**Design Pattern**

1. **Creational Design Pattern**
   1. **Singleton**

//SingleObject Class

package SingleObject;

public class SingleObject {

private static SingleObject instance;

private SingleObject(){

}

public static SingleObject getInstance(){

if (instance == null){

instance = new SingleObject();

}

return instance;

}

public void showMessage(){

System.out.println("Single Instance");

}

}

//SingletonPatternDemo Class

public class SingletonPatternDemo {

public static void main(String args[]){

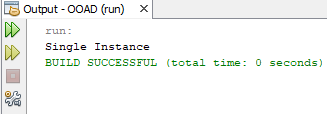
SingleObject a = SingleObject.getInstance();

a.showMessage();

}

}

Output



* 1. **Factory**

//Shape Interface

package ooad;

public interface Shape {

void draw();

}

//Circle Class

public class Circle implements Shape{

@Override

public void draw() {

System.out.println("This is Circle");

}

}

//Rectangle Class

public class Rectangle implements Shape{

@Override

public void draw() {

System.out.println("This is Rectangle");

}

}

//Square Class

public class Square implements Shape{

@Override

public void draw() {

System.out.println("This is Square");

}

}

//ShapeFactory Class

public class ShapeFactory {

public Shape getShape(String name){

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

return new Circle();

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

return new Rectangle();

}else{

return new Square();

}

}

}

//FactorDesignPatternDema Main method

public class FactorDesignPatternDema {

public static void main(String args[]){

ShapeFactory sf = new ShapeFactory();

Shape circle = sf.getShape("Circle");

circle.draw();

Shape rect = sf.getShape("Rectangle");

rect.draw();

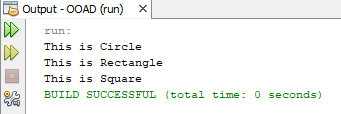
Shape Squ = sf.getShape("Square");

Squ.draw();

}

}

Output



1. **Structural Design Pattern**
   1. **Adapter**

//MediaPlayer interface

package Structural;

public interface MediaPlayer {

void playMusic(String audioType, String filename);

}

//AdvanceMediaPlayer

public interface AdvanceMediaPlayer {

void playVLCPlayer(String filename);

void playMp4Player(String filename);

}

//VLCPlayer Class

public class VLCPlayer implements AdvanceMediaPlayer{

@Override

public void playVLCPlayer(String filename) {

System.out.println("Playing VLC file:" +filename);

}

@Override

public void playMp4Player(String filename) {

}

}

//Mp4Player Class

public class Mp4Player implements AdvanceMediaPlayer{

@Override

public void playVLCPlayer(String filename) {

}

@Override

public void playMp4Player(String filename) {

System.out.println("Playing MP4 file: "+ filename);

}

}

//MediaAdopter Class

public class MediaAdapter implements MediaPlayer{

public static final String VLC = "vlc";

public static final String MP\_4 = "mp4";

private AdvanceMediaPlayer advancedMusicPlayer;

public MediaAdapter(String audioType){

if(audioType.equalsIgnoreCase(VLC)){

advancedMusicPlayer = new VLCPlayer();

}else if(audioType.equalsIgnoreCase(MP\_4)){

advancedMusicPlayer = new Mp4Player();

}

}

@Override

public void playMusic(String audioType, String filename) {

if (audioType.equalsIgnoreCase(VLC)) {

advancedMusicPlayer.playVLCPlayer("fileName");

} else if (audioType.equalsIgnoreCase(MP\_4)) {

advancedMusicPlayer.playMp4Player("fileName");

}

}

}

//AudioPlayer Class

public class AudioPlayer implements MediaPlayer{

private MediaAdapter mediaAdapter;

@Override

public void playMusic(String audioType, String fileName) {

if (audioType.equalsIgnoreCase("mp3")) {

System.out.println("Playing mp3 file: " + fileName);

}else if (audioType.equalsIgnoreCase("vlc") || audioType.equalsIgnoreCase("mp4")) {

mediaAdapter = new MediaAdapter(audioType);

mediaAdapter.playMusic(audioType, fileName);

} else {

System.out.println("The given format: " + audioType + " is not supported");

}

}

}

//AdopterPatternDemo Class

public class AdapterPatternDemo {

public static void main(String[] args) {

AudioPlayer audioPlayer = new AudioPlayer();

audioPlayer.playMusic("mp3", "song1.mp3");

audioPlayer.playMusic("mp4", "song2.mp4");

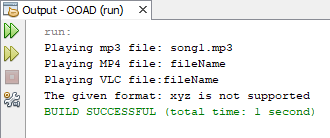
audioPlayer.playMusic("vlc", "song3.vlc");

audioPlayer.playMusic("xyz", "song4.avi");

}

}

Output



1. **Behavioral Design Pattern**
   1. **Observer**

//Observer Class

package Behavioral;

public abstract class Observer {

protected Subject subject;

public abstract void update();

}

//Subject Class

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();

}

}

}

//OctalObserver Class

public class OctalObserver extends Observer{

public OctalObserver(Subject subject) {

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println("Octal String: " + Integer.toOctalString(subject.getState()));

}

}

//HexaObserver Class

public class HexaObserver extends Observer{

public HexaObserver(Subject subject) {

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println("Hex String: " + Integer.toHexString(subject.getState()).toUpperCase());

}

}

//BinaryObserver Class

public class BinaryObserver extends Observer{

public BinaryObserver(Subject subject) {

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println("Binary String: " + Integer.toBinaryString(subject.getState()));

}

}

//ObserverPatternDemo Class

public class ObserverPatternDemo {

public static void main(String[] args) {

Subject subject = new Subject();

new HexaObserver(subject);

new OctalObserver(subject);

new BinaryObserver(subject);

System.out.println("First state change: 12");

subject.setState(12);

System.out.println("Second state change: 10");

subject.setState(10);

}

}

Output

