1. 外观模式

某软件公司为新开发的智能手机控制与管理软件提供了一键备份功能，通过该功能可以将原本存储在手机中的通讯录、短信、照片、歌曲等资料一次性全部拷贝到移动存储个质（例如 MMC 卡或 SD 卡）中。在实现过程中需要与多个已有的类进行交互，例如通讯录管理类、短信管理类等。为了降低系统的耦合度，试使用外观模式来设计并编程模拟实现该一键备份功能。

2. 中介者模式

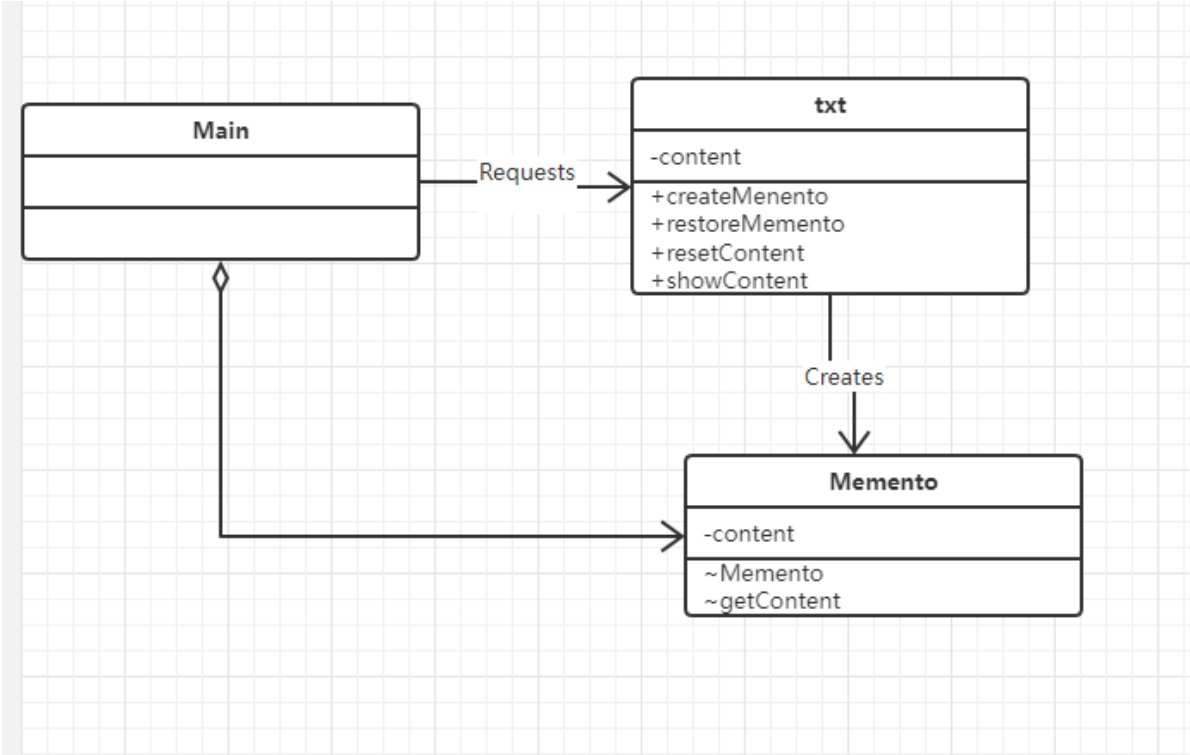
为了大力发展旅游业，某城市构建了一个旅游综合信息系统，该系统包括旅行社子系统（ravel companies Subsystem）、宾馆子系统（HotelsSubsystem）、餐厅于系统(Restaurants Subsystem)、机场子系统（Airport Subsystem）、旅游景点子系统（TourismAttractions Subsystem）等多个子系统，通过该旅游综合信息系统，各类企业之间可实现信息共享，一家企业可以将客户信息传递给其他合作伙伴。例如，当一家旅行社有一些客户后，该旅行社可以将客户信息传送到宾馆子系统、餐厅子系统、机场子系统和旅游景点子系统；宾馆也可以将顾客信息传送到旅行社子系统、餐子系统、机场子系统和旅游景点子系统；机场也可以将乘客信息传送到旅行社子系统、宾馆子系统、餐厅子系统和旅游景点子系统。由于这些子系统之间存在较为复杂的交互关系，现采用中介者模式为该旅游综合信息系统提供一个高层设计，绘制对应的类图并编程模拟实现

3. 观察者模式

某文字编辑软件须提供如下功能：在文本编辑窗口中包含一个可编辑文本区和3 个文本信息统计区，用户可以在可编辑文本区对文本进行编辑操作，第一个文本信息统计区用于显示可编辑文本区中出现的单词总数量和字符总数量，第二个文本信息统计区用于显示可编辑文本区中出现的单词（去重后按照字典序排序），第三个文本信息统计区用于按照出现频次降序显示可编辑文本区中出现的单词以及每个单词出现的次数（例如hello :5）。现采用观察者模式设计该功能，绘制对应的类图并编程模拟实现。

4. 备忘录模式

某文字编辑软件须提供撤销（Undo）和重做 / 恢复（Redo）功能，并且该软件可支持 文档对象的多步撤销和重做。开发人员决定采用备忘录模式来实现该功能，在实现过程中引人栈（Stack）作为数据结构。在实现时，可以将备忘录对象保存在两个栈中，一个栈包含用于实现撤销操作的状态对象，另一个栈包含用于实现重做操作的状态对象。在实现撤销操作时，会弹出撤销栈栈顶对象以获取前一个状态并将其设置给应用程序；同样，在实现重做操作时，会弹出重做栈栈顶对象以获取下一个状态并将其设置给应用程序。绘制对应的类图并编程模拟实现。




```
package menento_moudle;

public class Memento {
    private String content;
    Memento(String content){
        this.content=content;
    }
    String getContent(){
        return content;
    }
}
```

```
package menento_moudle;


import visitor_moudle.ScaleVisitor;

import java.util.Scanner;
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        txt t=new txt();
        Memento m=t.createMemento();
        Scanner c=new Scanner(System.in);
        Stack<Memento> undo=new Stack<Memento>();
        Stack<Memento> redo =new Stack<Memento>();
        while (true){
            System.out.println("开始操作，1修改，2显示，3撤销，4重做");
            String str=c.next();
            switch (str){
                case "1":
                    System.out.println("请输入字符串:");
```

```
                    undo.push(t.createMemento());
                    t.resetContent(c.next());
                    break;
                case "2":
                    t.showContent();
                    break;
                case "3":
                    if(undo.empty()){
                        System.out.println("近期无改写");
                        break;
                    }
                    redo.push(t.createMemento());
                    t.restoreMemento(undo.pop());
                    break;
                case "4":
                    if(redo.empty()){
                        System.out.println("近期无撤销");
                        break;
                    }
                    undo.push(t.createMemento());
                    t.restoreMemento(redo.pop());

            }
        }
    }
}
```

```
package menento_moudle;

public class txt {
    public String content="";
    public void showContent(){
        System.out.println("\r\n--------------\r\n"+content+"--------------
\r\n");
    }

    public void resetContent(String content){
        this.content=content;
    }
    public Memento createMemento(){
        return new Memento(content);
    }
    public void restoreMemento(Memento m){
        this.content=m.getContent();
    }

}
```