定义：责任链模式(Chain of Responsibility)使多个对象都有机会处理请求，从而避免请求的发送者和接受者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到有对象能够处理它。

类型：行为类模式

考虑以下这样的场景：

你要去给某公司借款 1 万元，当你来到柜台的时候向柜员发起 "借款 1 万元" 的请求时，柜员认为金额太多，处理不了这样的请求，他转交这个请求给他的组长，组长也处理不了这样的请求，那么他接着向经理转交这样的请求。

用 Java 代码表示为：

**public** **void** **test(**Request request**)** **{**

**int** money **=** request**.**getRequestMoney**();**

**if(**money **<=** 1000**)** **{**

Clerk**.**response**(**request**);**

**}** **else** **if(**money **<=** 5000**)** **{**

Leader**.**response**(**request**);**

**}** **else** **if(**money **<=** 10000**)** **{**

Manager**.**response**(**request**);**

**}**

**}**

代码的业务逻辑就是这样：根据的借款金额来判定谁来处理这个借款请求 (request)

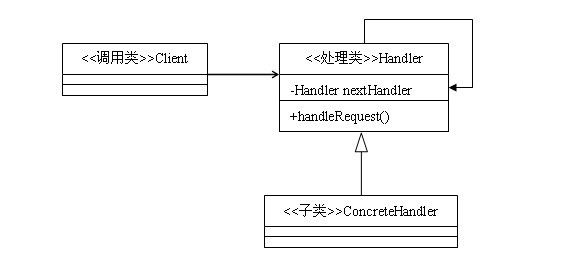
* 如果请求借款金额小于 1000 元，那么柜台职员就可以直接处理这个请求(比如签字)
* 如果请求借款金额小于 5000 元但大于 1000 元，那么职员处理不了，该请求转交给组长，组长能够处理这样的请求(比如签字)
* 如果请求借款金额大于 5000 元但小于 10000 元，那么职员和组长都处理不了(没有权限)，那么这个请求就会转交给经理，经理能够处理这样的请求(比如签字)

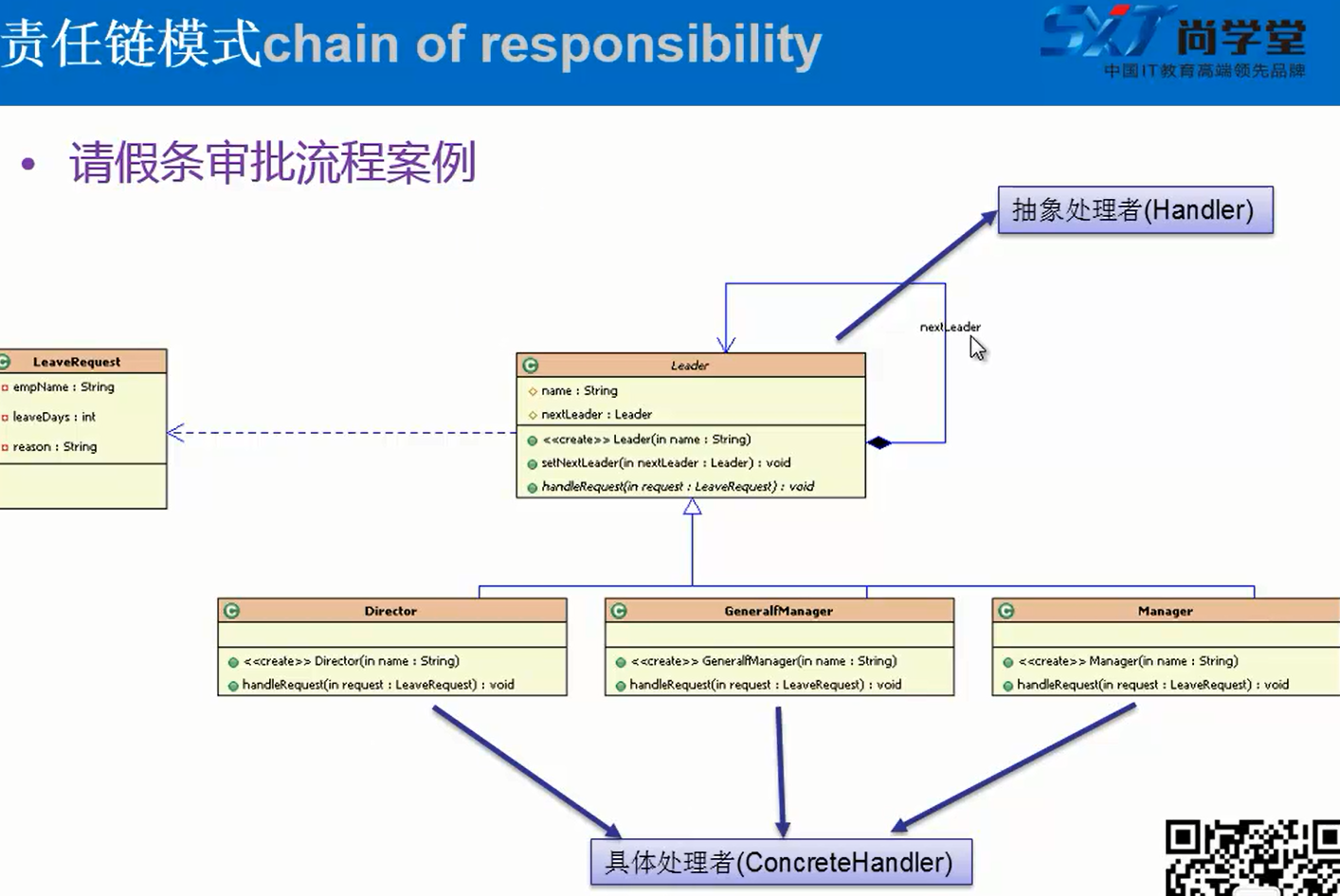
在编程中，这样处理业务逻辑的方法非常常见，这样的方法非常直观，简单明了，并且也比较容易维护，但是同时它也存在着着问题:

1. 代码臃肿: 实际应用中的判定条件通常不是这么简单地判断金额，也许需要复杂的操作，也许需要查询数据库等等，这就会产生许多额外的代码，如果判断条件再比较多的话，那么代码就会大量地堆积在同一个文件中。
2. 耦合度高：如果我们想继续添加处理请求的类，那么就需要添加 else if 的判定条件；另外，这个条件判定的顺序也是写死的。如果想改变顺序，那么也只能修改这个条件语句。

下面就介绍责任链模式：

**责任链模式的类图非常简单，如下图：**





1. **抽象处理类:** 主要包含一个指向下一处理类的成员变量 nextHandler 和一个处理请求的方法 handRequest，handRequest 方法的主要思想是，如果满足处理的条件，则有本处理类来进行处理，否则由 nextHandler 来处理。**(定义handRequest接口让具体处理类去实现)**

2. **具体处理类：**具体处理类的主要是对具体的处理逻辑和处理的适用条件进行实现。

了解完责任链的大体思想之后，在看看 Java 代码是如何实现的：

package design.responsibility;  
  
public class LeaveRequest {  
    private String name;  
    private Integer day;  
  
    public LeaveRequest(String name, Integer day) {  
        C:\Users\郑学炜\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = name;  
        this.day = day;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        C:\Users\郑学炜\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = name;  
    }  
  
    public Integer getDay() {  
        return day;  
    }  
  
    public void setDay(Integer day) {  
        this.day = day;  
    }  
}

public abstract class Leader {  
  
    protected String name;  
    protected Leader nextLeader;  
  
    public Leader(String name) {  
        C:\Users\郑学炜\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = name;  
    }  
  
    public void setNextLeader(Leader nextLeader) {  
        this.nextLeader = nextLeader;  
    }  
  
    public abstract void handlerequest(LeaveRequest request) ;

}

package design.responsibility;  
  
public class Diector extends Leader {  
  
    public Diector(String name) {  
        super(name);  
    }  
  
    @Override  
    public void handlerequest(LeaveRequest request) {  
        if (request.getDay() < 3) {  
            System.out.println("经理批准");  
        } else {  
            this.nextLeader.handlerequest(request);  
        }  
    }  
}  
package design.responsibility;  
  
public class Manager extends Leader {  
    public Manager(String name) {  
        super(name);  
    }  
  
    @Override  
    public void handlerequest(LeaveRequest request) {  
        if (request.getDay() < 10) {  
            System.out.println("老总批准");  
        } else {  
            this.nextLeader.handlerequest(request);  
        }  
    }  
}  
package design.responsibility;  
  
public class GeneralManager extends Leader {  
    public GeneralManager(String name) {  
        super(name);  
    }  
  
    @Override  
    public void handlerequest(LeaveRequest request) {  
        if (request.getDay() < 30) {  
            System.out.println("董事长批准");  
        } else {  
            System.out.println("辞职吧");  
        }  
    }  
}  
package design.responsibility;  
  
public class test {  
    public static void main(String[] args) {  
        Leader a = new Diector("小王");  
        Leader b = new Manager("大王");  
        Leader c = new GeneralManager("王炸");  
        a.setNextLeader(b);  
        b.setNextLeader(c);  
        LeaveRequest leaveRequest = new LeaveRequest("小红", 50);  
        a.handlerequest(leaveRequest);  
  
    }  
}