**Software Construction Lab Manual**

**Name Umer Hayat**

**Reg No# Fa16-bse-106**

**Lab 0**

**Problem 1:**

package lab0;

import java.util.Scanner;

import java.util.\*;

public class problem1 {

public void work(){

///////////////////////////////////////////////////////////////////

Scanner input=new Scanner(System.in);

///////////////////////////////////////////////////////////////////

System.out.print("how many numbers you want to add up: ");

int y=input.nextInt();

int [] x=new int [y];

///////////////////////////////////////////////////////////////////

for (int i = 0; i < x.length; i++) {

System.out.print("Enter number "+(i+1)+": ");

x[i]=input.nextInt();

}

///////////////////////////////////////////////////////////////////

int z=0;

for (int i = 0; i < x.length; i++) {

z=z+x[i];

}

///////////////////////////////////////////////////////////////////

System.out.println("The sum of numbers Entered: "+z);

///////////////////////////////////////////////////////////////////

}

}

**Problem 2**

public class problem2 {

public void work(){

//////////////////////////////////////////////////////////////////

int x=0;

int y=1;

int z=2;

int l=0;

//////////////////////////////////////////////////////////////////

Scanner input =new Scanner(System.in);

//////////////////////////////////////////////////////////////////

System.out.print("Enter Series Limit: ");

l=input.nextInt();

//////////////////////////////////////////////////////////////////

int [] f=new int [l];

//////////////////////////////////////////////////////////////////

for (int i = 0; i < f.length; i++) {

int check=x+y;

if(i==0||i==1){

f[i]=1;

x=f[i];y=f[i];

}else{

z=x+y;

f[i]=z;

x=f[i-1];

y=f[i];

}

}

//////////////////////////////////////////////////////////////////

System.out.print("Fibnocci series: ");

for (int i = 0; i < f.length; i++) {

System.out.print(f[i]+" ");

} }

}

**Problem 3**

public class problem3 {

public void work(){

Scanner input =new Scanner(System.in);

//////////////////////////////////////////////////////////////////

System.out.print("Enter the number to Check for palindrome: ");

int x=input.nextInt();

//////////////////////////////////////////////////////////////////

String str=Integer.toString(x);

//////////////////////////////////////////////////////////////////

StringBuilder rstr=new StringBuilder();

//////////////////////////////////////////////////////////////////

for (int i = str.length()-1;i>=0; i--) {

rstr.append(str.charAt(i));

}

String rs=rstr.toString();

//////////////////////////////////////////////////////////////////

if (str.equalsIgnoreCase(rs)){

System.out.println("the number "+str+" is palindrome.");

}else{

System.out.println("the number "+str+" is not palindrome.");

}}}

**Problem 4**

public class problem4 {

public void work(){

Scanner input =new Scanner(System.in);

//////////////////////////////////////////////////////////////////

System.out.print("Enter a number: ");

//////////////////////////////////////////////////////////////////

int x=input.nextInt();

//////////////////////////////////////////////////////////////////

System.out.print("Hailstone sequence: "+x+" ");

while(x!=1){

if(x%2==0){

x=x/2;

}else{

x=(3\*x)+1;

}

//////////////////////////////////////////////////////////////

System.out.print(x+" ");

//////////////////////////////////////////////////////////////

} }}

**Problem 5**

class problem5

public class problem5 {

public void work(){

//////////////////////////////////////////////////////////////////

int x;

int y;

//////////////////////////////////////////////////////////////////

char c;

//////////////////////////////////////////////////////////////////

Scanner input =new Scanner(System.in);

//////////////////////////////////////////////////////////////////

System.out.print("Enter operator symbol, Operation etc (+,-,/,\*): ");

c=input.next().charAt(0);

//////////////////////////////////////////////////////////////////

System.out.print("Enter ist nuber: ");

x=input.nextInt();

System.out.print("Enter 2nd nuber: ");

y=input.nextInt();

//////////////////////////////////////////////////////////////////

if(c=='+'){

System.out.println(x+" + "+y+" : "+(x+y));

