



# 课堂须知

---

- 全程带好口罩
- 扫台上的二维码登记
- 如有发热、咳嗽等不适请立即联系老师



哈爾濱工業大學(深圳)  
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

規格嚴格



功夫到家

1920 — 2017

# 面向对象的软件构造导论

## 实验六：观察者模式和模板模式

2022春

哈尔滨工业大学（深圳）



# 本学期实验总体安排

实验项目	一	二	三	四	五	六
学时数	2	2	2	4	2	4
实验内容	飞机大战 功能分析	单例模式 工厂模式	Junit与单 元测试	策略模式 数据访问 对象模式	Swing 多线程	模板模式 观察者模式
分数	4	6	4	6	6	14
提交内容	UML类图、 代码	UML类图、 代码	单元测试 代码 测试报告	UML类图、 代码	代码	项目代码、 实验报告

实验课程共**16**个学时，**6**个实验项目，总成绩为**40**分。



# 目录

---

01

实验目的

02

实验任务

03

实验步骤

04

实验要求

05

作业提交



# 实验目的

---

- 深入理解观察者模式和模板模式的模式动机和意图，掌握模式结构；
- 掌握绘制观察者模式和模板模式的UML类图；
- 能熟练使用代码实现观察者模式和模板模式。



# 实验任务

---

绘制类图、重构代码，完成以下功能：

1. 采用观察者模式实现炸弹道具；
2. 采用模板模式实现简单、普通、困难三种游戏难度。

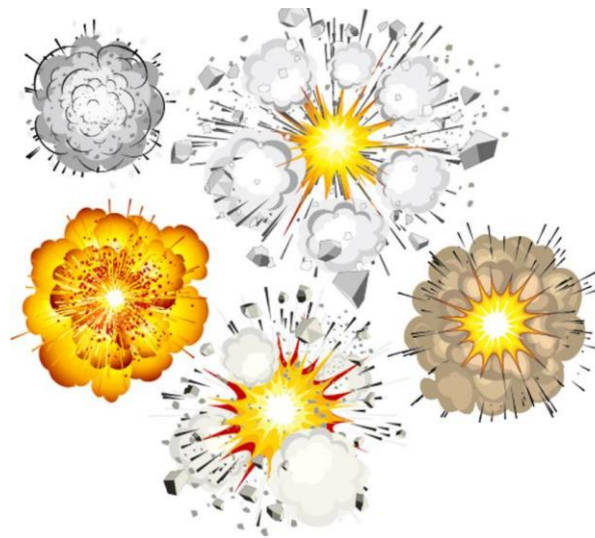


# 实验步骤

## 1 炸弹应用场景分析

在游戏中，精英敌机和Boss机坠毁时会以一定概率掉落**炸弹道具**。英雄机碰撞炸弹道具后，道具生效，可**清除界面**上除Boss机外的所有敌机和敌机子弹，并把炸掉的敌机分数加到英雄机得分里。

