



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

規格嚴格



功夫到家

设计模式实验 (3)

实验六：观察者模式和模板模式

2023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	2	4 (2+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	观察者模式 模板模式
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

作业提交



实验目的

- 理解**观察者模式**和**模板模式**的意义，掌握模式结构；
- 掌握绘制观察者和模板模式的**UML类图**；
- 熟练使用**代码实现**观察者和模板模式。



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**。

注意： 结合飞机大战实例，完成模式UML类图设计后，再进行编码，**先“设计”再“编码”**！

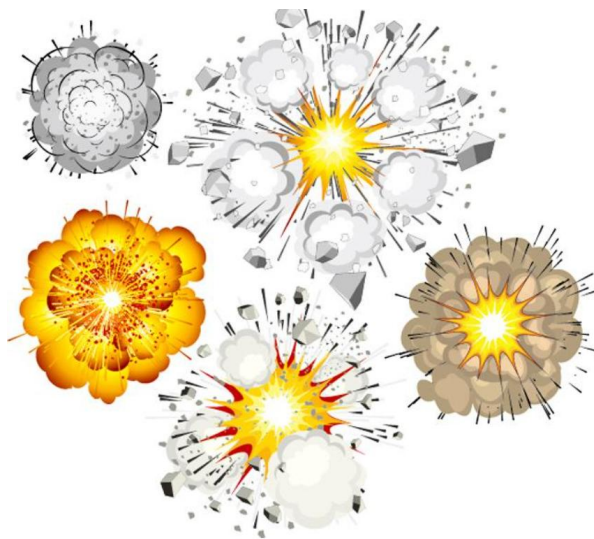


实验步骤

1 炸弹应用场景分析

应用场景
分析

精英和Boss敌机坠毁时会以较低概率掉落**炸弹道具**。它可**清除界面**上除Boss机外的所有敌机和敌机子弹。**Boss敌机血量减少。**
英雄机可获得坠毁的敌机分数。

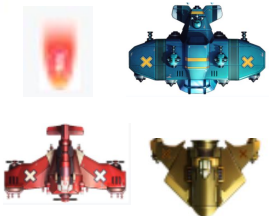


1 炸弹应用场景分析

请思考：

1. 在哪个类实现炸弹爆炸？有哪些角色对炸弹有响应？如何响应？

```
public class BombSupply extends AbstractFlyingSupply{  
    public BombSupply(int locationX, int locationY, int speedX, int speedY) {  
        super(locationX, locationY, speedX, speedY);  
    }  
  
    @Override  
    public void activate() {  
        System.out.println("BombSupply active!");  
    }  
}
```



违反
单一职责

X

违反
开闭原则

X

2. 若增加一个对炸弹有响应的新角色，如敌方炮台，需要修改哪些代码？

违反
依赖倒转

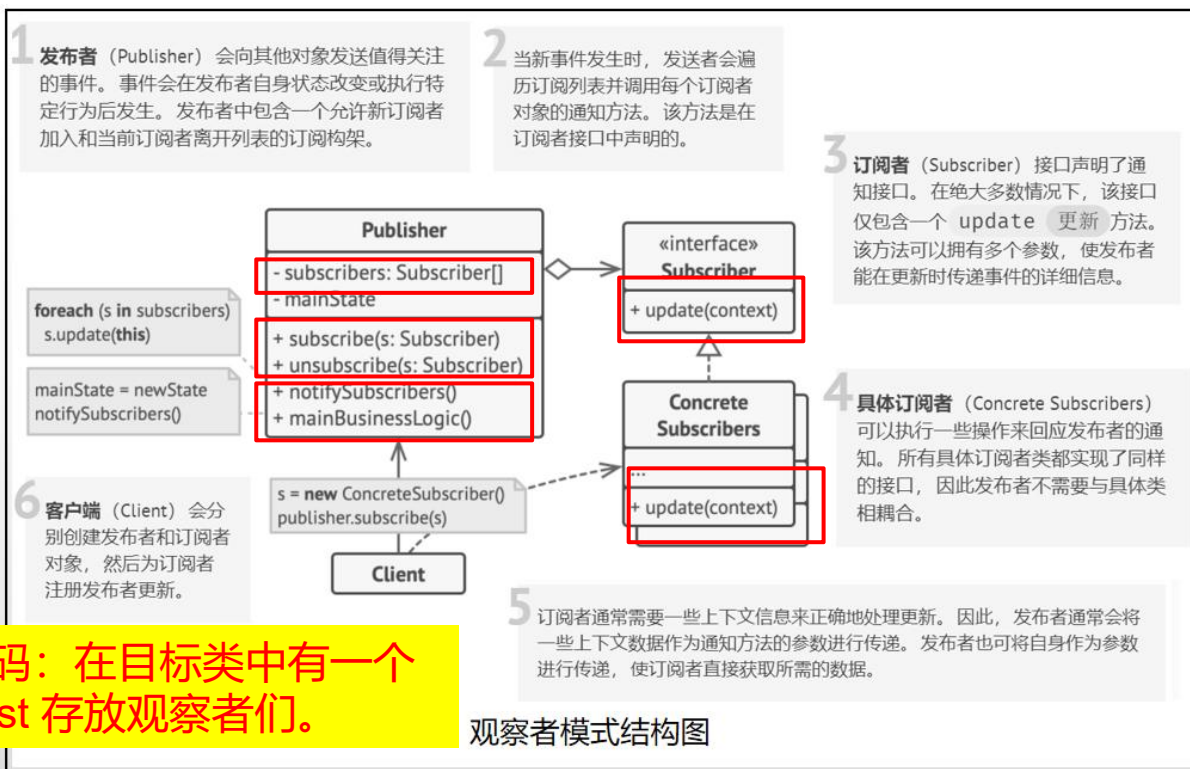
X

3. 若增加一个新型道具，如减速道具，需要修改哪些代码？

实验步骤

2 绘制观察者模式类图

观察者模式 (Observer Pattern) 也是一种行为设计模式，允许你定义一种订阅机制，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

