Proxy Pattern
===========

```
public interface Image {
   void display();
}
```

RealImage.java

```
public class RealImage implements Image {

   private String fileName;

   public RealImage(String fileName){
      this.fileName = fileName;
      loadFromDisk(fileName);
   }

   @Override
   public void display() {
      System.out.println("Displaying " + fileName);
   }

   private void loadFromDisk(String fileName){
      System.out.println("Loading " + fileName);
   }
}
```

Proxy.java

```
public class ProxyImage implements Image{

   private RealImage realImage;
   private String fileName;

   public ProxyImage(String fileName){
      this.fileName = fileName;
   }

   @Override
   public void display() {
      if(realImage == null){
         realImage = new RealImage(fileName);
      }
      realImage.display();
   }
}
```

COMPOSITE PATTERN
==================
Employee.java
```java
import java.util.ArrayList;
import java.util.List;

public class Employee {
   private String name;
   private String dept;
   private int salary;
   private List<Employee> subordinates;

   // constructor
   public Employee(String name,String dept, int sal) {
      this.name = name;
      this.dept = dept;
      this.salary = sal;
      subordinates = new ArrayList<Employee>();
   }

   public void add(Employee e) {
      subordinates.add(e);
   }

   public void remove(Employee e) {
      subordinates.remove(e);
   }

   public List<Employee> getSubordinates(){
     return subordinates;
   }

   public String toString(){
      return ("Employee :[ Name : " + name + ", dept : " + dept + ", salary :" + salary+" ]");
   }
}
```

CompositePatternDemo.java
```java
public class CompositePatternDemo {
   public static void main(String[] args) {

      Employee CEO = new Employee("John","CEO", 30000);

      Employee headSales = new Employee("Robert","Head Sales", 20000);

      Employee headMarketing = new Employee("Michel","Head Marketing", 20000);

      Employee clerk1 = new Employee("Laura","Marketing", 10000);
      Employee clerk2 = new Employee("Bob","Marketing", 10000);
```

```
    Employee salesExecutive1 = new Employee("Richard","Sales", 10000);
    Employee salesExecutive2 = new Employee("Rob","Sales", 10000);

    CEO.add(headSales);
    CEO.add(headMarketing);

    headSales.add(salesExecutive1);
    headSales.add(salesExecutive2);

    headMarketing.add(clerk1);
    headMarketing.add(clerk2);

    //print all employees of the organization
    System.out.println(CEO);

    for (Employee headEmployee : CEO.getSubordinates()) {
      System.out.println(headEmployee);

      for (Employee employee : headEmployee.getSubordinates()) {
        System.out.println(employee);
      }
    }
  }
}
```

DECORATOR PATTERN
=================
Shape.java

```
public interface Shape {
  void draw();
}


public class Rectangle implements Shape {

  @Override
  public void draw() {
    System.out.println("Shape: Rectangle");
  }
}


public class Circle implements Shape {

  @Override
  public void draw() {
    System.out.println("Shape: Circle");
  }
}


public abstract class ShapeDecorator implements Shape {
  protected Shape decoratedShape;

  public ShapeDecorator(Shape decoratedShape){
    this.decoratedShape = decoratedShape;
  }

  public void draw(){
    decoratedShape.draw();
  }
}


public class RedShapeDecorator extends ShapeDecorator {

  public RedShapeDecorator(Shape decoratedShape) {
    super(decoratedShape);
  }

  @Override
  public void draw() {
    decoratedShape.draw();
    setRedBorder(decoratedShape);
  }

  private void setRedBorder(Shape decoratedShape){
    System.out.println("Border Color: Red");
  }
}
```

DecoratorPatternDemo.java

```
public class DecoratorPatternDemo {
  public static void main(String[] args) {

    Shape circle = new Circle();

    Shape redCircle = new RedShapeDecorator(new Circle());

    Shape redRectangle = new RedShapeDecorator(new Rectangle());
    System.out.println("Circle with normal border");
    circle.draw();

    System.out.println("\nCircle of red border");
    redCircle.draw();

    System.out.println("\nRectangle of red border");
    redRectangle.draw();
  }
}
```

State.java
```
public interface State {
  public void doAction(Context context);
}
```

StartState.java
```
public class StartState implements State {

  public void doAction(Context context) {
    System.out.println("Player is in start state");
    context.setState(this);
  }

  public String toString(){
    return "Start State";
  }
}
```

StopState.java
```
public class StopState implements State {

  public void doAction(Context context) {
    System.out.println("Player is in stop state");
    context.setState(this);
  }

  public String toString(){
    return "Stop State";
  }
}
```

Context.java
```
public class Context {
  private State state;

  public Context(){
    state = null;
  }

  public void setState(State state){
    this.state = state;
  }
  public State getState(){
    return state;
  }
}
```

StatePatternDemo.java

```
public class StatePatternDemo {
  public static void main(String[] args) {
    Context context = new Context();

    StartState startState = new StartState();
    startState.doAction(context);

    System.out.println(context.getState().toString());

    StopState stopState = new StopState();
    stopState.doAction(context);

    System.out.println(context.getState().toString());
  }
}
```