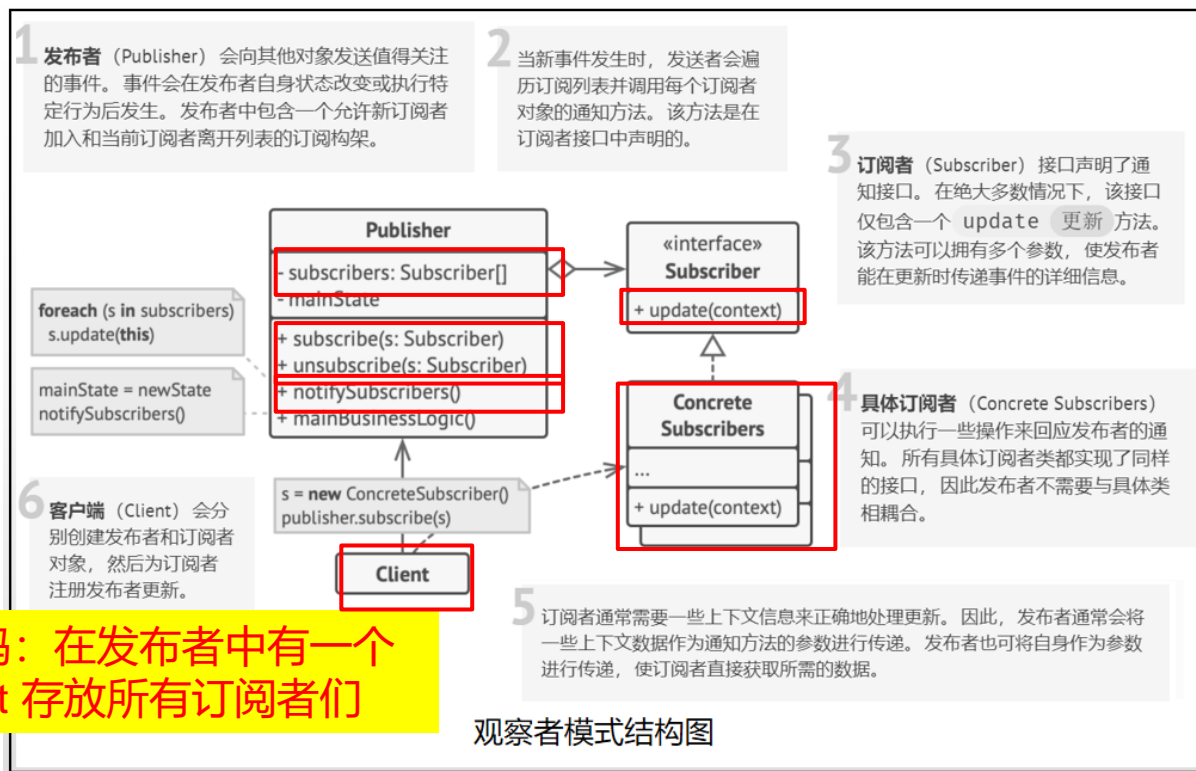
观察者模式



# 实验步骤

## 2 绘制观察者模式类图

**观察者模式** (Observer Pattern) 也是一种**行为型**设计模式，允许定义一种订阅机制，可在对象事件发生时通知多个“观察”该对象的其他对象。







# 实验步骤

## 2 绘制观察者模式类图

假如我们要用观察者模式实现一个功能：**人民币汇率对进出口公司的影响**。我们该如何绘制UML类图？





