import java.util.Scanner;

public class Abstract{

public static void main(String arg[])

{

USPIM\_Factory nameusa = new USPIM\_Factory();

INPIM\_Factory nameind= new INPIM\_Factory();

System.out.println("Enter the American address and 11 digit Phonenumber");

nameusa.createAddr();

nameusa.createphno();

System.out.println("Enter the Indian address and 10 digit Phonenumber");

nameind.createAddr();

nameind.createphno();

}

}

abstract class Address

{

String Housename,city,street,state;

public abstract String getcountry();

public void inputdata()

{

Scanner in = new Scanner(System.in);

System.out.println("Enter the Housename");

Housename = in.nextLine();

System.out.println("Enter the city");

city = in.nextLine();

System.out.println("Enter the street");

street = in.nextLine();

System.out.println("Enter the state");

state = in.nextLine();

}

public void display()

{

System.out.println(Housename+"\n"+street+"\n"+city+"\n"+state);

}

}

class InAddress extends Address

{

String coun="IND";

InAddress()

{

super.inputdata();

super.display();

printAddr();

}

public String getcountry(){

return coun;

}

public void printAddr()

{

System.out.println(coun);

}

}

abstract class Phno{

long pnum;

int lem;

int n;

public abstract void getlen();

public void getnum()

{

Scanner in = new Scanner(System.in);

System.out.println("Enter Phonenum");

pnum = in.nextLong();

lem = (int)(Math.log10(pnum)+1);

}

public Phno(int len)

{

this.n=len;

}

public void displaynum()

{

if(n!=lem)

{

System.out.println("Invalid number");

}

else

{

System.out.println(pnum);

}

}

}

class Inphone extends Phno{

static Integer len=10;

public void getlen(){

len=10;

}

Inphone(){

super(len);

super.getnum();

super.displaynum();

}

}

class INPIM\_Factory implements PIM\_AbstractFactory

{

public Address createAddr(){

return new InAddress();

}

public Phno createphno(){

return new Inphone();

}

}

interface PIM\_AbstractFactory

{

public Address createAddr();

public Phno createphno();

}

class UsAddress extends Address

{

String coun="USA";

UsAddress()

{

super.inputdata();

super.display();

printAddr();

}

public String getcountry()

{

return coun;

}

public void printAddr()

{

System.out.println(coun);

}

}

class Usphone extends Phno

{

static Integer len=11;

public void getlen()

{

len=11;

}

Usphone()

{

super(len);

super.getnum();

super.displaynum();

}

}

class USPIM\_Factory implements PIM\_AbstractFactory{

public Address createAddr(){

return new UsAddress();

}

public Phno createphno(){

return new Usphone();

}

}

public class Decorator {

public static void main(String[] args)

{

Contact c1= new Music(new MaleContact());

c1.getinterest();

Contact c2= new Dance(new MaleContact());

c2.getinterest();

Contact c3= new Dance(new femaleContact());

c3.getinterest();

}

}

abstract class Contact {

public abstract void getinterest();

}

abstract class Decorater extends Contact {

public Contact decorateContact;

public Decorater(Contact decorateContact)

{

this.decorateContact=decorateContact;

}

abstract public void getinterest();

}

class femaleContact extends Contact {

public void getinterest()

{

System.out.println("I'am a Female");

}

}

class MaleContact extends Contact {

public void getinterest(){

System.out.println("I am a male");

}

}

class Music extends Decorater {

public Music(Contact decorateContact) {

super(decorateContact);

}

public void getinterest(){

decorateContact.getinterest();

setTalent(decorateContact);

}

private void setTalent(Contact decorateContact) {

System.out.println("I can sing");

}

}

class Dance extends Decorater {

public Dance(Contact decorateContact) {

super(decorateContact);

}

public void getinterest(){

decorateContact.getinterest();

setTalent(decorateContact);

}

private void setTalent(Contact decorateContact) {

System.out.println("I can Dance");

}

}

public class ChocolateController {

public static void main(String args[]) {

ChocolateBoiler boiler = ChocolateBoiler.getInstance();

boiler.fill();

boiler.boil();

boiler.drain();

// will return the existing instance

ChocolateBoiler boiler2 = ChocolateBoiler.getInstance();