假如我们要实现一个功能：观察人民币汇率波动对进出口公司的影响。我们该如何绘制UML类图？



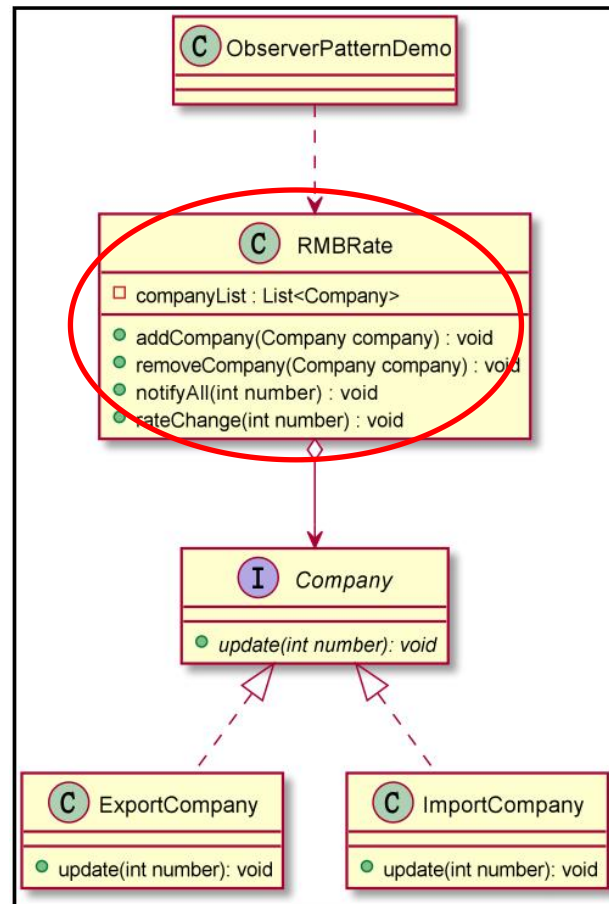


实验步骤

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；



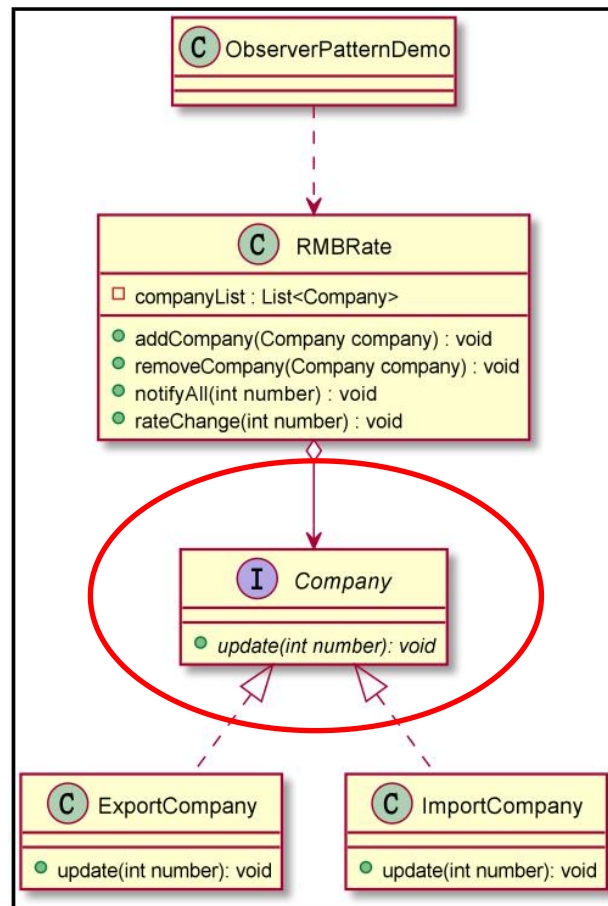


实验步骤

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；



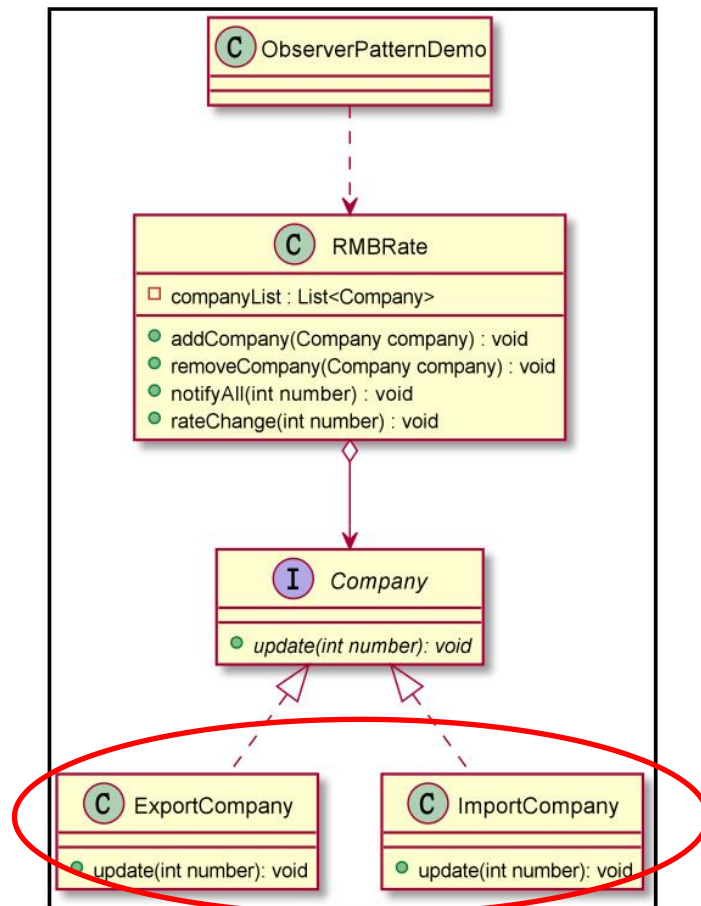