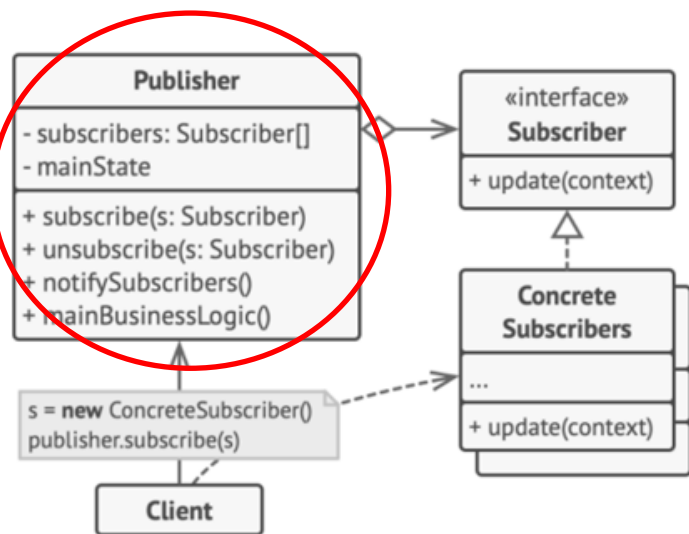
# 实验步骤

## 2

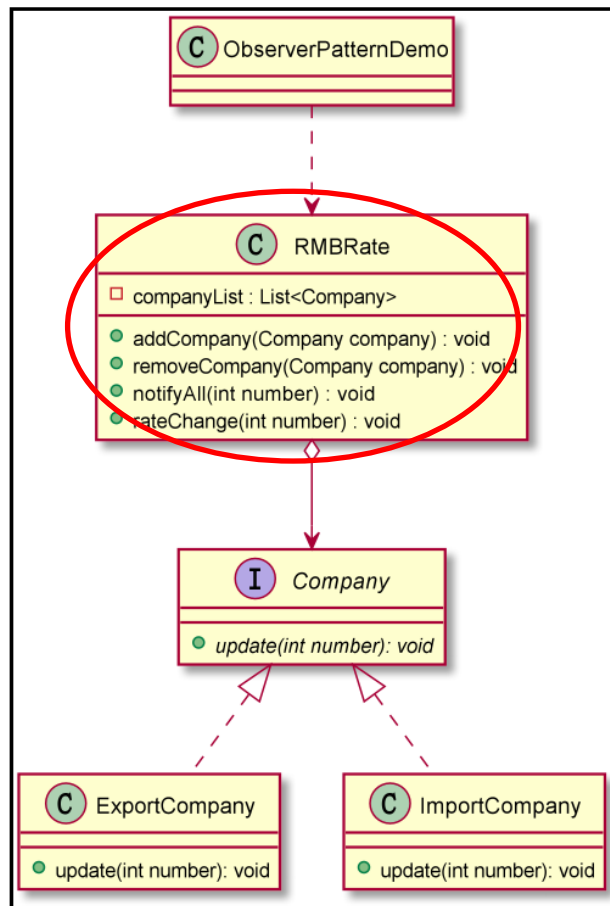
## 绘制观察者模式类图

举个例子：人民币汇率波动对进出口公司的影响

1、创建RMBRate作为发布者，它带有绑定观察者的方法；



观察者模式结构图



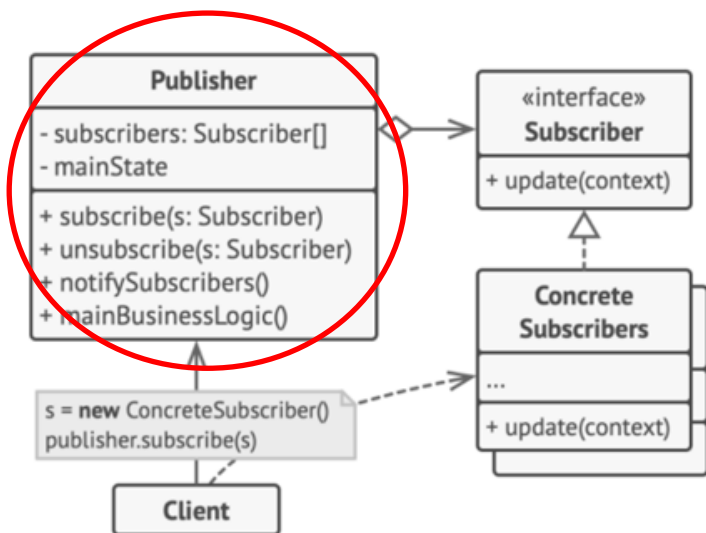


# 实验步骤

## 2 绘制观察者模式类图

思考问题:

- 1、飞机大战里面，炸弹爆炸这个功能，谁是发布者？
- 2、订阅者清单，存的是什么对象？



观察者模式结构图



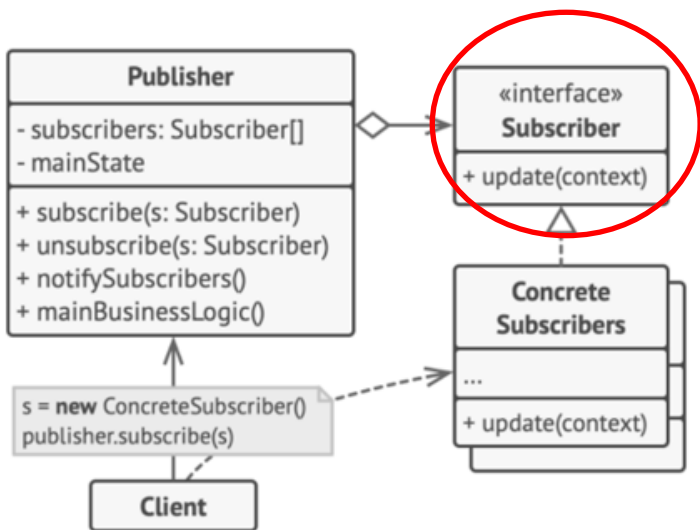


# 实验步骤

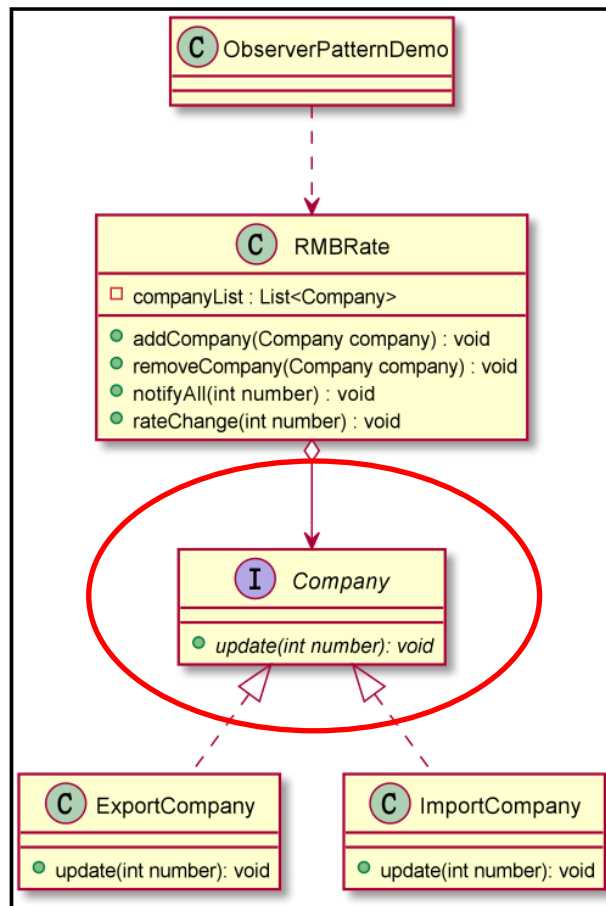
## 2 绘制观察者模式类图

举个例子：人民币汇率波动对进出口公司的影响

2、创建 **Company**接口，充当订阅者角色；



观察者模式结构图





# 实验步骤

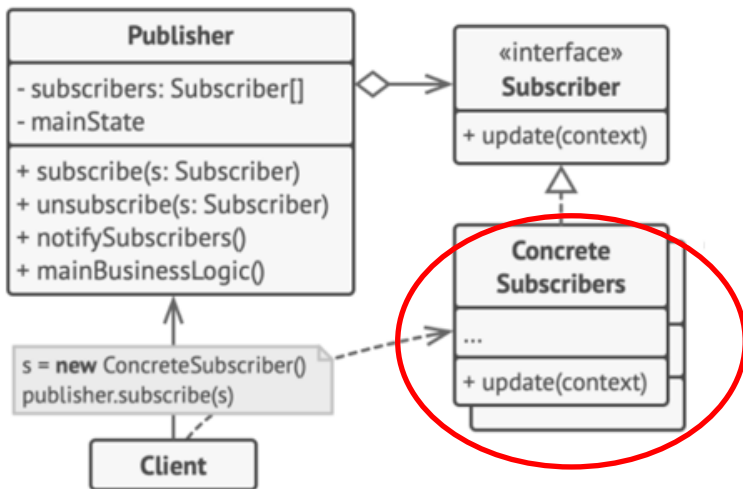
## 2

## 绘制观察者模式类图

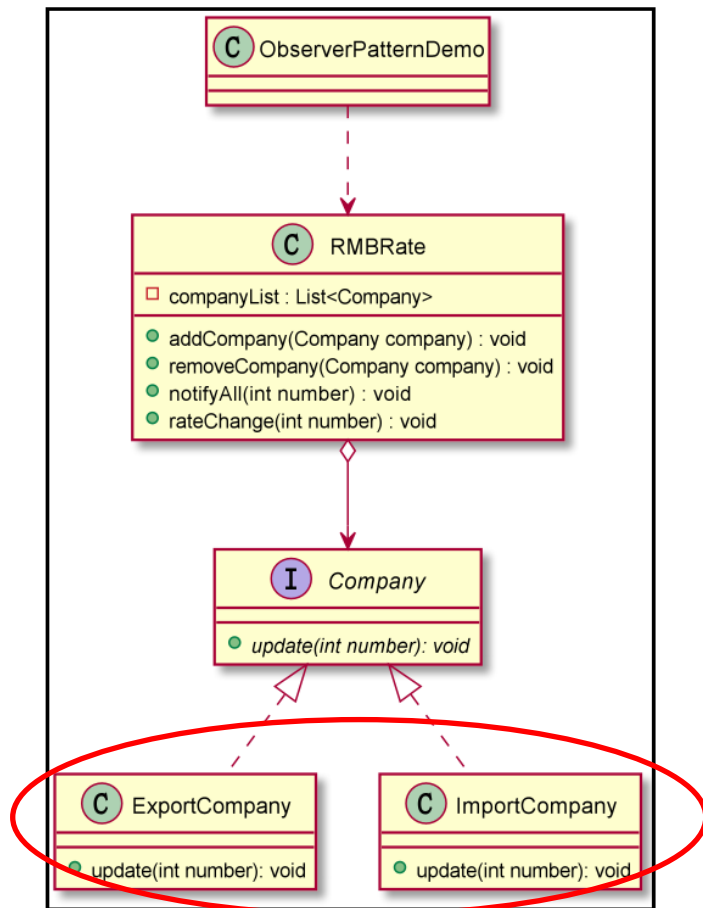
思考：炸弹爆炸这个功能，我们的update方法里面要做什么动作？

举个例子：人民币汇率波动对进出口公司的影响

### 3、创建实现订阅者接口的实体类；



观察者模式结构图





# 实验步骤

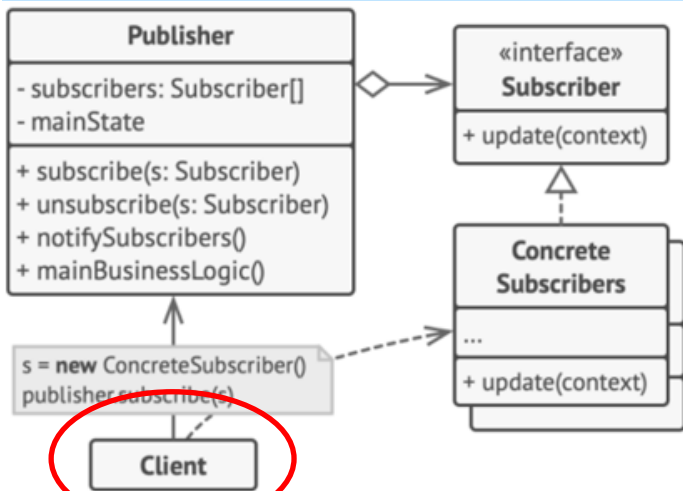
## 2

## 绘制观察者模式类图

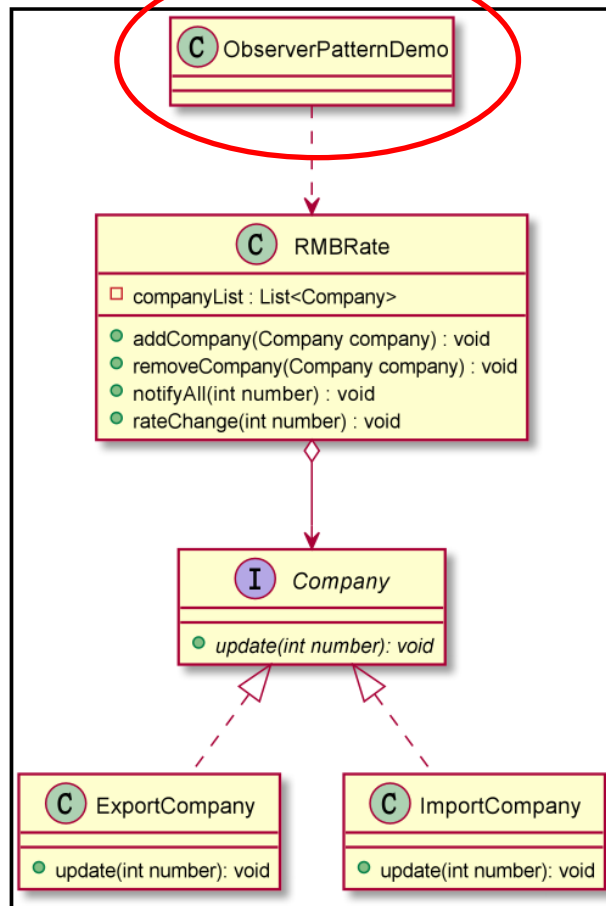
思考：炸弹爆炸这个功能中，谁是client?

举个例子：人民币汇率波动对进出口公司的影响

4、ObserverPatternDemo使用RMBRate对象和Company实体类对象来演示观察者模式。



观察者模式结构图





# 实验步骤

## 3

## 重构代码，实现观察者模式

- 观察者模式的代码实现方式示例：

- ① 创建 RMBRate 类，充当发布者角色。

- ② 创建 Company 接口，充当订阅者角色。

```
public interface Company {  
    /**  
     * 对汇率的反应  
     * @param number 汇率  
     */  
    void update(int number);  
}
```

订阅者

```
public class RMBRate {  
  
    //观察者列表  
    private List<Company> companyList = new ArrayList<>();  
  
    //增加观察者  
    public void addCompany(Company company) {  
        companyList.add(company);  
    }  
  
    //删除观察者  
    public void removeCompany(Company company) {  
        companyList.remove(company);  
