1. **ADAPTER PATTERN**

package adapterpattern;

public class AdapterPattern {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Movable bugattiVeyron = new BugattiVeyron();

MovableAdapter bugattiVeyronAdapter = new MovableAdapterImpl(bugattiVeyron);

double speedd =bugattiVeyronAdapter.getSpeed();

System.out.println(speedd);

}

package adapterpattern;

/\*\*

\*

\* @author hp

\*/

public class BugattiVeyron implements Movable {

@Override

public double getSpeed() {

return 268;

}

}

package adapterpattern;

/\*\*

\*

\* @author hp

\*/

public interface Movable {

// returns speed in MPH

double getSpeed();

}

package adapterpattern;

/\*\*

\*

\* @author hp

\*/

public interface MovableAdapter {

// returns speed in KM/H

double getSpeed();

}

/\*

package adapterpattern;

/\*\*

\*

\* @author hp

\*/

public class MovableAdapterImpl implements MovableAdapter

{

private Movable luxuryCars;

MovableAdapterImpl(Movable bugattiVeyron)

{

this.luxuryCars=bugattiVeyron;

}

// standard constructors

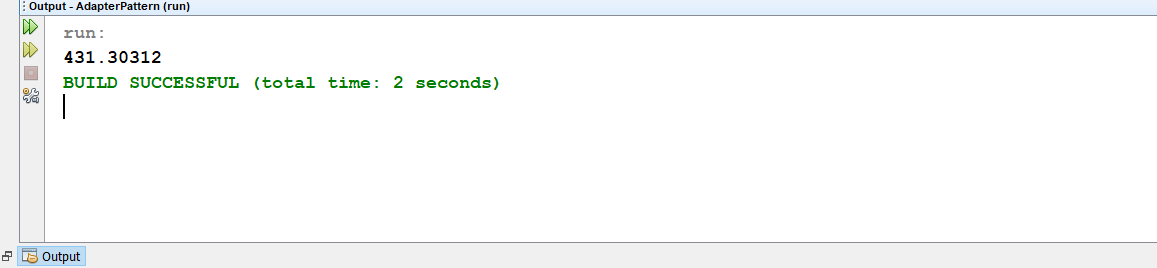
@Override

public double getSpeed() {

return convertMPHtoKMPH(luxuryCars.getSpeed());

}

private double convertMPHtoKMPH(double mph) {

return mph \* 1.60934; }

1. **BUILDER PATTERN**

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public class BuilderDemo {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

CDBuilder cdBuilder=new CDBuilder();

CDType cdType1=cdBuilder.buildSonyCD();

cdType1.showItems();

CDType cdType2=cdBuilder.buildSamsungCD();

cdType2.showItems();

}

}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public abstract class CD implements Packing {

@Override

public abstract String pack();

}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public class CDBuilder {

public CDType buildSonyCD(){

CDType cds=new CDType();

cds.addItem(new Sony());

return cds;

}

public CDType buildSamsungCD(){

CDType cds=new CDType();

cds.addItem(new Samsung());

return cds;

}

}

package builderdemo;

import java.util.\*;

/\*\*

\*

\* @author hp

\*/

public class CDType {

private List<Packing> items=new ArrayList<Packing>();

public void addItem(Packing packs) {

items.add(packs);

}

public void getCost(){

for (Packing packs : items) {

packs.price();

}

}

public void showItems(){

for (Packing packing : items){

System.out.print("CD name : "+packing.pack());

System.out.println(", Price : "+packing.price());

}}}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public abstract class Company extends CD {

@Override

public abstract int price();

}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public interface Packing {

public String pack();

public int price();

}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public class Samsung extends Company {

@Override

public int price(){

return 15;

}

@Override

public String pack(){

return "Samsung CD";

}

}

package builderdemo;

/\*\*

\*

\* @author hp

\*/

public class Sony extends Company{

@Override

public int price(){

return 20;

}

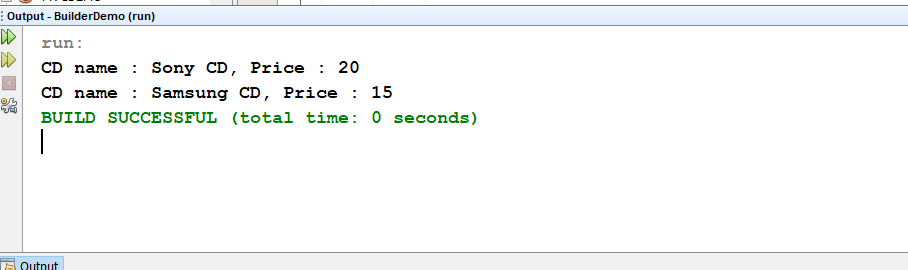
@Override

public String pack(){

return "Sony CD";