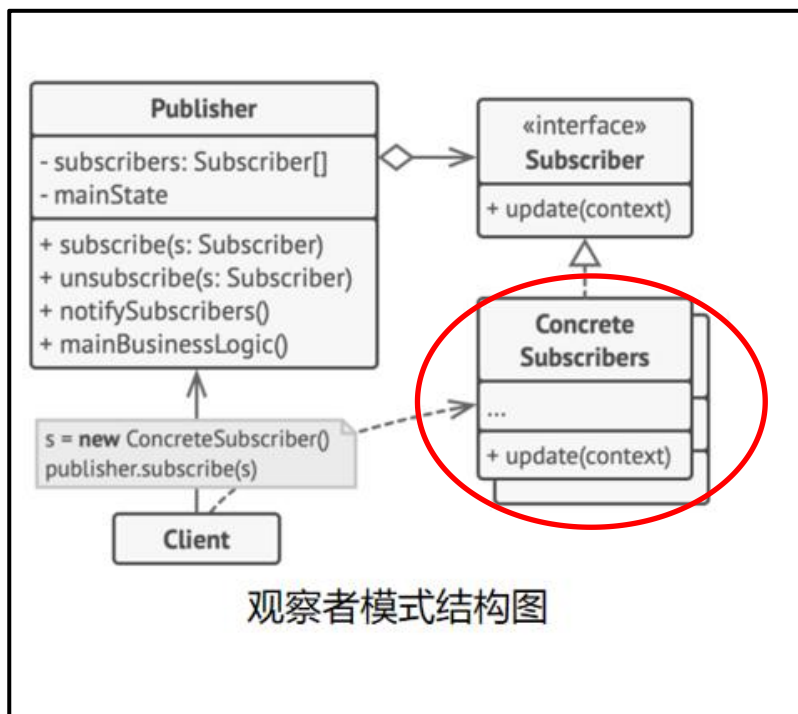


实验步骤

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

③ 创建实现订阅者接口的实体类；



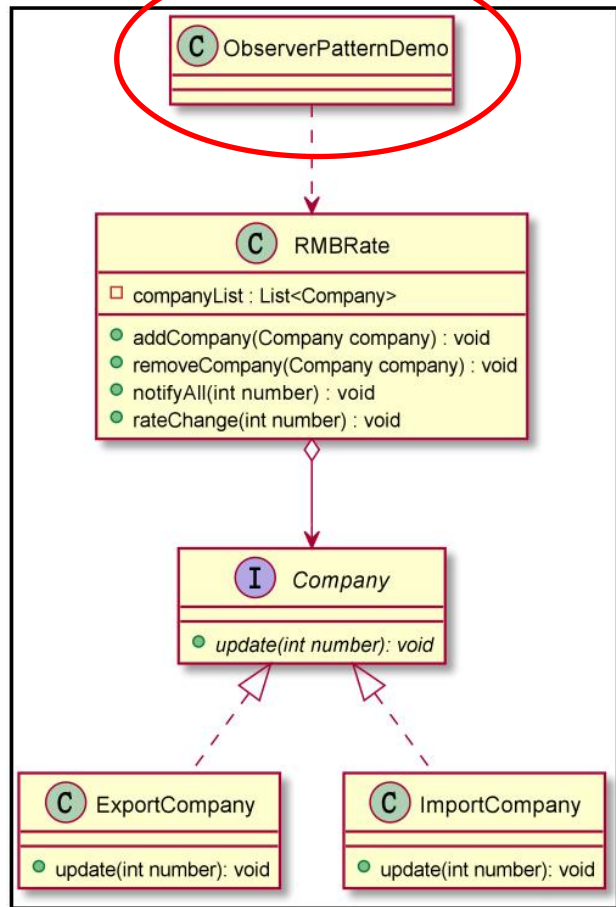


实验步骤

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

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





实验步骤

2 绘制观察者模式类图



请思考:

- 炸弹爆炸这个功能，谁是发布者？
- 订阅者清单，存的是什么对象？
- update() 方法里面要做什么动作？
- 谁是Client？





实验步骤

3 重构代码，实现观察者模式

根据你所设计的UML类图，重构代码，采用**观察者模式**实现**炸弹**道具；



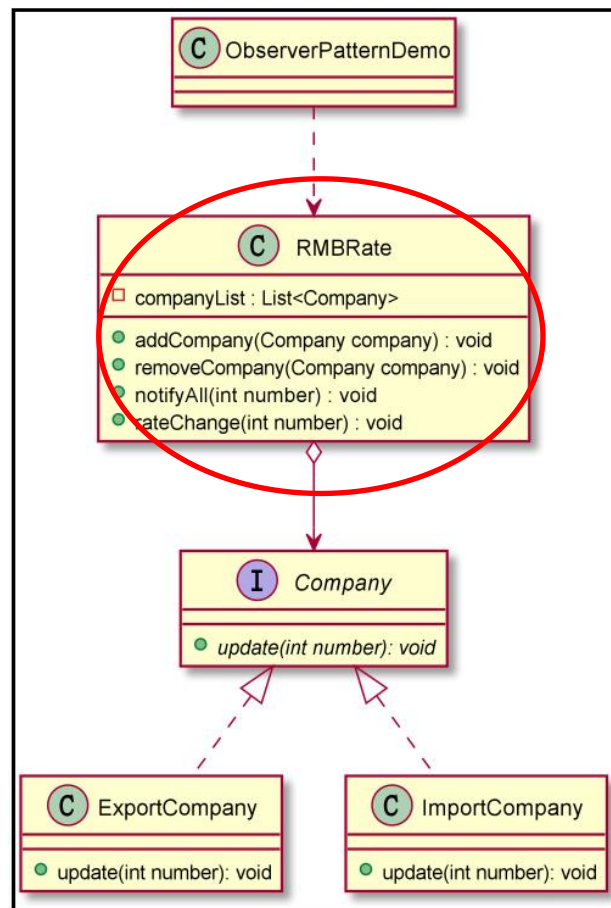
3 重构代码，实现观察者模式

● 观察者模式代码示例（人民币汇率）

① 创建 RMBRate 类，充当发布者角色。

```
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



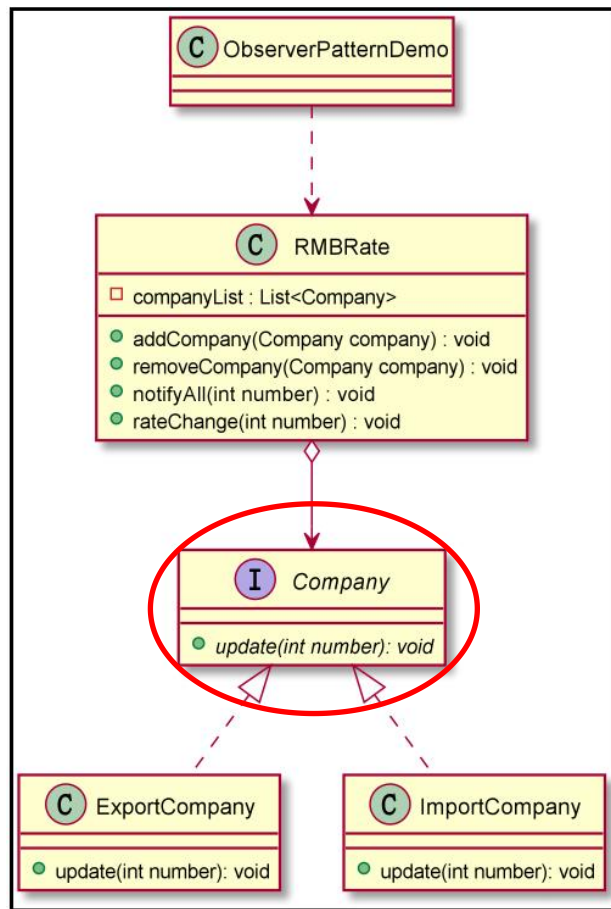
3 重构代码，实现观察者模式

● 观察者模式代码示例（人民币汇率）

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

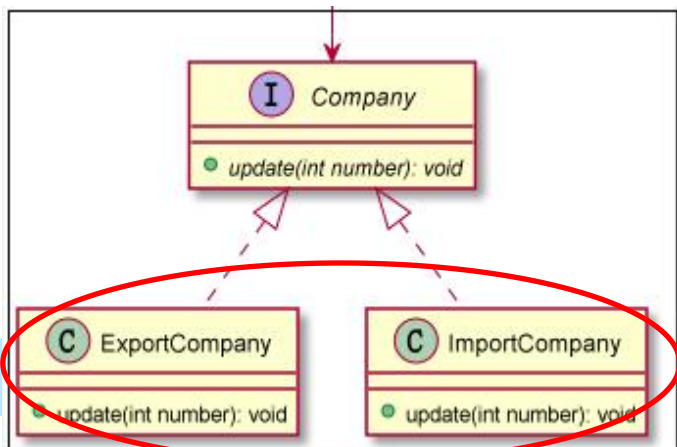


实验步骤

3 重构代码，实现观察者模式

- 观察者模式代码示例（人民币汇率）

③ 创建Company接口实体类，充当具体订阅者角色。



```
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，公司销售利润提升。");
        }
    }
}
```

具体订阅者1

```
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2

实验步骤

3 重构代码，实现观察者模式

● 观察者模式代码示例（人民币汇率）

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。

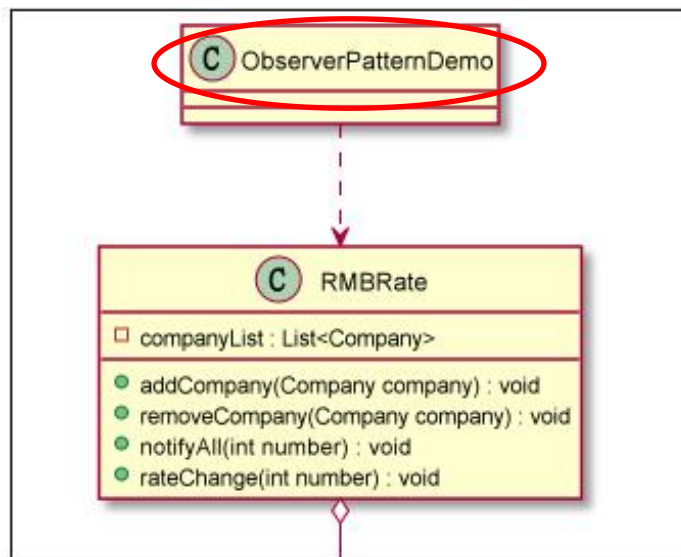
人民币汇率改变：

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，公司销售利润提升。

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);