    }  
  
    //通知所有观察者  
    public void notifyAll(int number) {  
        for (Company company : companyList) {  
            company.update(number);  
        }  
    }  
  
    //人民币汇率改变  
    public void rateChange (int number) {  
        notifyAll(number);  
    }  
}
```

发布者





# 实验步骤

## 3

## 重构代码，实现观察者模式

- 观察者模式的代码实现方式示例：

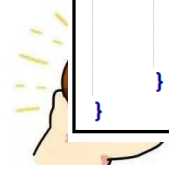
③ 创建Company接口实体类，充当具体订阅者角色。

```
public class ImportCompany implements Company {  
    @Override  
    public void update(int number) {  
        System.out.print("进口公司收到消息: ");  
        if (number > 0) {  
            System.out.println("人民币汇率升值" + number + "个基点，进口产品成本降低，公司利润提升。");  
        } else if (number < 0) {  
            System.out.println("人民币汇率贬值" + (-number) + "个基点，进口产品成本提高，公司利润降低。");  
        }  
    }  
}
```

具体订阅者

```
public class ExportCompany implements Company {  
    @Override  
    public void update(int number) {  
        System.out.print("出口公司收到消息: ");  
        if (number > 0) {  
            System.out.println("人民币汇率升值" + number + "个基点，出口产品收入降低，公司销售利润降低。");  
        } else if (number < 0) {  
            System.out.println("人民币汇率贬值" + (-number) + "个基点，出口产品收入提高，公司销售利润降低提高。");  
        }  
    }  
}
```