}

}

class ChocolateBoiler {

private boolean empty;

private boolean boiled;

private static ChocolateBoiler uniqueInstance;

private ChocolateBoiler() {

empty = true;

boiled = false;

}

public static ChocolateBoiler getInstance() {

if (uniqueInstance == null) {

System.out.println("Creating unique instance of Chocolate Boiler");

uniqueInstance = new ChocolateBoiler();

}

System.out.println("Returning instance of Chocolate Boiler");

return uniqueInstance;

}

public void fill() {

if (isEmpty()) {

empty = false;

boiled = false;

// fill the boiler with a milk/chocolate mixture

}

}

public void drain() {

if (!isEmpty() && isBoiled()) {

// drain the boiled milk and chocolate

empty = true;

}

}

public void boil() {

if (!isEmpty() && !isBoiled()) {

// bring the contents to a boil

boiled = true;

}

}

public boolean isEmpty() {

return empty;

}

public boolean isBoiled() {

return boiled;

}

}

import java.util.\*;

interface customize{

void change();

}

class fontSize implements customize{

public void change(){

int s;

System.out.println("Enter the new font size");

Scanner in = new Scanner(System.in);

s = in.nextInt();

System.out.println("Font size changed to "+s);

}

}

class fontType implements customize{

public void change(){

String t;

System.out.println("Enter the new font type");

Scanner c = new Scanner(System.in);

t = c.next();

System.out.println("Font type changed to "+t);

}

}

class color implements customize{

public void change(){

String color;

System.out.println("Enter the new color");

Scanner m = new Scanner(System.in);

color = m.nextLine();

System.out.println("Color changed to "+color);

}

}

class customizeApp{

private customize size;

private customize type;

private customize col;

public customizeApp(){

size = new fontSize();

type = new fontType();

col = new color();

}

public void changeSize(){

size.change();

}

public void changeType(){

type.change();

}

public void changeColor(){

col.change();

}

}

public class FacadePattern {

public static void main(String[] args){

customizeApp custapp = new customizeApp();

custapp.changeSize();

custapp.changeType();

custapp.changeColor();

}

}

public class TestAdapter{

public static void main(String[] args) {

Shape rect = new Rectangle();

Shape cir = new Circle();

rect.fill();

rect.draw();

rect.display();

cir.display();

}

}

abstract class Shape {

abstract void fill();

abstract void draw();

abstract void display();

}

class Rectangle extends Shape {

XXRectangle myRectangle = new XXRectangle();

void fill(){

myRectangle.fillIt();

}

void draw(){

myRectangle.drawIt();

}

void display(){

myRectangle.displayIt();

}

}

class Circle extends Shape{

void fill(){

System.out.println("I am getting a color");

}

void draw(){

System.out.println("This is how im creted");

}

void display(){

System.out.println("This is how i look");

}

}

class XXRectangle{

public void fillIt(){

System.out.println("I am getting a color");

}

public void drawIt(){

System.out.println("This is how im creted");

}

public void displayIt(){

System.out.println("This is how i look");

}

}

class DP1{

public static void draw\_a\_line(double x1, double y1, double x2, double y2){

System.out.println("Line from ("+x1+","+y1+") to ("+x2+","+y2+")");

}

}

class DP2{

public static void drawline(double x1, double y1, double x2, double y2){

System.out.println("Line from ("+x1+","+y1+") to ("+x2+","+y2+")");

}

}

class DP1{

public static void draw\_a\_line(double x1, double y1, double x2, double y2){

System.out.println("Line from ("+x1+", "+y1+" to ("+x2+", "+y2+")");

}

public static void draw\_a\_circle(double x1,double y1,double r){

System.out.println("circle at ("+x1+","+y1+") of radius ("+r+")");

}

}

class DP2{

public static void drawline(double x1, double y1, double x2, double y2){

System.out.println("Line from ("+x1+", "+y1+" to ("+x2+", "+y2+")");

}

public static void drawcircle(double x1,double y1,double r){

System.out.println("circle at ("+x1+","+y1+") of radius ("+r+")");

}

}

abstract class Drawing{

abstract void getDescription();

abstract void drawLine(double x1, double y1, double x2, double y2);

abstract void drawCircle(double x1, double y1, double r);

}

class V1Drawing extends Drawing{

void getDescription(){

System.out.println(" Im using Dp1");