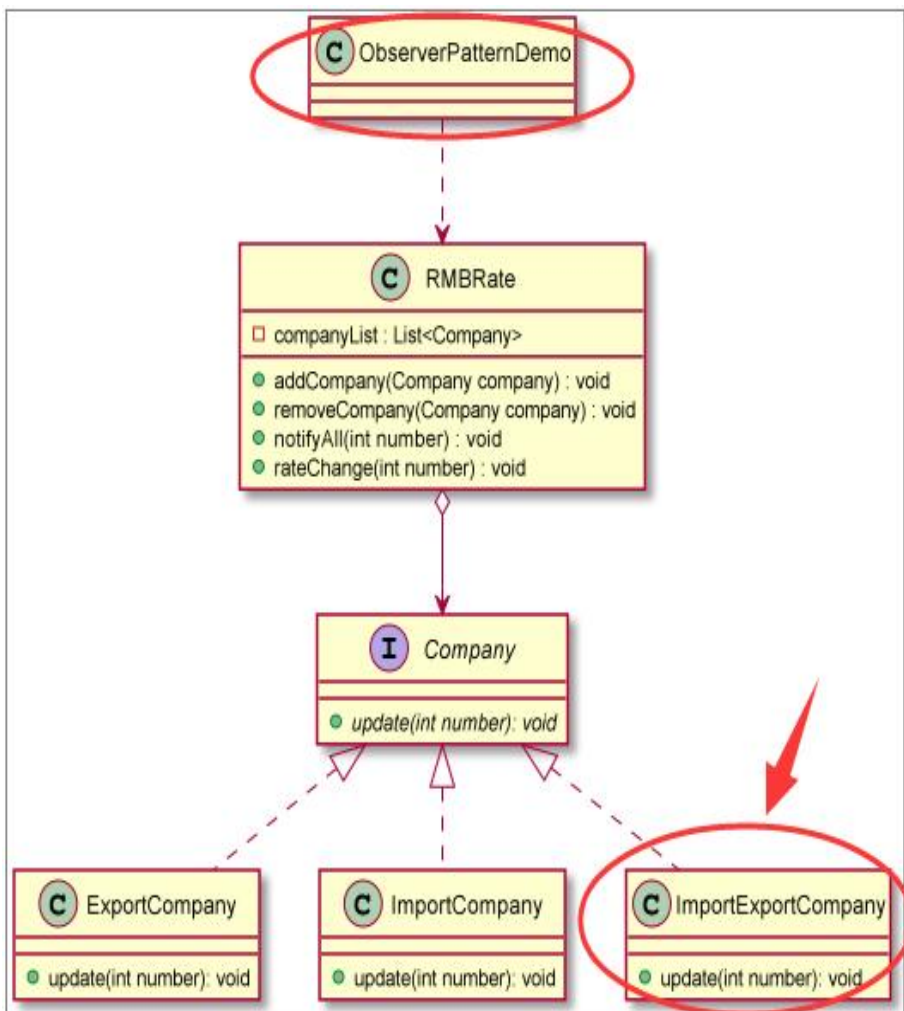
        rate.removeCompany(company1);

        System.out.println("人民币汇率改变：");
        rate.rateChange(8);
    }
}
```



实验步骤

请思考：如何添加一个进出口公司？



```
public class ObserverPatternDemo {
    public static void main(String[] args) {

        RMBRate rate = new RMBRate();
        Company company1 = new ImportCompany();
        Company company2 = new ExportCompany();

        Company company3 = new ImportExportCompany();

        rate.addCompany(company1);
        rate.addCompany(company2);

        rate.addCompany(company3);

        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);