具体订阅者





# 实验步骤

## 3 重构代码，实现观察者模式

### ● 观察者模式的代码实现方式示例：

④ Client类中使用RMBRate对象和Company实体类对象来演示观察者模式。

人民币汇率改变：

进口公司收到消息：人民币汇率升值 10 个基点，进口产品成本降低，公司利润提升。

出口公司收到消息：人民币汇率升值 10 个基点，出口产品收入降低，公司销售利润降低。

人民币汇率改变：

进口公司收到消息：人民币汇率贬值 5 个基点，进口产品成本提高，公司利润降低。

出口公司收到消息：人民币汇率贬值 5 个基点，出口产品收入提高，公司销售利润降低提高。

人民币汇率改变：

出口公司收到消息：人民币汇率升值 8 个基点，出口产品收入降低，公司销售利润降低。

```
public class ObserverPatternDemo {  
    public static void main(String[] args) {  
  
        RMBRate rate = new RMBRate();  
        Company company1 = new ImportCompany();  
        Company company2 = new ExportCompany();  
  
        rate.addCompany(company1);  
        rate.addCompany(company2);  
  
        System.out.println("人民币汇率改变：");  
        rate.rateChange(10);  
  
        System.out.println("人民币汇率改变：");  
        rate.rateChange(-5);  
  
        rate.removeCompany(company1);  
  
        System.out.println("人民币汇率改变：");  
        rate.rateChange(8);  
  
    }  
}
```



# 实验步骤

## 4

### 难度选择应用场景分析

用户进入飞机大战游戏界面后，可选择某种游戏难度：**简单** / **普通** / **困难**。用户选择后，出现该难度对应的地图，且游戏难度会相应调整。





# 实验步骤

## 4

## 难度选择应用场景分析

游戏难度设置可考虑如下因素（至少设置5个）：

- 游戏界面中同一时刻出现的敌机数量的最大值
- 敌机的属性值，如血量、速度
- 英雄机的射击周期
- 敌机的射击周期
- 精英敌机的产生概率
- 普通和精英敌机的产生周期
- Boss敌机产生的得分阈值
- ...

