JAVA means DURGA SOFT
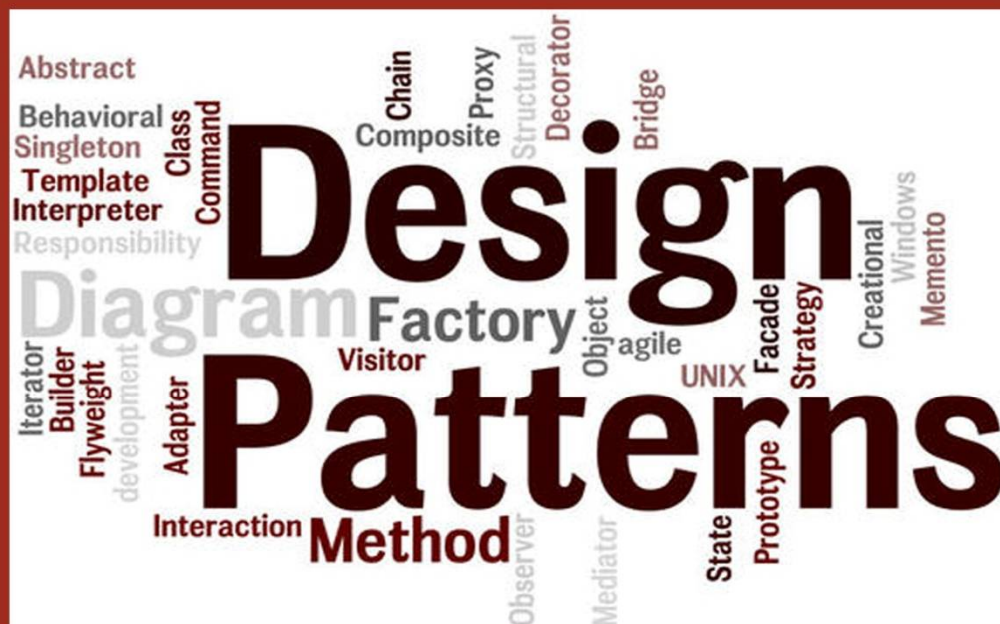
Design Patterns

Core Level Design Patterns

9. Adapter Class Design Pattern

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Adapter Class Design Pattern

Problem: If an interface contains huge number of methods then the implementation class of that interface should implement all the methods of that interface irrespective of programmer's interest otherwise that implementation class acts as **Abstract Class.**

Ex:

interface Test

{

      public void add();

      public void sub();

      public void mul();

}

Now if you implement above **Test** interface directly, then you have to implement all the methods of that interface.

public class ConcreteClass implements Test

{

      public void add()

      {

      }

      public void sub()

      {

```
        }
        public void mul()
        {
                ---
                ---
        }
}
```

Solution: Take a helper class (Adapter class) implementing that interface and providing **Null method** definitions for interface methods. If you make your class extending from this Adapter class (Helper class), you will be having freedom to implement only those methods in which you are interested in.

Note: Null Methods are the java methods which are having only method definition without implementation.

Ex:

```
public class AdapterClass implements Test
{
        public void add()
        {
        }
        public void sub()
        {
        }
        public void mul()
        {
        }
}
```

Now extend this **AdapterClass** and then override your required methods as shown below.

```
public class Add extends AdapterClass
{
        public void add()
        {
                ---
                ---
        }
}
public class Sub extends AdapterClass
{
        public void sub()
        {
                ---
```

```
        ---

    }

}
```

Since the Add, Sub classes are extending from AdapterClass so they got freedom to provide the definition for those methods of Test interface in which they are interested in.

Predefined Ex: java.awt.event.WindowAdapter, java.awt.event.KeyAdapter etc.

❖ Since we are place more null method definitions in AdapterClasses it is recomemded to declare AdapterClass as abstract class. So that no one can instantiate this AdapterClass directly (since no use by creating objects for class having null methods).

```
Eg: public abstract class AdapterClass implements Test
{
        public void add()
        {
        }
        public void sub()
        {
```

```
            }
            public void mul()
            {
            }
        }
```

Note: In general we also use this Adapter Class design pattern to make two unrelated interfaces can work together. The joining between them is called an Adapter. This is something like we convert interface of one class into interface expected by the client. We do that using an Adapter.

Application Areas: Use the Adapter pattern when

- You want to use an existing class, and its interface does not match the one you need.
- You want to create a reusable class that cooperates with unrelated or unforeseen classes, that is, classes that don't necessarily have compatible interfaces.