Generic Class

In this lesson, we explain how to create and use Generic Class Objects & functions of the Generic Class.

We'll cover the following

Generic objects & type members

As another powerful feature of Java, you can also make *Generic classes*, which are *classes* that can have *members* of the **generic** type.

Example

```
class Test<T>
{
    T obj;     // An object of type T is declared
    Test(T obj) // parameterized constructor
    {
      this.obj = obj;
    }
    public T getObject() // get method
    {
      return this.obj;
    }
}
```

Explanation

- We have declared a class `Test` that can keep the `obj` of any type.
- The constructor `Test(T obj)` assigns the value passed as a parameter to data member `obj` of type `T`.
- The get method `T getObject()` returns the `obj` of type `T`.

Syntax to instantiate object

To create objects of the generic class, we use the following syntax.

```
// To create an instance of generic class
Test <DataType> obj = new Test <DataType>()
```

Have a look at the detailed implementation of Generic Class and its methods.

```
// We use <> to specify Parameter type
class Test < T > {
    T obj;
    Test(T obj) {
        this.obj = obj;
    }
    public T getObject() {
        return this.obj;
    }
}
class Main {
    public static void main(String[] args) {
        // Test for Integer type
        Test < Integer > obj1 = new Test < Integer > (5);
        System.out.println(obj1.getObject());

        // Test for double type
        Test < Double > obj2 = new Test < Double > (15.777755);
        System.out.println(obj2.getObject());

        // Test for String type
        Test < String > obj3 = new Test < String > ("Yayy! That's my first Generic Class.");
        System.out.println(obj3.getObject());
    }
}
```

As you can see we create 3 different `Integer`, `Double` and `String` type variables for three different generic class objects.

You can play around with more datatypes to test the above class to check your understanding.

This marks the end of our discussion on *Generics*. Next up we'll look at some more coding challenges and will step through them in detail.