}else if(c=='-'){

System.out.println(x+" - "+y+" : "+(x-y));

}else if(c=='\*'){

System.out.println(x+" \* "+y+" : "+(x\*y));

}else if(c=='/'){

System.out.println(x+" / "+y+" : "+(x/y));

}

//////////////////////////////////////////////////////////////////

}}

**Class: Test Program**

package lab0;

public class Program {

public static void main(String[] args) {

//////////////////////////////////////////////////////////////////

//problem1 p1=new problem1();p1.work();

//////////////////////////////////////////////////////////////////

//problem2 p2=new problem2();p2.work();

//////////////////////////////////////////////////////////////////

//problem3 p3=new problem3();p3.work();

//////////////////////////////////////////////////////////////////

//problem4 p4=new problem4();p4.work();

//////////////////////////////////////////////////////////////////

//problem5 p5=new problem5();p5.work();

//////////////////////////////////////////////////////////////////

}}

//uncomment the problem object you want to run

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 1**

**Problem:** Design a Saving Account class that stores a saving account balance…...... After the last iteration the program should display ending balance.

Solution:

**Class: SavingAccount**

package LAB1;

public class SavingAccount {

/////////////////////////////////////////////////////////////////

Private double accBalance=0.000;

private double annIntrestrate=0.00;

/////////////////////////////////////////////////////////////////

public SavingAccount(double starting, double annintrest){

accBalance=starting;

annIntrestrate=annintrest;

}

/////////////////////////////////////////////////////////////////

public double getAccBalance() {

return accBalance;

}

public double getAnnIntrestrate() {

return annIntrestrate;

}

public void setAccBalance(double accBalance) {

this.accBalance = accBalance;

}

public void setAnnIntrestrate(double annIntrestrate) {

this.annIntrestrate = annIntrestrate;

}

////////////////////////////////////////////////////////////////

public void Withdraw(double wamt){

setAccBalance((accBalance-wamt));

}

////////////////////////////////////////////////////////////////

public void Deposit(double Damt){

setAccBalance((accBalance+Damt));

}

////////////////////////////////////////////////////////////////

public double addmonthlyintrest(){

double monthlyintrestrate=(getAnnIntrestrate()/1200.00);

double monthlyintrest=(getAccBalance()\*monthlyintrestrate);

setAccBalance(accBalance+monthlyintrest);

return monthlyintrest;

}}

////////////////////////////////////////////////////////////////////////////////////////////////////

**Class: Test Program**

package LAB1;

import java.util.Scanner;

import java.util.\*;

public class TestProgram {

public static void main(String[] args) {

//////////////////////////////////////////////

Scanner input=new Scanner(System.in);

double annintrest=0.00;

double startbal=0.000;

double noofmonths=0;

//////////////////////////////////////////////

while(annintrest<=0){

System.out.print("Please Enter Annual Intrest Rate: ");

annintrest=input.nextDouble();

if(annintrest<=0){

System.out.print("Please Enter a valid Annual Intrest Rate (greater then zero): ");

annintrest=input.nextDouble();

}}

//////////////////////////////////////////////

System.out.print("Please Enter Starting Balance: ");

startbal=input.nextDouble();

//////////////////////////////////////////////

System.out.print("Please Enter no of months passed since oppening of account: ");

noofmonths=input.nextDouble();

//////////////////////////////////////////////

SavingAccount s1=new SavingAccount(startbal,annintrest);

//////////////////////////////////////////////

double amtwithdrawn=0;

double amtdeposited=0;

for (int i = 0; i < noofmonths; i++) {

System.out.print("Please Enter the amount withdrwan douring month "+(i+1)+" : ");

amtwithdrawn=input.nextDouble();

s1.Withdraw(amtwithdrawn);

//////////////////////////////////////////////

System.out.print("Please Enter the amount deposited douring month "+(i+1)+" : ");

amtdeposited=input.nextDouble();

s1.Deposit(amtdeposited);

//////////////////////////////////////////////

System.out.println("month "+(i+1)+"'s intrest is: "+s1.addmonthlyintrest());

System.out.println("Total Ammount after addition of intrest: "+s1.getAccBalance());

