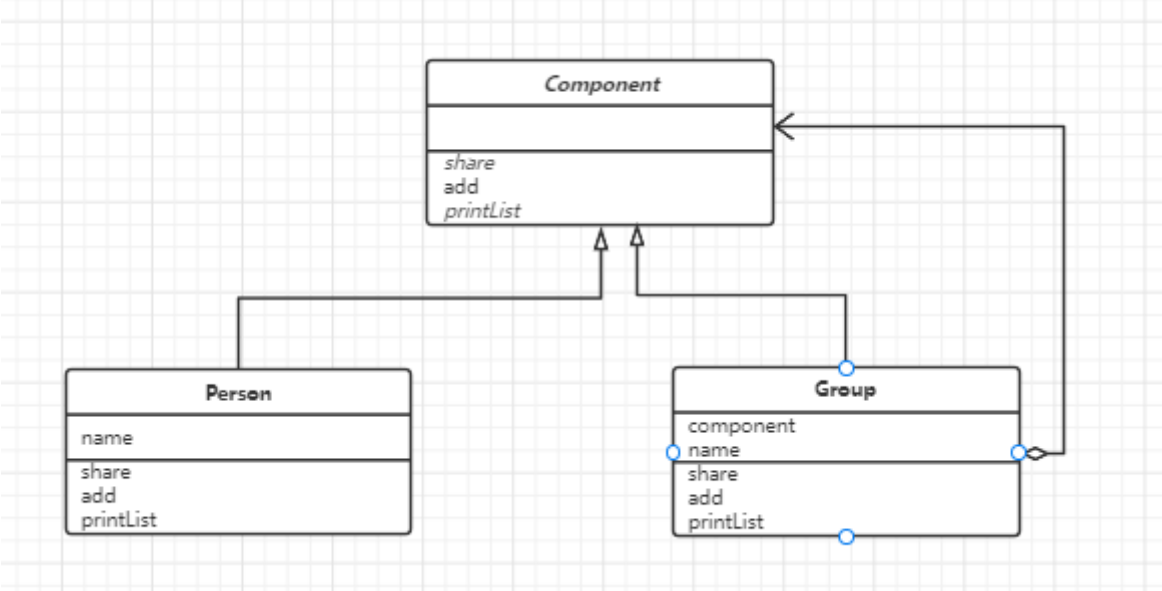
1. 组合模式

某移动社交软件要增加一个群组（Group）功能。通过设置，用户可以将自己的动态信息（包括最新动态、新上传的视频以及分享的链接等）分享给某个特定的成员（Member）. 也可以分享给某个群组中的所有成员；用户可以将成员加至某个指定的群组；此外，还允许用户在一个群组中加子群组，以便更加灵活地实现面向特定人群的信息共享。现采用组合模式设计该群组功能，绘制对应的类图并编程模拟实现。




```
package composite_moudle;

public abstract class Component {
    public abstract void share(String s);
    public void add(Component c){
        System.out.println("警告：你不应该调用这个方法");
    }
    public void printList(){
        printList("");
    }
    protected abstract void printList(String Prefix);

}
```

```
package composite_moudle;

public class Person extends Component{
    String name;
    public Person(String name){
        this.name=name;
    }
    @Override
    public void share(String s) {
        System.out.println("用户"+name+"收到信息:"+s);
    }

    @Override
```

```
    protected void printList(String Prefix) {
        System.out.println(Prefix+"/"+name);
    }

}
```

```
package composite_moudle;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Group extends Component{
    String name;
    List component=new ArrayList<Component>();
    public Group(String name){
        this.name=name;
    }
    @Override
    public void share(String s) {
        System.out.println("群聊"+name+"收到信息:"+s);
    }

    public void add(Component c){
        component.add(c);
    }

    @Override
    protected void printList(String Prefix) {
        System.out.println(Prefix+"/"+name);
        Iterator it =component.iterator();
        while (it.hasNext()){
            Component c=(Component) it.next();
            c.printList(Prefix+"/"+name);
        }
    }



}
```