模板模式

	简单	普通	困难
Boss 敌机	无	有 每次召唤不改变 Boss 机血量	有 每次召唤提升 Boss 机血量
难度是否随时间增加	否	是	是

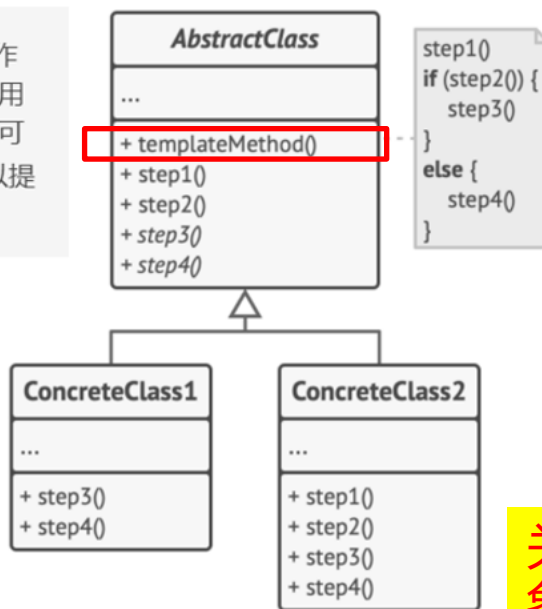
## 5

## 绘制模板模式类图

**模板模式** (Template Pattern) 是一种**行为型**设计模式，它在抽象类中定义了一个算法的框架，允许子类在不修改结构的情况下重写算法的特定步骤。

1 抽象类 (AbstractClass) 会声明作为算法步骤的方法，以及依次调用它们的实际模板方法。算法步骤可以被声明为抽象类型，也可以提供一些默认实现。

2 具体类 (ConcreteClass) 可以重写所有步骤，但不能重写模板方法自身。



模板模式结构图

关键代码：模板方法在抽象类实现，某些特定步骤在子类实现



# 实验步骤

---

## 5

## 绘制模板模式类图

- 模板模式代码示例（银行业务办理）：

步骤：

- 1、取号排队
- 2、办理业务（存款、取款、转账.....）
- 3、对银行工作人员进行评分



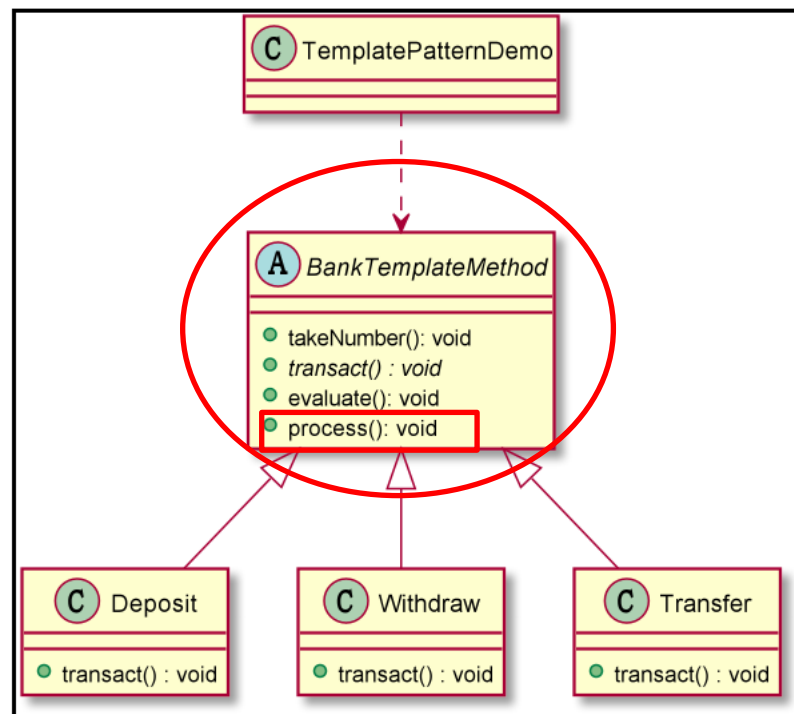
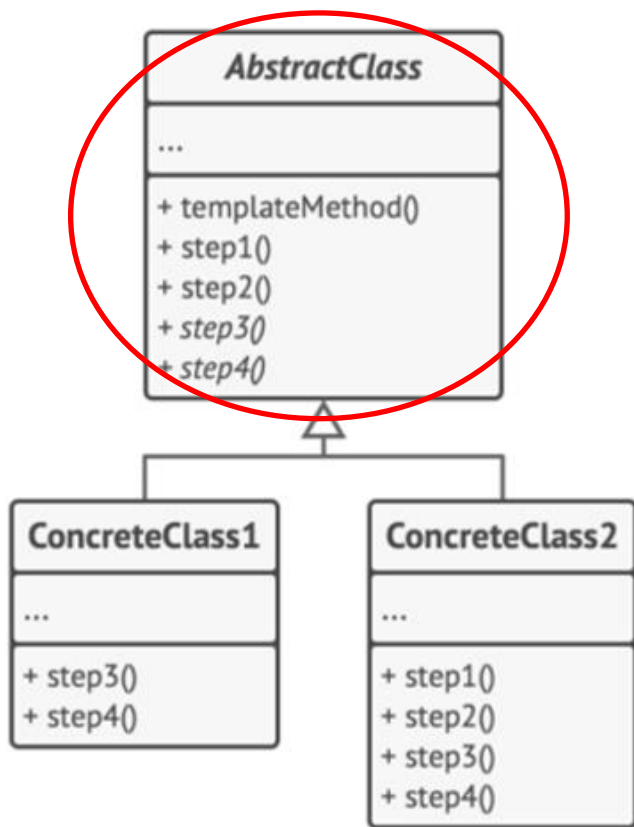


# 实验步骤

## 5

## 绘制模板模式类图

- 模板模式代码示例（银行业务办理）：





# 实验步骤

## 6 重构代码，实现模板模式

- 模板模式代码示例：

① 创建一个抽象类，它的模板方法被设置为 final。

```
public abstract class BankTemplateMethod {  
    public final void takeNumber()  
    {  
        System.out.println("取号排队");  
    }  
  
    public abstract void transact();  
  
    public void evaluate()  
    {  
        System.out.println("反馈评分");  
    }  
  
    public final void process()  
    {  
        this.takeNumber();  
        this.transact();  
        this.evaluate();  
    }  
}
```

模板方法

② 创建扩展了上述类的实体类，它们重写了抽象类的某些方法。

```
public class Deposit extends BankTemplateMethod {  
    @Override  
    public void transact() {  
        System.out.println("存款");  
    }  
}  
  
public class Transfer extends BankTemplateMethod {  
    @Override  
    public void transact() {  
        System.out.println("转账");  
    }  
}  
  
public class Withdraw extends BankTemplateMethod {  
    @Override  
    public void transact() {  
        System.out.println("取款");  
    }  
}
```



# 实验步骤

## 4

## 重构代码，实现模板模式

- 模板模式代码示例：

③ 使用BankTemplateMethod 的模板方法 process() 来演示模板模式。

```
public class TemplatePatternDemo {  
  
    public static void main(String[] args) {  
  
        BankTemplateMethod bank;  
        System.out.println("顾客1: ");  
        bank = new Deposit();  
        bank.process();  
  
        System.out.println("顾客2: ");  
        bank = new Withdraw();  
        bank.process();  
  
        System.out.println("顾客3: ");  
        bank = new Transfer();  
        bank.process();  
  
    }  
}
```

顾客 1:

取号排队

存款

反馈评分

顾客 2:

取号排队

取款

反馈评分

顾客 3:

取号排队

转账

反馈评分





# 实验要求

---

- 本次实验提交版本需完成以下功能：
  - ✓ 设计并实现三种游戏难度，其中普通和困难模式随着游戏时长增加而提升难度（控制台输出），且当得分每超过一次阈值，则产生一次Boss机；
  - ✓ 炸弹道具生效时清除界面上除Boss机外的所有敌机和敌机子弹，被清除的敌机计入英雄机得分。