//////////////////////////////////////////////

} }}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 2**

**Problem :** Create a program that takes no of students , takes three grades for each of them and calculates their passing and failing status according to their graduate and undergraduate academics

**Class: Graduate**

**package LAB2a;**

public class Graduate extends Student{

Graduate(String name){

super(name);

for (int i = 0; i < test; i++) {

System.out.print("Enter Test "+(1+i)+" Scores: ");

TestResults[i]=input.nextInt();

}

System.out.println("");

}

@Override public String getCoursegrade(){

int total=0;

for (int i = 0; i < TestResults.length; i++) {

total+=getTestResults(i+1);

}

if(total>80){

return "Pass";

}else{

return "Fail";

} }}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Student**

package LAB2a;

import java.util.\*;

public abstract class Student {

/////////////////////////////////////////////////////////

protected final int test = 3;

protected String name;

protected String CourseGrade;

protected int [] TestResults = new int [test];

Scanner input=new Scanner(System.in);

/////////////////////////////////////////////////////////

public Student(){

}

/////////////////////////////////////////////////////////

public Student(String name){

this.name=name;

}

/////////////////////////////////////////////////////////

public int getTest() {

return test;

}

public String getName() {

return name;

}

public int getTestResults(int testnumber) {

return TestResults[testnumber-1];

}

/////////////////////////////////////////////////////////

public void setName(String name) {

this.name = name;

}

public void setCourseGrade(String CourseGrade) {

this.CourseGrade = CourseGrade;

}

/////////////////////////////////////////////////////////

public abstract String getCoursegrade();

/////////////////////////////////////////////////////////

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: UnderGraduate**

package LAB2a;

public class UnderGraduate extends Student {

public UnderGraduate(String name){

super(name);

for (int i = 0; i < test; i++) {

System.out.print("Enter Test "+(1+i)+" Scores: ");

TestResults[i]=input.nextInt();

}

System.out.println("");

}

@Override public String getCoursegrade(){

int total=0;

for (int i = 0; i < test; i++) {

total+=getTestResults(i+1);

}

if(total>70){

return "Pass";

}else{

return "Fail";

}}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Program**

package LAB2a;

import java.util.Scanner;

public class Program {

public static void main(String [] args){

//////////////////////////////////////////////////

Scanner input=new Scanner(System.in);

//////////////////////////////////////////////////

Student std[]=new Student[40];

String names;

String Status;

//////////////////////////////////////////////////

System.out.print("Enter number of Students:");

int noofstd=input.nextInt();

//////////////////////////////////////////////////

for (int i = 0; i < noofstd ; i++) {

System.out.print("Enter Student name: ");

names=input.next();

//////////////////////////////////////////////////

System.out.print("Enter Student's Accadamic Status (Graduate/Undergraduate): ");

Status=input.next();

if(Status.equalsIgnoreCase("Undergraduate")||Status.equalsIgnoreCase("U")){

std[i]=new UnderGraduate(names);

}else{

std[i]=new Graduate(names);

}

}

//////////////////////////////////////////////////

for (int i = 0; i < noofstd; i++) {

System.out.println(std[i].getName()+"'s Grade: "+std[i].getCoursegrade());

}

/////////////////////////////////////////////////

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Class: Circle**

package LAB2b;

public class Circle {

double radius;

String color;

public Circle(){

radius=1.0;

color="red";

}

public Circle(double radius){

color="red";

this.radius=radius;

}

public Circle(double radius, String color){

this.radius=radius;

this.color=color;

}

public double getRadius() {

return radius;

}

public String getColor() {

return color;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Program**

package LAB2b;

public class Program {

public static void main(String [] args){

Circle c1 = new Circle();

Circle c2 = new Circle(10.5);

Circle c3 = new Circle(10);

Circle c4 = new Circle(11.5,"yellow");

System.out.println("color: "+c1.getColor()+"\nRadius: "+c1.getRadius()+"\n");

System.out.println("color: "+c2.getColor()+"\nRadius: "+c2.getRadius()+"\n");