2. 装饰模式

在某 OA 系统中提供一个报表生成工具，用户可以通过该工具为报表增加表头和表尾，允许用户为报表增加多个不同的表头和表尾，用户还可以自行确定表头和表尾的次序。为了能够灵活设置表头和表尾的次序并易于增加新的表头和表尾，现采用装饰模式设计该报表生成工具，绘制对应的类图并编程模拟实现。

```java
package decorator_moudle;

public abstract class Component {
    public abstract String getRowText(int i);
    public abstract int getRows();
    public void show(){
        for(int i=0;i<getRows();i++){
            System.out.println(getRowText(i));
        }
    }
}
```

```java
package decorator_moudle;

public class Context extends Component{
    String string;
    @Override
    public String getRowText(int i) {
        return string;
    }

    public Context(String str){
        this.string=str;
    }

    @Override
    public int getRows() {
        return 1;
    }
}
```

```java
package decorator_moudle;

public abstract class Border extends Component {
    public Component display;
}
```

```java
package decorator_moudle;

public class Head extends Border {
    String borderChar;

    public Head(String str,Component c){
        this.borderChar=str;
        this.display=c;
    }

    @Override
    public String getRowText(int i) {
        if(i==0)
            return "表头"+borderChar;
        else
            return this.display.getRowText(i-1);
```

```
    }

    @Override
    public int getRows() {
        return display.getRows()+1;
    }
}
```

```
package decorator_moudle;

public class Tail extends Border{
    String borderChar;
    public Tail(String str,Component c){
        this.borderChar=str;
        this.display=c;
    }

    @Override
    public String getRowText(int i) {
        if(i<display.getRows())
            return display.getRowText(i);
        else
            return "表尾"+borderChar;
    }

    @Override
    public int getRows() {
        return display.getRows()+1;
    }
}
```

```
package decorator_moudle;

public class Main {
    public static void main(String[] args) {
        Component c1=new Context("111");
        Component c2=new Head("222",c1);
        Component c3=new Tail("333",c2);

        c1.show();
        System.out.println("------");
        c2.show();
        System.out.println("------");
        c3.show();
        System.out.println("------");

    }
}
```

运行main结果如下：

```
D:\jdk1.8.0\bin\java.exe ...
111
------
表头222
111
------
表头222
111
表尾333
------

Process finished with exit code 0
```

3. 访问者模式

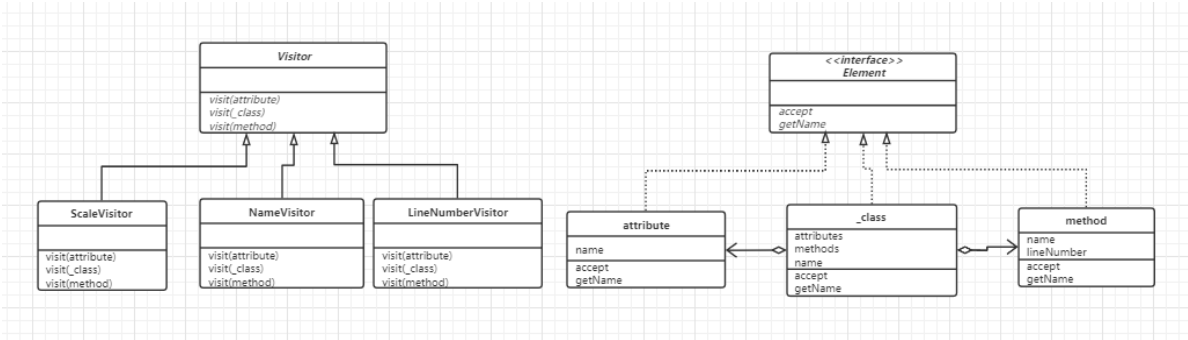
某软件公司需要设计一个源代码解析工具，该工具可以对源代码进行解析和处理，在该工具的初始版本中，主要提供了以下 3 个功能。

