }

        Test(int i) {
                this(10.5);
                System.out.println("int argument constructor");
        }

        Test() {
                this(10);
                System.out.println("no argument constructor");
        }
}
public class MainApp {
        public static void main(String[] args) throws Exception {
                Test t1= new Test();//double int no argument constructor
                Test t2= new Test(10);// double int argument constructor
                Test t3= new Test(10.5);//double argument constructor
        }
}
```

super() vs this()

1. The first line inside the constructor can be super()/ this().
2. If we are not writing anything then compiler will generate super();

case1:
We have to take super()/this() only in the first line of constructor, if we are writing anywhere else it would result in a compile time error.

eg1

```
class Test{
        Test(){
                System.out.println("Constructor");//CE
               super();
        }
}
```

eg2
we can either use super()/this() but not simultaneously

```
class Test{
        Test(){
              super();
              this();//CE
        }
}
```

eg3
we can use super()/this() only inside the constructor otherwise it would result in compile time error.

```
class Test{
        void methodOne(){
                super();
                this();
        }
}
```

Note

super()

- It should be the first line in the constructor.
- It should be used only in constructor.
- It will take control to the parent class constructor.

this()

- It should be the first line in the constructor.
- It should be used only in constructor.
- It will make the call of the current class constructor.

Difference b/w super(),this()?

super(),this()

- These are constructor calls
- These are used to invoke super class and current class constructor directly
- We should use only inside the constructor that to first line otherwise we get compile time error.