System.out.println("color: "+c3.getColor()+"\nRadius: "+c3.getRadius()+"\n");

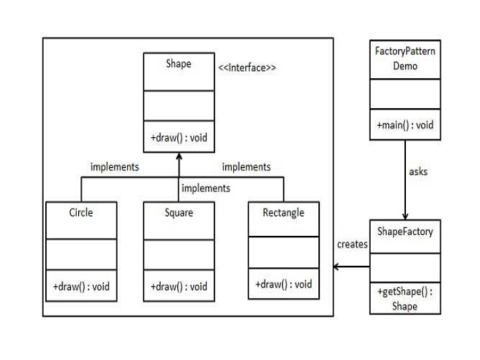
System.out.println("color: "+c4.getColor()+"\nRadius: "+c4.getRadius());

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 3**

****

**Class: Circle**

package Lab;

public class Circle implements Shape{

@Override public void draw(){

System.out.println("I am Circle O.");

System.out.println("-----------");

System.out.println("| |");

System.out.println("| |");

System.out.println("| |");

System.out.println("| |");

System.out.println("-----------");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Rectangle**

package Lab;

public class Rectangle implements Shape{

@Override public void draw(){

System.out.println("I am Rectange [ ].");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Square**

package Lab;

public class Square implements Shape{

@Override public void draw(){

System.out.println("I am Sqaure [ ].");

System.out.println("-----------");

System.out.println("| |");

System.out.println("| |");

System.out.println("| |");

System.out.println("| |");

System.out.println("-----------");

} }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Shape**

package Lab;

public interface Shape {

public void draw();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: ShapeFactory**

package Lab;

public class ShapeFactory {

public Shape getshape(String Shapetype){

if(Shapetype==null){

return null;

}

if(Shapetype.equalsIgnoreCase("Circle")){

return (new Circle());

}else if(Shapetype.equalsIgnoreCase("Square")){

return (new Square());

}else if(Shapetype.equalsIgnoreCase("Rectangle")){

return (new Rectangle());

}

return null;

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: FactoryPattern**

package Lab;

import java.util.\*;

public class FactoryPattern {

public static void main(String [] args){

Scanner input=new Scanner(System.in);

ShapeFactory ShapeGetter=new ShapeFactory();

System.out.print("Enter the Shape:");

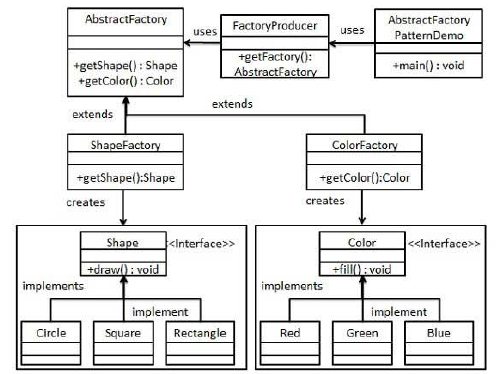
String shape=input.next();

Shape S=ShapeGetter.getshape(shape);

S.draw();

}}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**Class: AbstractFactory**

package Lab4\_2;

public abstract class AbstractFactory {

public abstract Shape getshape(String Shape);

public abstract Color getcolor(String Color);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Blue**

package Lab4\_2;

public class Blue implements Color{

@Override

public String fill(){

return("filled with blue.");

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Circle**

package Lab4\_2;

public class Circle implements Shape {

@Override

public String draw(){

return("I am Circle.");

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Color**

package Lab4\_2;

public interface Color {

public String fill();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: ColorFactory**

package Lab4\_2;

public class ColorFactory extends AbstractFactory{

@Override

public Color getcolor(String color){

if(color.equalsIgnoreCase("red")){

return new Red();

}else if(color.equalsIgnoreCase("Blue")){

return new Blue();

}else if(color.equalsIgnoreCase("green")){

return new Green();

}else{

return null;

}

}

@Override

public Shape getshape(String Shape){

return null;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: FactoryProducer**

package Lab4\_2;

public class FactoryProducer {

public static AbstractFactory getfactory(String choise){

if(choise.equalsIgnoreCase("color")){

return new ColorFactory();