}

void drawLine(double x1, double y1, double x2, double y2){

DP1.draw\_a\_line(x1, y1, x2, y2);

}

void drawCircle(double x1, double y1, double r){

DP1.draw\_a\_circle(x1,y1,r);

}

}

class V2Drawing extends Drawing{

void getDescription(){

System.out.println(" Im using Dp2");

}

void drawLine(double x1, double y1, double x2, double y2){

DP2.drawline(x1, y1, x2, y2);

}

void drawCircle(double x1, double y1, double r){

DP2.drawcircle(x1,y1,r);

}

}

abstract class Shape{

Drawing drawing;

public Shape(Drawing drawing){

this.drawing = drawing;

}

abstract void draw();

}

class Rectangle extends Shape{

double \_x1, \_y1, \_x2, \_y2;

Rectangle(Drawing dp, double x1, double y1, double x2, double y2){

super(dp);

\_x1 = x1;

\_y1 = y1;

\_x2 = x2;

\_y2 = y2;

}

public void draw(){

drawing.getDescription();

drawing.drawLine(\_x1, \_y1, \_x2, \_y1);

drawing.drawLine(\_x2, \_y1, \_x2, \_y2);

drawing.drawLine(\_x2, \_y2, \_x1, \_y2);

drawing.drawLine(\_x1, \_y2, \_x1, \_y1);

}

}

public class TestBridge{

public static void main(String[] args) {

Drawing dp1 = new V1Drawing();

Drawing dp2 = new V2Drawing();

Shape rect1 = new Rectangle(dp1, 10, 200, 20, 100);

rect1.draw();

Shape rect2 = new Rectangle(dp2, 10, 200, 20, 100);

rect2.draw();

}

}

import java.util.ArrayList;

class Appointment {

String location,date,time;

public void setAppointment(String l,String d,String t){

this.location = l;

this.date = d;

this.time = t;

System.out.println("Appointment set: "+ l + "" + d + "" + t + "\n" );

}

}

interface Subject {

public void attach(Observer ob);

public void notifyObserver();

}

class ClubMeeting implements Subject {

ArrayList<Observer> CMAL;

Appointment ap;

public ClubMeeting(String l,String d,String t) {

ap = new Appointment();

ap.setAppointment(l, d, t);

CMAL = new ArrayList<Observer>();

}

@Override

public void attach(Observer ob) {

CMAL.add(ob);

}

@Override

public void notifyObserver() {

for(int i=0; i<CMAL.size(); i++){

Observer CMObserver = (Observer)CMAL.get(i);

CMObserver.update(ap);

}

}

public void setTheAppointment(String l,String d,String t){

this.ap.setAppointment(l, d, t);

notifyObserver();

}

}

interface Observer {

public void update(Appointment a);

}

class ClubMember implements Observer {

Subject sub;

Appointment app;

public ClubMember(Subject s){

this.sub = s;

sub.attach(this);

}

@Override

public void update(Appointment a) {

app = a;

}

public void display(){

System.out.println("Called by : " + this.toString());

System.out.println("Appointment details: ");

System.out.println("Location : " + this.app.location );

System.out.println("Date : " + this.app.date );

System.out.println("Time : " + this.app.time + "\n");

}

}

class ObserverTest {

public static void main(String[] args) {

ClubMeeting CM = new ClubMeeting("LHC", "5th Nov", "1:30");

ClubMember cMem = new ClubMember(CM);

ClubMember cMem2 = new ClubMember(CM);

CM.notifyObserver();

cMem.display();

cMem2.display();

CM.setTheAppointment("ESB", "6th Nov", "2:30");

cMem.display();

cMem2.display();

CM.setTheAppointment("DES", "8th Nov", "3:30");

cMem.display();

cMem2.display();

}

}

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Arrays;

public class Strategy{

public static void main(String[] arg)

{

Contact a = new ContactList1();

a.add1("Ram", "Piper");

a.add1("John","Company");

a.add1("Suresh","Company2");

a.add1("Abhi","Company2");

a.displayContact();

//Contact b =new ContactList2();

//b.displayContact();

}

}

abstract class Contact{

NameList name;

public Contact(NameList nam)

{

this.name=nam;

}

int i=0;