    }
}
```

开闭原则



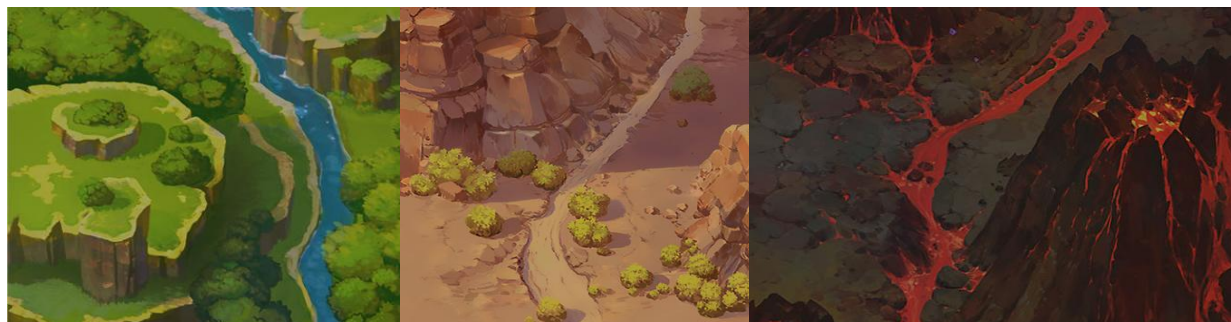
实验步骤

4

难度选择应用场景分析

应用场景
分析

用户进入游戏界面后，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
- Boss敌机每次出现的血量
- ...

★ 基本要求 (必须)	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



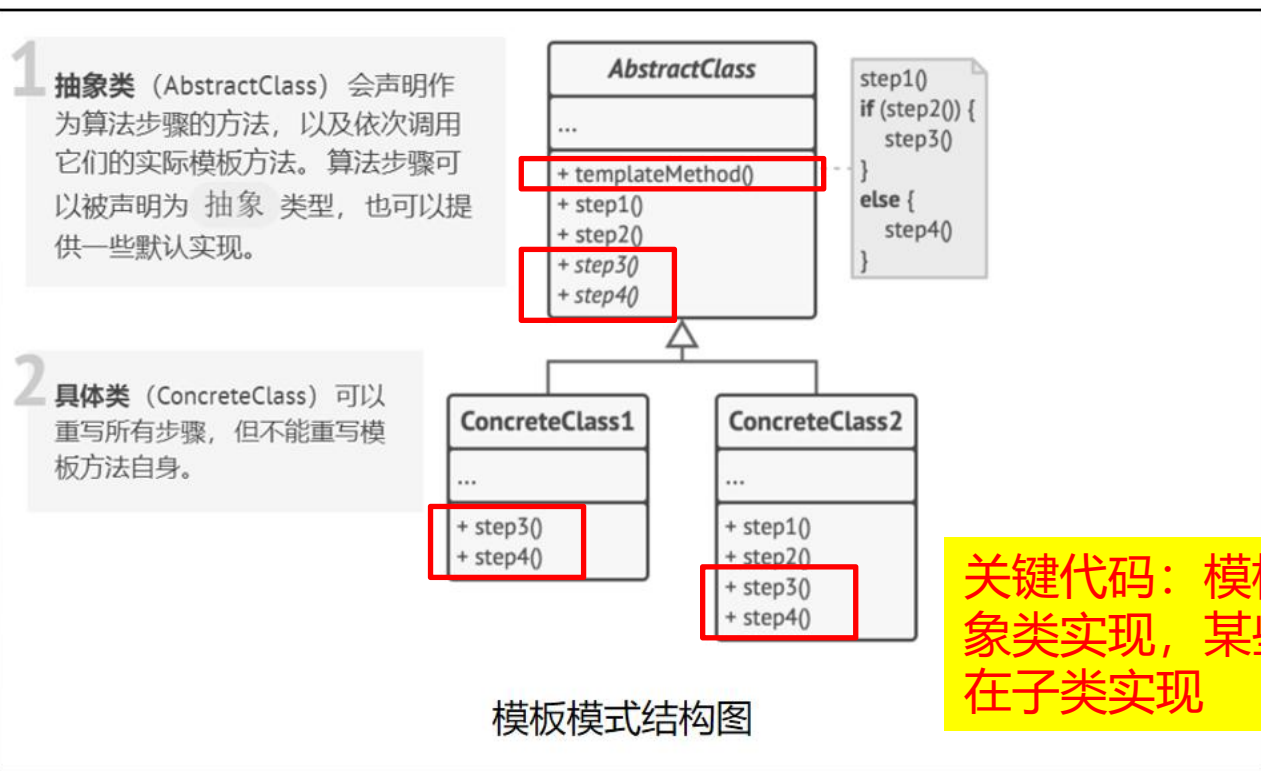
4 难度选择应用场景分析

请思考：

1. 三种游戏难度有哪些共性的地方？ 哪些不同的地方？
2. 若要增加一种新的游戏难度，需要改动哪些地方？
3. 如何实现代码复用？