(1)度量软件规模。可以统计源代码中类的个数、每个类属性的个数以及每个类方法的个数等。

(2)提取标识符名称，以便检查命名是否合法和规范。可以提取类名、属性名和方法名等。

(3)统计代码行数。可以统计源代码中每个类和每个方法中源代码的行数。

将来还会在工具中增加一些新功能，为源代码中的类、属性和方法等提供更多的解析操作。现采用访问者模式设计该源代码解析工具，可将源代码中的类、属性和方法等设计为待访问的元素，上述不同功能由不同的具体访问者类实现，绘制对应的类图并编程模拟实现。




```java
package visitor_moudle;

public abstract class Visitor {
    public abstract void visit(attribute a);
    public abstract void visit(_class c);
    public abstract void visit(method m);
}
```

```java
package visitor_moudle;

public class NameVisitor extends Visitor{
    @Override
    public void visit(attribute a) {
        System.out.println("属性的名称"+a.getName());
    }

    @Override
```

```
    public void visit(_class c) {
        System.out.println("类的名称"+c.getName());
    }

    @Override
    public void visit(method m) {
        System.out.println("方法的名称"+m.getName());
    }
}
```

```
package visitor_moudle;

public class ScaleVisitor extends Visitor{
    @Override
    public void visit(attribute a) {
        throw new RuntimeException("你不该访问它的！");
    }

    @Override
    public void visit(_class c) {
        System.out.printf("类有%d个属性，有%d个方法
\n",c.attributes.size(),c.methods.size());
    }

    @Override
    public void visit(method m) {
        throw new RuntimeException("你不该访问它的！");
    }
}
```

```
package visitor_moudle;

public class LineNumberVisitor extends Visitor{
    @Override
    public void visit(attribute a) {
        throw new RuntimeException("你不该访问它的！");
    }

    @Override
    public void visit(_class c) {
        int res=0;
        for(method m:c.methods){
            res+=m.lineNumber;
        }
        System.out.printf("方法的行数+%d\n",res);
    }

    @Override
    public void visit(method m) {
        System.out.printf("方法的行数+%d\n",m.lineNumber);
    }
}
```

```java
package visitor_moudle;

public interface Element {
    public void accept(Visitor v);
    public String getName();
}
```

```java
package visitor_moudle;

public class attribute implements Element{

    String name;
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }

    @Override
    public String getName() {
        return name;
    }
}
```

```java
package visitor_moudle;

import java.util.ArrayList;
import java.util.List;

public class _class implements Element{
    String name;
    List<attribute> attributes=new ArrayList<attribute>();
    List<method> methods=new ArrayList<method>();
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }

    @Override
    public String getName() {
        return name;
    }
}
```