}else if(choise.equalsIgnoreCase("shape")){

return new ShapeFactory();

}else{

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Green**

package Lab4\_2;

public class Green implements Color{

@Override

public String fill(){

return("filled with Green.");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Rectangle**

package Lab4\_2;

public class Rectangle implements Shape {

@Override

public String draw(){

return("I am Rectangle.");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Red.**

package Lab4\_2;

public class Red implements Color{

@Override

public String fill(){

return("filled with red.");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Shape**

package Lab4\_2;

public interface Shape {

public String draw();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: ShapeFactory**

package Lab4\_2;

public class ShapeFactory extends AbstractFactory{

@Override

public Shape getshape(String shape){

if(shape.equalsIgnoreCase("Circle")){

return new Circle();

}else if(shape.equalsIgnoreCase("Square")){

return new Square();

}else if(shape.equalsIgnoreCase("Rectangle")){

return new Rectangle();

}else {

return null;

}

}

@Override

public Color getcolor(String color){

return null;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Square**

package Lab4\_2;

public class Square implements Shape {

@Override

public String draw(){

return("I am Square.");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class: Program**

package Lab4\_2;

import java.util.\*;

import java.\*;

public class Program {

public static void main (String [] args){

Scanner input = new Scanner(System.in);

System.out.print("Enter the Name of the Shape (Circle/Rectangle/Square):");

String Shape=input.next();

System.out.print("Enter the Clour to fill the Shape (Red/Blue/Green):");

String Color=input.next();

AbstractFactory Shaper=FactoryProducer.getfactory("Shape");

Shape s=Shaper.getshape(Shape);

AbstractFactory Colorer=FactoryProducer.getfactory("Color");

Color c=Colorer.getcolor(Color);

System.out.println(s.draw().toString()+" "+c.fill().toString());

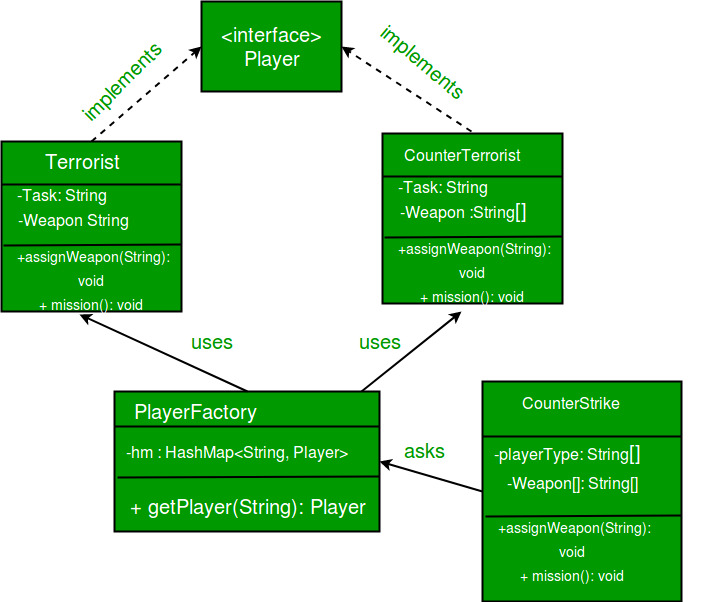
}

{

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 6**

# Flyweight Design Pattern

[](https://media.geeksforgeeks.org/wp-content/uploads/flyweight.jpg)