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。





实验步骤

5

绘制模板模式类图

假如我们要去**银行办理业务**，要经过取号排队、办理业务、对银行工作人员进行评分三个步骤。我们该如何绘制UML类图？





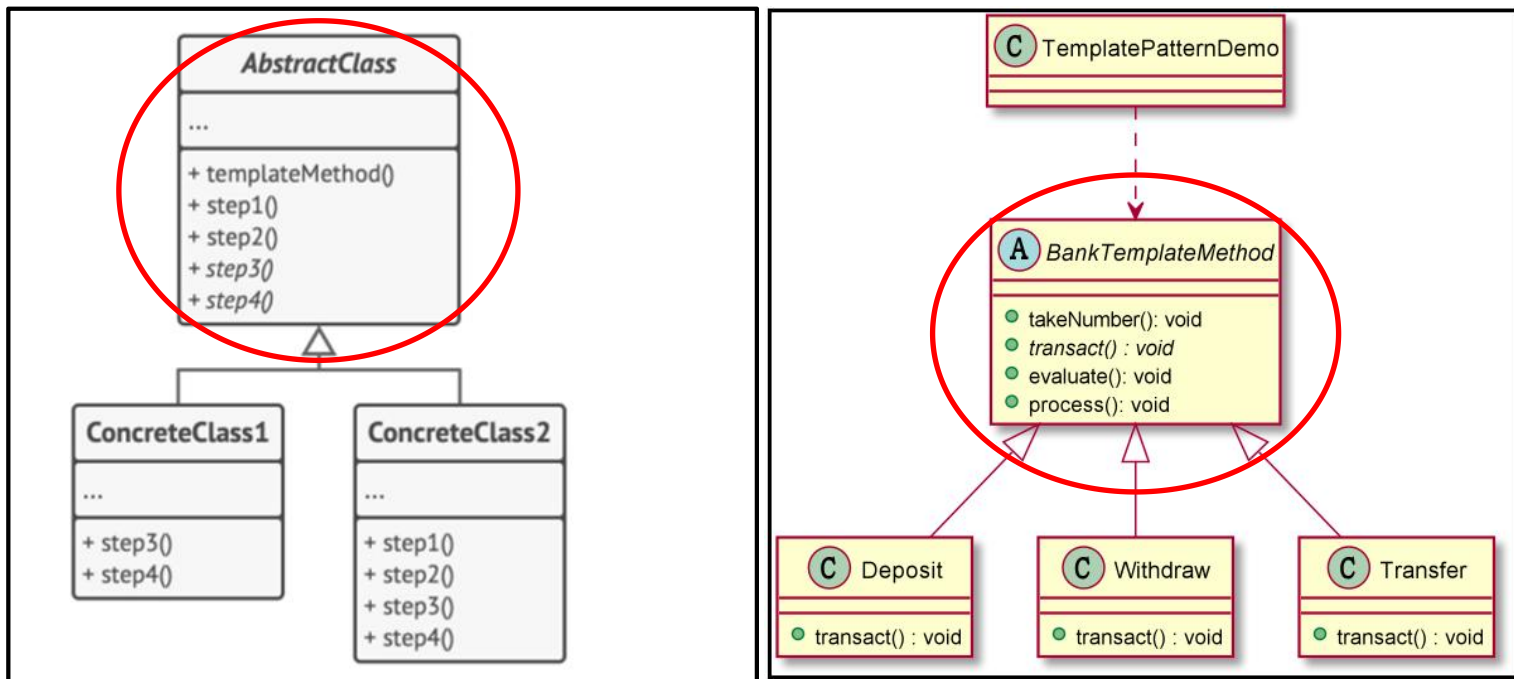
实验步骤

5

绘制模板模式类图

举个例子：银行业务办理

① 创建一个定义操作的BankTemplateMethod 抽象类及模板方法 process()。





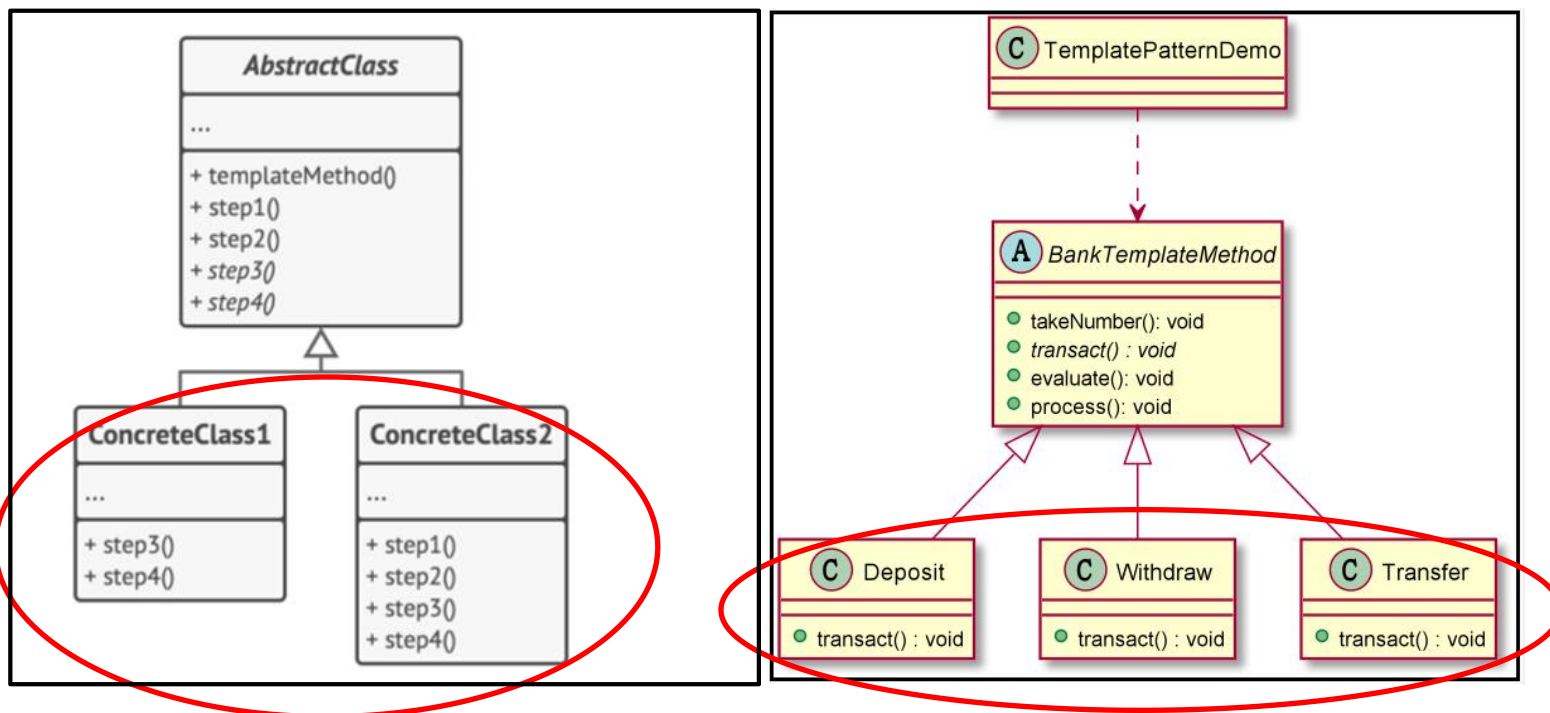
实验步骤

5

绘制模板模式类图

举个例子：银行业务办理

② **Deposit**、**Withdraw** 和 **Transfer** 是扩展了该抽象类的实体类，它们重写了抽象类的某些方法。





