

```

import java.util.*;
//import java.lang.*;
import java.io.*;
class Page implements Cloneable {
    private int pageNum;

    // Helping function
    private void trace(String s) {
        System.out.println(s);
    }
    // Manager function
    public Page(int pageNum1) {
        pageNum=pageNum1;
    }
    // Access function
    // get
    public int getPageNum() { return pageNum; }

    // set
    public void setPageNum(int pageNum1) { pageNum = pageNum1; }
    public void setPage(Page page1) { pageNum = page1.getPageNum(); }

    // predicate
    // Implementor
    public void addOneMorePage(int pageNum1) { pageNum++; }
    public void display() {
        System.out.println(pageNum);
    }

    public String toString() {
        return ("pageNum: " + pageNum + "\n");
    }
    public boolean equals(Page obj) {
        if (!(obj instanceof Page)) return false;

        Page tstA;
        tstA = (Page) obj;

        return (pageNum == tstA.pageNum);
    }
    public Object clone() {
        try
        {
            return super.clone();

```

```

    }
    catch (CloneNotSupportedException e)
    {
        // This shouldn't happen, since we are Cloneable
        return null;
    }
}

```

```

class FrontCover implements Cloneable {
    private String name;
    private int numOfDishes = 10;

    // Helping function
    private void trace(String s) {
        System.out.println(s);
    }
    // Manager function
    public FrontCover(String na, int n) {
        name = na;
        numOfDishes = n;
    }
    // Access function
    // get
    public int getNumOfDishes() { return numOfDishes; }
    public String getName(){return name;}
    // set
    public void setNumOfDishes(int numOfDishes1) { numOfDishes =
numOfDishes1; }
    public void setNumOfDishes(FrontCover fc) { numOfDishes =
fc.getNumOfDishes(); }
    public void setName(String na){name = na;}
    public void setName(FrontCover na){name = na.getName();}
    // predicate
    // Implementor
    public void addOneMoreDishes(int numOfDishes1) { numOfDishes++; }
    public void display() {
        System.out.println(name);
        System.out.println(numOfDishes);
    }

    public String toString() {
        return ("name: " + name + "\n\t" + "numOfDishes: " + numOfDishes + "\n");
    }
}

```

```

}
public boolean equals(FrontCover obj) {
    if (!(obj instanceof FrontCover)) return false;

    FrontCover tstA;
    tstA = (FrontCover) obj;

    return (numOfDishes == obj.numOfDishes && name == obj.name);
}
public Object clone() {
    try
    {
        return super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // This shouldn't happen, since we are Cloneable
        return null;
    }
}
}

```

```

class BackCover implements Cloneable {
    private String phoneNum;

    // Helping function
    private void trace(String s) {
        System.out.println(s);
    }

    // Manager function
    public BackCover(String na) {
        phoneNum = na;
    }

    // Access function
    // get
    public String getPhoneNum(){return phoneNum;}
    // set
    public void setPhoneNum(String na){phoneNum = na;}
    public void setPhoneNum(BackCover na){phoneNum = na.getPhoneNum();}
    // predicate

    // Implementor

```

```

public void display() {
    System.out.println(phoneNum);
}
public String toString() {
    return ("phoneNum: " + phoneNum + "\n" );
}
public boolean equals(BackCover obj) {
    if (!(obj instanceof BackCover)) return false;

    BackCover tstA;
    tstA = (BackCover) obj;
    return (phoneNum == obj.phoneNum);
}
public Object clone() {
    try
    {
        return super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // This shouldn't happen, since we are Cloneable
        return null;
    }
}
}

```

```

class Menu implements Cloneable {
    private String title;
    private FrontCover fc;
    private BackCover bc;
    private Vector pages;

    // Helping function
    private void trace(String s) {
        System.out.println(s);
    }

    // Manager function
    public Menu(Vector pages1, FrontCover fc1, BackCover bc1, String title1) {
        pages=(Vector)pages1.clone();
        fc = (FrontCover)fc1.clone();
        bc = (BackCover)bc1.clone();
        title = title1;
    }
}