}

}//End of the Sony



1. **DECORATOR PATTERN**

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

public class Circle implements Shape {

@Override

public void draw() {

System.out.println("Shape: Circle");

}

}

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

public class DecoratorPattern {

/\*\*

\* @param args the command line arguments

\*/

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();

}

}

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Shape: Rectangle");

}

}

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

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");

}

}

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

public interface Shape {

void draw();

}

package decoratorpattern;

/\*\*

\*

\* @author hp

\*/

public abstract class ShapeDecorator implements Shape {

protected Shape decoratedShape;

public ShapeDecorator(Shape decoratedShape){

this.decoratedShape = decoratedShape;

}

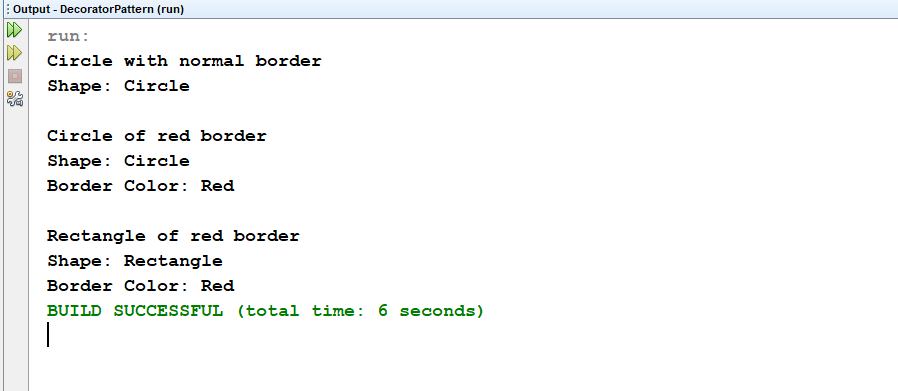
@Override

public void draw(){

decoratedShape.draw();

}

}



1. **FAÇADE PATTERN**

package facadepattern;

/\*\*

\*

\* @author hp

\*/

public class Blackberry implements MobileShop {

@Override

public void modelNo() {

System.out.println(" Blackberry Z10 ");

}

@Override

public void price() {

System.out.println(" Rs 55000.00 ");

}

}

package facadepattern;

import java.io.\*;

/\*\*

\*

\* @author hp

\*/

public class FacadePattern {

private static int choice;

public static void main(String args[])throws NumberFormatException,IOException {

do{

System.out.print("========= Mobile Shop ============ \n");

System.out.print(" 1. IPHONE. \n");

System.out.print(" 2. SAMSUNG. \n");

System.out.print(" 3. BLACKBERRY. \n");

System.out.print(" 4. Exit. \n");

System.out.print("Enter your choice: ");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

choice=Integer.parseInt(br.readLine());

ShopKeeper sk=new ShopKeeper();

switch (choice) {

case 1:

{

sk.iphoneSale();

}

break;

case 2:

{

sk.samsungSale();

}

break;

case 3:

{

sk.blackberrySale();

}

break;

default:

{

System.out.println(" You purchased Nothing.");

}

return;

}

}while(choice!=4);

}

}

package facadepattern;

/\*\*

\*

\* @author hp

\*/

public class Iphone implements MobileShop {

@Override

public void modelNo() {

System.out.println(" Iphone 6 ");

}

@Override

public void price() {

System.out.println(" Rs 65000.00 ");

}

}

package facadepattern;

/\*\*

\*

\* @author hp

\*/

public interface MobileShop

{

public void modelNo();

public void price();

}

package facadepattern;

/\*\*

\*

\* @author hp

\*/

public class Samsung implements MobileShop {

@Override

public void modelNo() {

System.out.println(" Samsung galaxy tab 3 ");

}

@Override

public void price() {

System.out.println(" Rs 45000.00 ");

}

}

package facadepattern;

/\*\*

\*

\* @author hp

\*/

public class ShopKeeper {

private MobileShop iphone;

private MobileShop samsung;

private MobileShop blackberry;

public ShopKeeper(){

iphone= new Iphone();

samsung=new Samsung();

blackberry=new Blackberry();

}

public void iphoneSale(){

iphone.modelNo();

iphone.price();

}

public void samsungSale(){

samsung.modelNo();

samsung.price();

}

public void blackberrySale(){

blackberry.modelNo();

blackberry.price();

}

}