|  |
| --- |
| // A Java program to demonstrate working of  // FlyWeight Pattern with example of Counter  // Strike Game  **Interface : Player**  import java.util.Random;  import java.util.HashMap;    // A common interface for all players  interface Player  {      public void assignWeapon(String weapon);      public void mission();  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    **Class : Terrorist**  // Terrorist must have weapon and mission  class Terrorist implements Player  {      // Intrinsic Attribute      private final String TASK;        // Extrinsic Attribute      private String weapon;        public Terrorist()      {          TASK = "PLANT A BOMB";      }      public void assignWeapon(String weapon)      {          // Assign a weapon          this.weapon = weapon;      }      public void mission()      {          //Work on the Mission          System.out.println("Terrorist with weapon "                             + weapon + "|" + " Task is " + TASK);      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    **Class : CounterTerrorist**  // CounterTerrorist must have weapon and mission  class CounterTerrorist implements Player  {      // Intrinsic Attribute      private final String TASK;        // Extrinsic Attribute      private String weapon;        public CounterTerrorist()      {          TASK = "DIFFUSE BOMB";      }      public void assignWeapon(String weapon)      {          this.weapon = weapon;      }      public void mission()      {          System.out.println("Counter Terrorist with weapon "                             + weapon + "|" + " Task is " + TASK);      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    **Class : PlayerFactory**  // Claass used to get a playeer using HashMap (Returns  // an existing player if a player of given type exists.  // Else creates a new player and returns it.  class PlayerFactory  {      /\* HashMap stores the reference to the object         of Terrorist(TS) or CounterTerrorist(CT).  \*/      private static HashMap <String, Player> hm =                           new HashMap<String, Player>();        // Method to get a player      public static Player getPlayer(String type)      {          Player p = null;            /\* If an object for TS or CT has already been             created simply return its reference \*/          if (hm.containsKey(type))                  p = hm.get(type);          else          {              /\* create an object of TS/CT  \*/              switch(type)              {              case "Terrorist":                  System.out.println("Terrorist Created");                  p = new Terrorist();                  break;              case "CounterTerrorist":                  System.out.println("Counter Terrorist Created");                  p = new CounterTerrorist();                  break;              default :                  System.out.println("Unreachable code!");              }                // Once created insert it into the HashMap              hm.put(type, p);          }          return p;      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Class : CounterStrike**  // Driver class  public class CounterStrike  {      // All player types and weopons (used by getRandPlayerType()      // and getRandWeapon()      private static String[] playerType =                      {"Terrorist", "CounterTerrorist"};      private static String[] weapons =        {"AK-47", "Maverick", "Gut Knife", "Desert Eagle"};          // Driver code      public static void main(String args[])      {          /\* Assume that we have a total of 10 players             in the game. \*/          for (int i = 0; i < 10; i++)          {              /\* getPlayer() is called simply using the class                 name since the method is a static one \*/              Player p = PlayerFactory.getPlayer(getRandPlayerType());                /\* Assign a weapon chosen randomly uniformly                 from the weapon array  \*/              p.assignWeapon(getRandWeapon());                // Send this player on a mission              p.mission();          }      }        // Utility methods to get a random player type and      // weapon      public static String getRandPlayerType()      {          Random r = new Random();            // Will return an integer between [0,2)          int randInt = r.nextInt(playerType.length);            // return the player stored at index 'randInt'          return playerType[randInt];      }      public static String getRandWeapon()      {          Random r = new Random();            // Will return an integer between [0,5)          int randInt = r.nextInt(weapons.length);            // Return the weapon stored at index 'randInt'          return weapons[randInt];      }  } |

Output:

Counter Terrorist Created

Counter Terrorist with weapon Gut Knife| Task is DIFFUSE BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB

Terrorist Created

Terrorist with weapon AK-47| Task is PLANT A BOMB

Terrorist with weapon Gut Knife| Task is PLANT A BOMB

Terrorist with weapon Gut Knife| Task is PLANT A BOMB

Terrorist with weapon Desert Eagle| Task is PLANT A BOMB

Terrorist with weapon AK-47| Task is PLANT A BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB

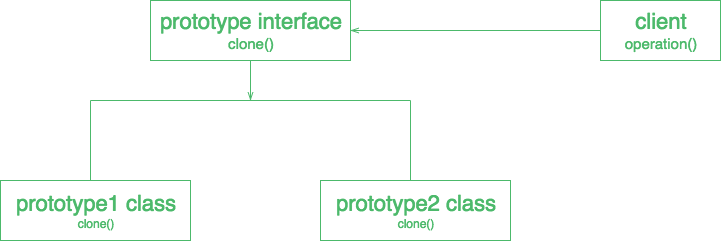
Counter Terrorist with weapon Gut Knife| Task is DIFFUSE BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Prototype Design Pattern