```

```

// Access function
// get
public Page getPage(int i) {
    return ((Page)pages.elementAt(i)); }
public String getTitle(){
    return title;
}
public String getFrontCover(){
    return fc.toString();
}
public String getBackCover(){
    return bc.toString();
}
public int getPageNum(int i) { return ((Page)pages.elementAt(i)).getPageNum(); }
public Vector getPages() { return pages; }
public int getNumOfPages() { return pages.size(); }
// set
public void setPage(int i, int pageNum) {pages.setElementAt(new
Page(pageNum), i); }
public void setPages(Vector pages1) {
    pages=(Vector)pages1.clone();
    for (int i=0; i < pages1.size(); i++) {
        pages.setElementAt(
            ((Page)pages1.elementAt(i)).clone(), i);
    }
}
// predicate
// Implementor
public void noPages() { pages = null; }
public void addPage(int pageNum) { pages.addElement(new Page(pageNum)); }
// Add at the end
public void insertPage(int pageNum, int index) {
    pages.insertElementAt(new Page(pageNum), index); }
// Add at the front: insertPage(9, 0);
public Page firstPage() { return ((Page)pages.firstElement()); }
public Page lastPage() { return ((Page)pages.lastElement()); }
public void dispPages() {
    for (int i=0; i < getNumOfPages(); i++)
        getPage(i).display();
}

public String toString() {
    StringBuffer s = new StringBuffer();
    s.append("Title: " + getTitle() + "\n");
}

```

```

s.append("Front Cover: \n\t" + getFrontCover() + "");
s.append("Back Cover: \n\t" + getBackCover() + "");
s.append("Total number of pages: " + getNumOfPages() + "\n");
for (int i=0; i < getNumOfPages(); i++){
s.append("\t");
s.append(getPage(i).toString());
}
return s.toString();
// return pages.toString();
}

```

```

public boolean equals(Menu obj) {
    if (!(obj instanceof Menu)) return false;
    Menu tstA;
    tstA = (Menu) obj;
    for (int i=0; i < getNumOfPages(); i++)
        if (!getPage(i).equals(tstA.getPage(i)))
            return false;
    return (title.equals(tstA.title) &&
        fc.equals(tstA.fc) &&
        bc.equals(tstA.bc));
}

```

```

public Object clone() {
    try
    {
        Menu e = (Menu)super.clone();
        e.pages = (Vector)pages.clone();
        for (int i=0; i < pages.size(); i++) {
            e.pages.setElementAt(
                ((Page)pages.elementAt(i)).clone(), i);
        }
        return e;
    }
    catch (CloneNotSupportedException e)
    {
        // This shouldn't happen, since we are Cloneable
        return null;
    }
}

```

```

class FancyMenu extends Menu {
    private String color;

```

```

// Helping function
private void trace(String s) {
    System.out.println(s);
}

public FancyMenu(String color1, Menu p,FrontCover fc, BackCover bc, String
title1) {
    super(p.getPages(),fc, bc, title1);
    color=color1;
    trace("");
}

// Access function
// get
public String getColor() { return color; }

// set
public void setColor(String color1) { color = color1; }
// predicate
// Implementor
public void addColor(String color1) { color += color1; }
public String toString() {
    // StringBuffer s = new StringBuffer();
    // for (int i=0; i < getNumOfPages(); i++)
    // s.append(getPage(i).toString());
    // return s.toString();
    return super.toString() + "Color : " + color;
}

public boolean equals(FancyMenu obj) {
    if (!(obj instanceof FancyMenu)) return false;

    FancyMenu tstE;
    tstE = (FancyMenu) obj;

    return (color == tstE.color && super.equals((Menu)obj));
}

}

public class Test {
    public static void main (String argv[])
    {
        System.out.println("===== Menu =====");
    }
}