HashMap<Integer, ArrayList<String>> hmap = new HashMap<Integer, ArrayList<String>>();

ArrayList<String> detail=new ArrayList<String>();

public abstract void displayContact();

public void add1(String s1,String s2)

{

java.util.List<String> a = new ArrayList<String>();

a.add(s1.toString());

a.add(s2.toString());

detail.addAll(a);

hmap.put(i,(ArrayList<String>)a);

name.listOfDetails(hmap);

i=i+1;

}

}

class ContactList1 extends Contact{

static NameList nam = new SummerizeName();

public ContactList1()

{

super(nam);

}

public void displayContact()

{

name.Summerize();

}

}

class ContactList2 extends Contact{

static NameList nam = new SummerizeOrganization();

public ContactList2() {

super(nam);

}

public void displayContact()

{

name.Summerize();

}

}

interface NameList {

public void Summerize();

public void listOfDetails(HashMap<Integer , ArrayList<String>> hmap);

}

class SummerizeName implements NameList {

HashMap<Integer, ArrayList<String>> map = new HashMap<Integer, ArrayList<String>>();

ArrayList<String> key = new ArrayList<String>();

String[] val;

public void listOfDetails(HashMap<Integer, ArrayList<String>> hmap)

{

map=hmap;

val = new String[map.size()];

int j;

for (int i=0;i<map.size();i++)

{

j=0;

key = map.get(i);

val[i]=(key.get(j).toString());

}

}

public void Summerize()

{

Arrays.sort(val);

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

{

System.out.println(val[i]);

}

}

}

class SummerizeOrganization implements NameList {

//No need to implement

HashMap<Integer,ArrayList<String>> map = new HashMap<Integer, ArrayList<String>>();

public void listOfDetails(HashMap<Integer, ArrayList<String>> hmap)

{

}

public void Summerize()

{

}

}

import java.util.\*;

public class Factory

{

public static void main(String[] args)

{

Appointment myAppt;

AppointmentCreator myApptCreator = new DocApptCreator();

myAppt = myApptCreator.createAppt("EYE");

Scanner in = new Scanner(System.in);

String date, time, venue, person\_name;

System.out.println("Enter your appointment details:\n");

System.out.println("Enter the date:\n");

date = in.nextLine();

System.out.println("Enter the time:\n");

time = in.nextLine();

System.out.println("Enter the venue:\n");

venue = in.nextLine();

System.out.println("Enter the name of the person you are meeting:\n");

person\_name = in.nextLine();

myAppt.setAppt(date,time,venue,person\_name);

myAppt.displayApptDetails();

System.out.println("Enter the changed timing of Appointment:\n");

time = in.nextLine();

myAppt.setAppt(date,time,venue,person\_name);

System.out.println("Enter the another person's name you are meeting:\n");

person\_name = in.nextLine();

myAppt.setAppt(date,time,venue,person\_name);

System.out.println("Your Appointment details are:\n");

myAppt.displayApptDetails();

}

}

class Appointment

{

String appDate, appTime, appVenue, appName;

public void setAppt(String d, String t, String v, String n)

{

appDate = d;

appTime = t;

appVenue = v;

appName = n;

}

public void displayApptDetails()

{

System.out.println("Date:" + appDate);

System.out.println("Time:" + appTime);

System.out.println("Venue:" + appVenue);

System.out.println("Person to meet:" + appName+"\n");

}

}

class docEyeAppt extends Appointment

{

public void displayApptDetails()

{

System.out.println("Your Opthamologist Appointment details are:\n");

super.displayApptDetails();

}

}

class docEntAppt extends Appointment

{

public void displayApptDetails()

{

System.out.println("Your ENT Appointment details are:\n");

super.displayApptDetails();

}

}

class docVetAppt extends Appointment

{

public void displayApptDetails()

{

System.out.println("Your VET Appointment details are:\n");

super.displayApptDetails();

}

}

abstract class AppointmentCreator

{

public abstract Appointment createAppt(String x);

}

class DocApptCreator extends AppointmentCreator

{

public Appointment createAppt(String type)

{

if(type.equals("EYE"))

{

return new docEyeAppt();

}

else if(type.equals("ENT"))

{

return new docEntAppt();

}

else if(type.equals("VET"))

{

return new docVetAppt();

}

else

{

return null;

}

}

}