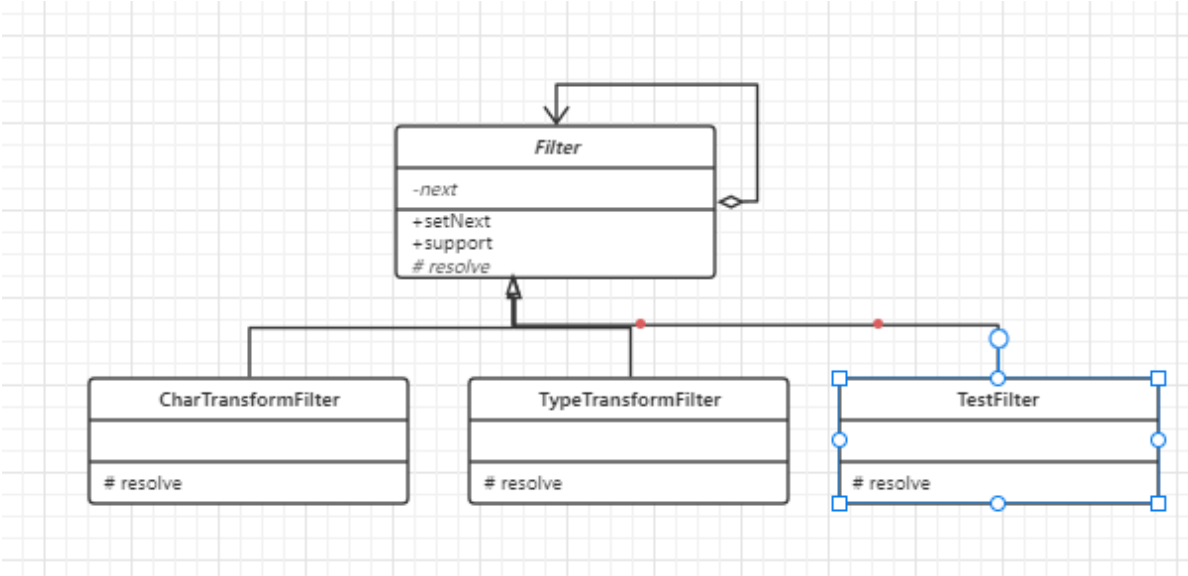
```java
package visitor_moudle;

public class method implements Element{
    String name;
    int lineNumber;
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }
```

```
    @Override
    public String getName() {
        return name;
    }
}
```

4. 职责链模式

在某 Web 框架中采用职责链模式来组织数据过滤器，不同的数据过滤器提供了不同的功能，例如字符编码转换过滤器、数据类型转换过滤器、数据校验过滤器等，可以将多个过滤器连接成一一个过滤器链，进而对数据进行多次处理。根据以上描述，绘制对应的类图并 编程模拟实现。




```
package chain_moudle;

public abstract class Filter {
    private Filter next;
    public void setNext(Filter f){
        next=f;
    }
    public void support(int i){
        if(resolve(i)){

        }else if(next!=null){
            next.support(i);
        }else{
            System.out.printf("%d号问题无法处理\n",i);
        }
    }
    protected abstract boolean resolve(int i);
}
```

```java
package chain_moudle;

public class CharTransformFilter extends Filter{

    @Override
    protected boolean resolve(int i) {
        if(i<10){
            System.out.printf("%d号问题被编码转换过滤器处理结束\n",i);
            return true;
        }
        return false;
    }
}
```

```java
package chain_moudle;

public class TypeTransformFilter extends Filter{
    @Override
    protected boolean resolve(int i) {
        if(i%2==0){
            System.out.printf("%d号问题被类型转换过滤器处理结束\n",i);
            return true;
        }
        return false;
    }
}
```

```java
package chain_moudle;

public class TestFilter extends Filter{

    @Override
    protected boolean resolve(int i) {
        if(i%3==0){
            System.out.printf("%d号问题被数据校验过滤器处理结束\n",i);
            return true;
        }
        return false;
    }
}
```

```java
package chain_moudle;

public class Main {
    public static void main(String[] args) {
        Filter f=new CharTransformFilter();
        f.setNext(new TypeTransformFilter());
        f.setNext(new TestFilter());
        for(int i=0;i<100;i++){
            f.support(i);
        }
    }
}
```

```
D:\jdk1.8.0\bin\java.exe ...
0号问题被编码转换过滤器处理结束
1号问题被编码转换过滤器处理结束
2号问题被编码转换过滤器处理结束
3号问题被编码转换过滤器处理结束
4号问题被编码转换过滤器处理结束
5号问题被编码转换过滤器处理结束
6号问题被编码转换过滤器处理结束
7号问题被编码转换过滤器处理结束
8号问题被编码转换过滤器处理结束
9号问题被编码转换过滤器处理结束
10号问题被类型转换过滤器处理结束
11号问题无法处理
12号问题被类型转换过滤器处理结束
13号问题无法处理
14号问题被类型转换过滤器处理结束
```