**The UML Diagram of the Prototype Design Pattern**

[](https://media.geeksforgeeks.org/wp-content/uploads/download-1.png)

|  |
| --- |
| // A Java program to demonstrate working of  // Prototype Design Pattern with example  // of a ColorStore class to store existing objects.    **Abstract class : Color**  import java.util.HashMap;  import java.util.Map;      abstract class Color implements Cloneable  {        protected String colorName;        abstract void addColor();        public Object clone()      {          Object clone = null;          try          {              clone = super.clone();          }          catch (CloneNotSupportedException e)          {              e.printStackTrace();          }          return clone;      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Class : blueColor**  class blueColor extends Color  {      public blueColor()      {          this.colorName = "blue";      }     @Override      void addColor()      {          System.out.println("Blue color added");      }    }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    **Class : blackColor**  class blackColor extends Color{     public blackColor()      {          this.colorName = "black";      }      @Override      void addColor()      {          System.out.println("Black color added");      }  }    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Class : ColorStore**  class ColorStore {        private static Map<String, Color> colorMap = new HashMap<String, Color>();        static      {          colorMap.put("blue", new blueColor());          colorMap.put("black", new blackColor());      }        public static Color getColor(String colorName)      {          return (Color) colorMap.get(colorName).clone();      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Class : prototype**  class Prototype  {      public static void main (String[] args)      {          ColorStore.getColor("blue").addColor();          ColorStore.getColor("black").addColor();          ColorStore.getColor("black").addColor();          ColorStore.getColor("blue").addColor();      } } |

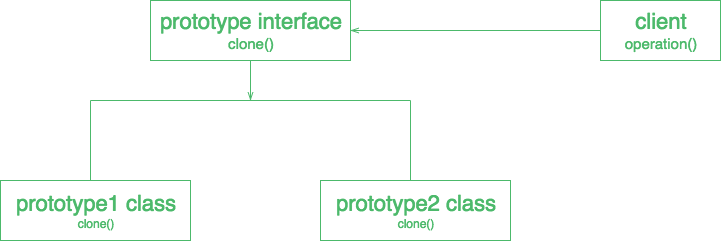
Output :

Blue color added

Black color added

Black color added

Blue color added

**UML diagram of example:**  
[](https://media.geeksforgeeks.org/wp-content/uploads/download-1.png)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 7**

Template Design Pattern

Suppose we want to provide an algorithm to build a house. The steps need to be performed to build a house are – building foundation, building pillars, building walls and windows. The important point is that the we can’t change the order of execution because we can’t build windows before building the foundation. So, in this case we can create a template method that will use different methods to build the house.



**Class: HouseTemplate**

public abstract class HouseTemplate {

public final void buildHouse(){

buildFoundation();

buildPillars();

buildWalls();

buildWindows();

System.out.println("House is built.");

}

//default implementation

private void buildWindows() {

System.out.println("Building Glass Windows");

}

//methods to be implemented by subclasses

public abstract void buildWalls();

public abstract void buildPillars();

private void buildFoundation() {

System.out.println("Building foundation with cement,iron rods and sand");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : WoodenHouse**

public class WoodenHouse extends HouseTemplate {

@Override

public void buildWalls() {

System.out.println("Building Wooden Walls");

}

@Override

public void buildPillars() {

System.out.println("Building Pillars with Wood coating");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : GlassHouse**

public class GlassHouse extends HouseTemplate {

@Override

public void buildWalls() {

System.out.println("Building Glass Walls");

}

@Override

public void buildPillars() {

System.out.println("Building Pillars with glass coating");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : HousingClient**

public class HousingClient {

public static void main(String[] args) {

HouseTemplate houseType = new WoodenHouse();

//using template method

houseType.buildHouse();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*");

houseType = new GlassHouse();

houseType.buildHouse();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Output of the above program is:**

Building foundation with cement,iron rods and sand

Building Pillars with Wood coating

Building Wooden Walls

Building Glass Windows

House is built.