# 实验要求

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AircraftWar-base [D:\code\java\AircraftWar-base] - Main.java

AircraftWar-base > src > edu > hitsz > application > Main

Project

- aircraft
  - application
    - game
      - HeroController
      - ImageManager
      - Main
      - MusicThread
    - basic
    - bullet
    - factory
    - ranklist

Main.java

```
1 package edu.hitsz.applic
2
3 import edu.hitsz.MenuBoa
4 import edu.hitsz.ScoreBo
5 import edu.hitsz.applica
6
7 import javax.swing.*;
8 import java.awt.*;
9
10
```

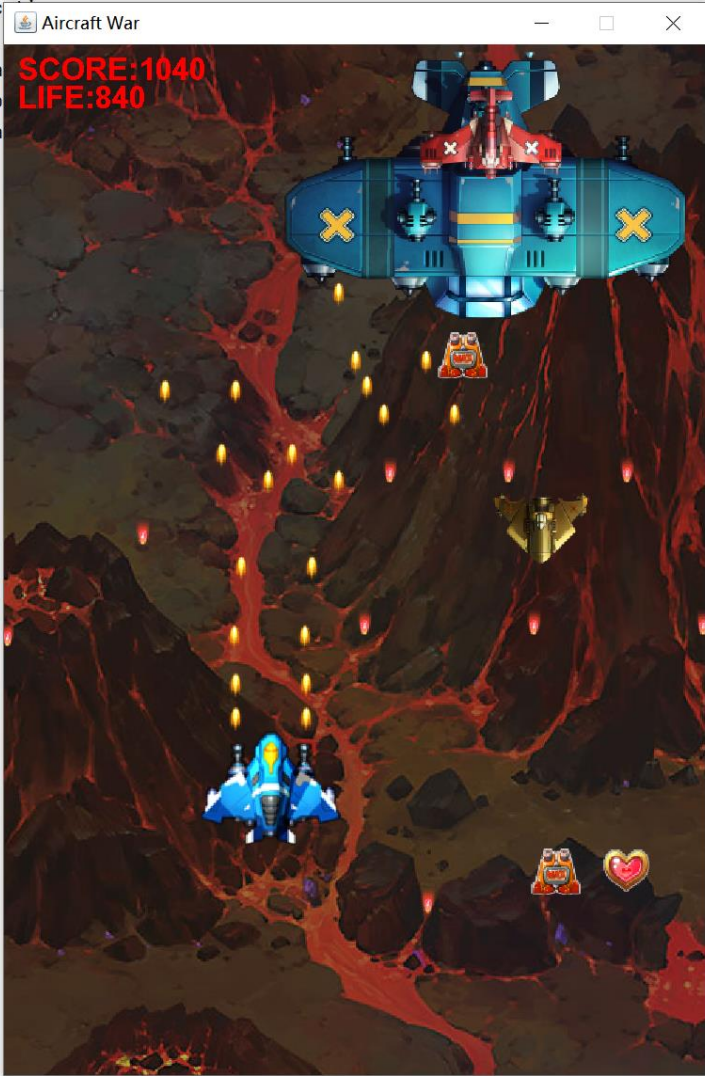
Run: Main

No supply generated!  
No supply generated!  
提高难度! 精英机概率:0.20,敌机周期:19.60, 敌机属性提升倍率:1.02。  
FireSupply active  
提高难度! 精英机概率:0.21,敌机周期:19.21, 敌机属性提升倍率:1.04。  
No supply generated!  
提高难度! 精英机概率:0.21,敌机周期:18.82, 敌机属性提升倍率:1.06。  
No supply generated!  
HpSupply active  
提高难度! 精英机概率:0.22,敌机周期:18.45, 敌机属性提升倍率:1.08。  
No supply generated!  
产生BOSS敌机  
Boss敌机血量倍率:1.10。  
FireSupply active  
BombSupply active  
提高难度! 精英机概率:0.22,敌机周期:18.08, 敌机属性提升倍率:1.10。  
No supply generated!  
No supply generated!  
BombSupply active  
HpSupply active  
FireSupply active  
提高难度! 精英机概率:0.23,敌机周期:17.72, 敌机属性提升倍率:1.13。  
产生BOSS敌机  
Boss敌机血量倍率:1.21。

随游戏时长增加提升难度

Aircraft War

SCORE:1040  
LIFE:840





# 作业提交

---

## • 提交内容

- ① 提交完整的项目代码（包含UML类图）；
- ② 提交一段游戏的视频（小于2min），展示你的游戏的所有功能点；
- ③ 提交实验报告（按照实验报告模板）。

## • 截止时间

实验课后一周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。



**同学们  
请开始实验吧！**