```

```

Vector pages1 = new Vector();
pages1.addElement(new Page(1));
pages1.addElement(new Page(2));
pages1.addElement(new Page(3));
FrontCover fc = new FrontCover("Forbidden City", 123);
BackCover bc = new BackCover("408-123-4356");
Menu chinese = new Menu(pages1, fc, bc, "Chinese Food");

//test
Menu chinese2 = new Menu(pages1, fc, bc, "Chinese Food");
Menu chinese3 = (Menu)chinese.clone();
System.out.println(chinese + "\n");
//-----
Vector pages2 = new Vector();
pages2.addElement(new Page(1));
pages2.addElement(new Page(2));
pages2.addElement(new Page(3));
pages2.addElement(new Page(4));
FrontCover fc2 = new FrontCover("Taco Bell", 46);
BackCover bc2 = new BackCover("510-123-4567");
Menu mexicon = new Menu(pages2, fc2, bc2, "Mexicon Food");

System.out.println(mexicon + "\n");
//-----
System.out.println("\n===== Fancy Menu =====");

FancyMenu chinese_food = new FancyMenu ("Purple",chinese,fc,
bc,"Chinese Food");
System.out.println (chinese_food + "\n");

FancyMenu mexicon_food = new FancyMenu ("red", mexicon,fc2, bc2,
"Mexicon Food");
System.out.println (mexicon_food + "\n");
System.out.println ("Compare the chinese menu and mexican menu to see does is equal :
"+chinese.equals(mexicon));
System.out.println ("Compare the chinese menu and chinese2 menu to see does is equal :
"+chinese.equals(chinese2));
System.out.println ("Compare the chinese menu and a clone chinese menu chinese3 to see
does is equal : "+chinese.equals(chinese3));
}
}

```



```

336 bc,"Chinese Food");
337 System.out.println (chinese_food + "\n");
338
339 FancyMenu mexicon_food = new FancyMenu ("red", mexicon,fc2, bc2,
340 "Mexicon Food");
341 System.out.println (mexicon_food + "\n");
342 System.out.println ("Compare the chinese menu and mexican menu to see does is equal : "+chinese.equals(mexicon_food));
343 System.out.println ("Compare the chinese menu and chinese2 menu to see does is equal : "+chinese.equals(chinese2));
344 System.out.println ("Compare the chinese menu and a clone chinese menu chinese3 to see does is equal : "+chinese.equals(chinese3));
345 }
346 }

```

Execute Mode, Version, Inputs & Arguments

JDK 11.0.4



Interactive

Stdin Inputs

CommandLine Arguments

Execute

Result

CPU Time: 0.40 sec(s). Memory: 38796 kilobyte(s)

compiled and executed in 0.928 sec(s)

```

===== Menu =====
Title: Chinese Food
Front Cover:
  name: Forbidden City
  numOfDishes: 123
Back Cover:
  phoneNum: 408-123-4356
Total number of pages: 3
  pageNum: 1
  pageNum: 2
  pageNum: 3

Title: Mexicon Food
Front Cover:
  name: Taco Bell
  numOfDishes: 46
Back Cover:
  phoneNum: 510-123-4567
Total number of pages: 4
  pageNum: 1
  pageNum: 2
  pageNum: 3
  pageNum: 4

```



```
Title: Chinese Food
Front Cover:
  name: Forbidden City
  numOfDishes: 123
Back Cover:
  phoneNum: 408-123-4356
Total number of pages: 3
  pageNum: 1
  pageNum: 2
  pageNum: 3
```

```
Title: Mexican Food
Front Cover:
  name: Taco Bell
  numOfDishes: 46
Back Cover:
  phoneNum: 510-123-4567
Total number of pages: 4
  pageNum: 1
  pageNum: 2
  pageNum: 3
  pageNum: 4
```

===== Fancy Menu =====

```
Title: Chinese Food
Front Cover:
  name: Forbidden City
  numOfDishes: 123
Back Cover:
  phoneNum: 408-123-4356
Total number of pages: 3
  pageNum: 1
  pageNum: 2
  pageNum: 3
Color : Purple
```

```
Title: Mexican Food
Front Cover:
  name: Taco Bell
  numOfDishes: 46
Back Cover:
  phoneNum: 510-123-4567
Total number of pages: 4
  pageNum: 1
  pageNum: 2
  pageNum: 3
  pageNum: 4
Color : red
```

```
Compare the chinese menu and mexican menu to see does is equal : false
Compare the chinese menu and chinese2 menu to see does is equal : true
Compare the chinese menu and a clone chinese menu chinese3 to see does is equal : true
```