\*\*\*\*\*\*\*\*\*\*\*\*

Building foundation with cement,iron rods and sand

Building Pillars with glass coating

Building Glass Walls

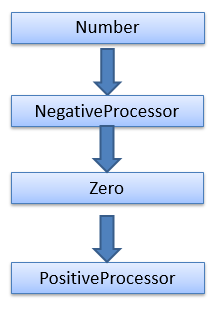
Building Glass Windows

House is built.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Chain of responsibility Design pattern

Set up a chain to process a request on a number object specified by the client. The number can be processed by negative processor, zero and positive processor. Specify handlers for each of the above types. Each of the handlers in the chain takes its turn at trying to handle the request it receives from the client.



**Interface : Chain**

Interface Chain {

    public abstract void setNext(Chain nextInChain);

    public abstract void process(Number request);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : Number**

class Number {

    private int number;

   public Number(int number)

    {

        this.number = number;

    }

   public int getNumber()

    {

        return number;

    }

  } \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : NegativeProcessor**

class NegativeProcessor implements Chain {

    private Chain nextInChain;

  public void setNext(Chain c)

    {

        nextInChain = c;

    }

    public void process(Number request)

    {

        if (request.getNumber() < 0)

        {

            System.out.println("NegativeProcessor : " + request.getNumber());

        }

        else

        {

            nextInChain.process(request);

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : ZeroProcessor**

class ZeroProcessor implements Chain

{

private Chain nextInChain;

  public void setNext(Chain c)

    {

        nextInChain = c;

    }

    public void process(Number request)

    {

        if (request.getNumber() == 0)

        {

            System.out.println("ZeroProcessor : " + request.getNumber());

        }

        else

        {

            nextInChain.process(request);

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : PositiveProcessor**

class PositiveProcessor implements Chain

{

private Chain nextInChain;

    public void setNext(Chain c)

    {

        nextInChain = c;

    }

   public void process(Number request)

    {

        if (request.getNumber() > 0)

        {

            System.out.println("PositiveProcessor : " + request.getNumber());

        }

        else

        {

            nextInChain.process(request);

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : TestChain**

class TestChain {

    public static void main(String[] args) {

        //configure Chain of Responsibility

        Chain c1 = new NegativeProcessor();

        Chain c2 = new ZeroProcessor();

        Chain c3 = new PositiveProcessor();

        c1.setNext(c2);

        c2.setNext(c3);

  //calling chain of responsibility

        c1.process(new Number(90));

        c1.process(new Number(-50));

        c1.process(new Number(0));

        c1.process(new Number(91));

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Output :**

PositiveProcessor : 90

NegativeProcessor : -50

ZeroProcessor : 0

PositiveProcessor : 91

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Command Design Pattern

Use a remote control as the example for command pattern. For this assignment use a light as a receiver, that we can switch on or off (concrete commands). Ultimately set up a client command that invokes the on and off commands using invoker class.

**Interface : Command**

public interface Command{

public void execute();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : LightOnCommand**

public class LightOnCommand implements Command{

//reference to the light

Light;

public LightOnCommand(Light light){

this.light = light;

}

@Override

public void execute(){

light.switchOn();

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : LightOffCommand**

public class LightOffCommand implements Command{

//reference to the light

Light light;

public LightOffCommand(Light light){

this.light = light;

}

@Override

public void execute(){

light.switchOff();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : Light**

public class Light{

private boolean on;

public void switchOn(){

on = true;

System.out.println("The Light is On");

}

public void switchOff(){

on = false;

System.out.println("The Light is Off");

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : RemoteControl**

public class RemoteControl{

private Command command;

public void setCommand(Command command){

this.command = command;

}

public void pressButton(){

command.execute();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class : Client**

public class Client{

public static void main(String[] args) {

RemoteControl control = new RemoteControl();

Light light = new Light();

Command lightOn = new LightOnCommand(light);

Command lightOff = new LightOffCommand(light);

//switch on

control.setCommand(lightOn);

control.pressButton();

//switch off

control.setCommand(lightOff);

control.pressButton();

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**