1) Strategy

- **Add.java**

```
package strategy;
public class Add implements St{
   @Override
   public double Op(int num1,int num2)
   {
      return (num1+num2);
   }
}
```

- **Xor.java**

```
package strategy;
public class Xor implements St
{
   @Override
   public double Op(int num1,int num2)
   {
      return (num1^num2);
   }
}
```

- **Context.java**

```
package strategy;
public class Context {
  private St strategy;
```

```
    public Context(St strategy){
      this.strategy = strategy;
    }
    public double executeStrategy(int num1, int num2){
      return strategy.Op(num1, num2);
    }
  }
```

- **St.java**

```
  package strategy;
  public interface St {
     public double Op(int num1,int num2);
  }
```

- **Strategy.java**

```
  package strategy;
  public class Strategy {
     public static void main(String[] args) {
        // TODO code application logic here
        Context context = new Context(new Add());
        System.out.println("10 + 5 = " + context.executeStrategy(1, 5));
        context = new Context(new Xor());
        System.out.println("10 ^ 5 = " + context.executeStrategy(1, 5));
     }
}
```

2) Observer

- **Add5.java**

```
package observerdemo;
public class Add5 extends Observer {
    public Add5(Subject subject)
   {
     this.subject = subject;
     this.subject.attach(this);
  }
  @Override
  public void update() {
     System.out.println( "Add 5: " +  subject.getState() +5 );
 }
}
```

- **Binary.java**

```
package observerdemo;
public class Binary extends Observer{
    public Binary(Subject subject)
   {
     this.subject = subject;
     this.subject.attach(this);
  }
  @Override
  public void update() {
     System.out.println( "Binary String: " + Integer.toBinaryString( subject.getState() ) );
  }     }
```

- **Subject.java**

```
package observerdemo;
import java.util.ArrayList;
import java.util.List;
public class Subject {
   private List<Observer> observers = new ArrayList<Observer>();
  private int state;

  public int getState() {
    return state;
  }
  public void setState(int state) {
    this.state = state;
    notifyAllObservers();
  }
  public void attach(Observer observer){
    observers.add(observer);
  }
  public void notifyAllObservers(){
    for (Observer observer : observers) {
      observer.update();
    }
  }
}
```

- **Observer.java**

```
package observerdemo;
public abstract class Observer {
   protected Subject subject;
```

```
    public abstract void update();
}
```

- **ObserverDemo.java**

```
package observerdemo;
public class ObserverDemo {
    public static void main(String[] args) {
        Subject subject = new Subject();
      new Binary(subject);
      new Add5(subject);
      System.out.println("First state change: 15");
      subject.setState(15);
      System.out.println("First state change: 15");
      subject.setState(10);
        // TODO code application logic here
    }
}
```

3) Decorator

- **Coffee.java**

```
package decoratordemo;
public interface coffee {
      public int cost();
}
```

- **Capacino.java**

```
package decoratordemo;
public class Capacino implements coffee{
private int cost=40;
@Override
public int cost()
{
   return cost;
}
}
```

- **BlackCoffee.java**

```
package decoratordemo;
public class BlackCoffee implements coffee{
   private int cost=50;
   @Override
   public int cost()
   {
      return cost;
   }
}
```

- **MocaCoffee.java**

```
package decoratordemo;
public class MocaCoffee extends CoffeeDecorator{
   public MocaCoffee(coffee mycoffee)
   {
      super(mycoffee);
   }
```

```
    @Override
    public int cost()
    {
        setcost(mycoffee);
        return mycoffee.cost()+20;
    }
    private void setcost(coffee mycoffee)
    {
        System.out.println("total cost is " + (mycoffee.cost()+20));
    }
}
```

- **CoffeeDecorator.java**

```
package decoratordemo;
public abstract class CoffeeDecorator implements coffee{
    protected coffee mycoffee;
    public CoffeeDecorator(coffee mycoffee)
    {
        this.mycoffee = mycoffee;
    }
    public int cost()
    {
        return mycoffee.cost();
    }
}
```

- **DecoratorDemo.java**

```
package decoratordemo;
public class DecoratorDemo {
```

```
    public static void main(String[] args) {
        // TODO code application logic here
        coffee newcoffee = new Capacino();
        coffee mocacapacino = new MocaCoffee(new Capacino());
        coffee mocablackcoffee = new MocaCoffee(new BlackCoffee());
        System.out.println("\nCapacino with Moca");
        mocacapacino.cost();

        System.out.println("\nBlackCoffee with Moca");
        mocablackcoffee.cost();
    }
}
```