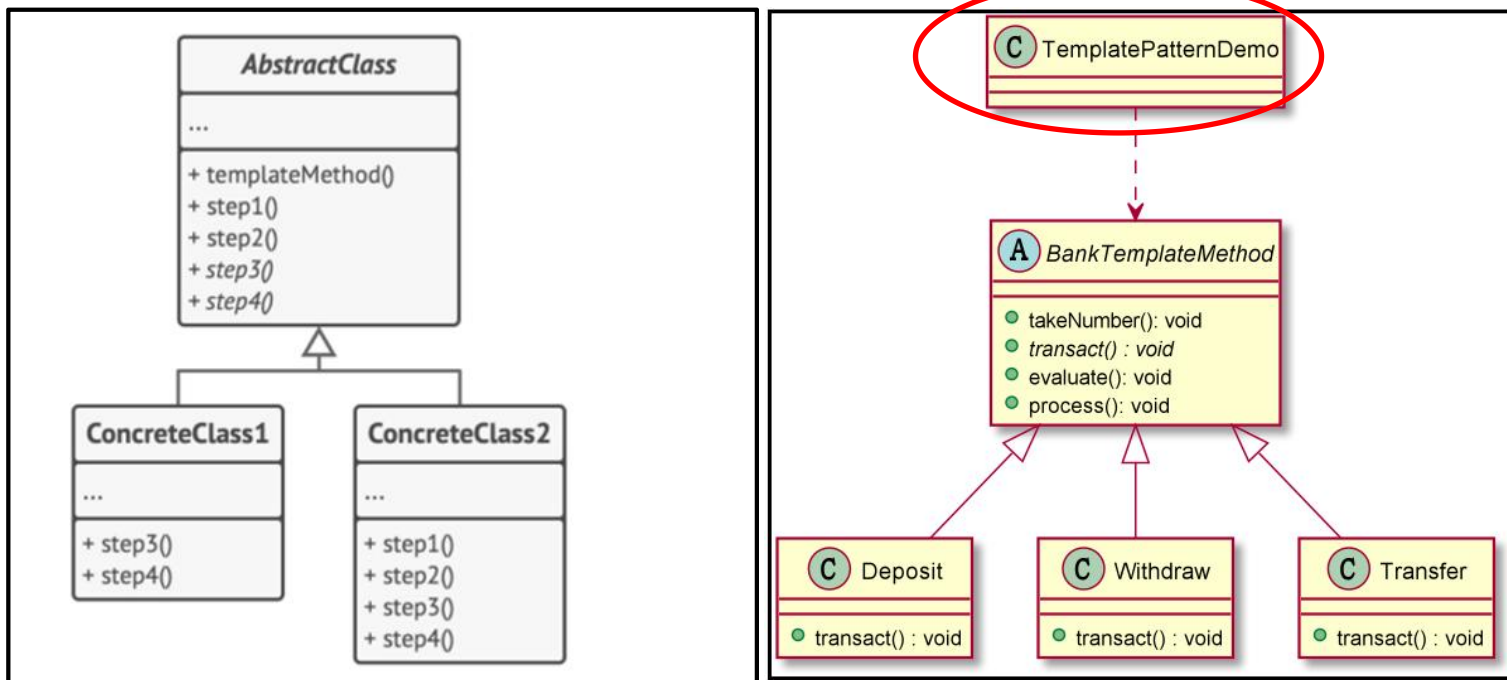
实验步骤

5

绘制模板模式类图

举个例子：银行业务办理

③ **TemplatePatternDemo**用于演示模板模式的使用。





实验步骤

6 重构代码，实现模板模式

根据你所设计的UML类图，重构代码，采用**模板模式**实现简单、普通、困难**三种游戏难度**。



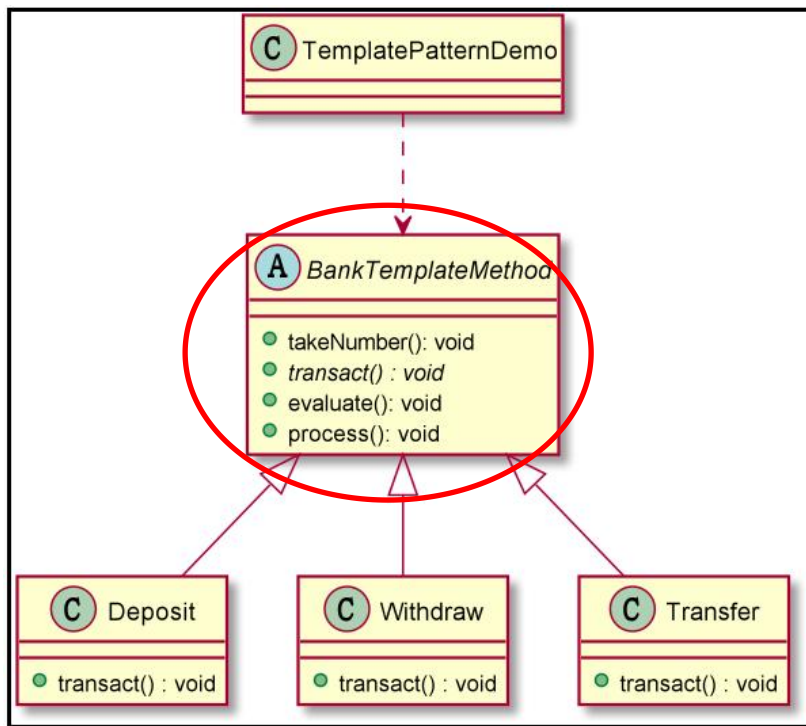


实验步骤

6 重构代码，实现模板模式

- 模板模式代码示例（银行业务办理）

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 ();
    }

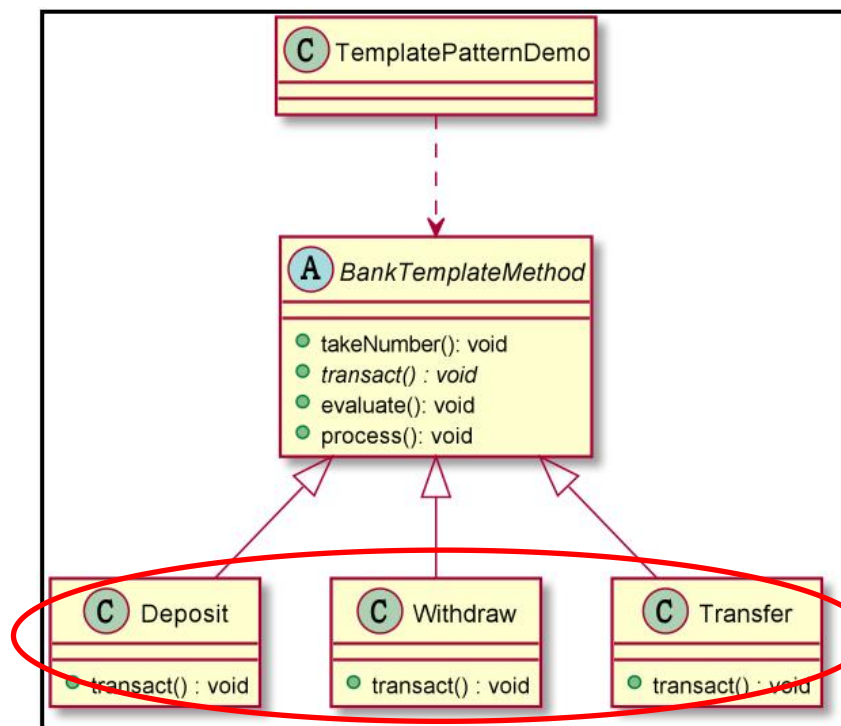
}
```

模板方法

6 重构代码，实现模板模式

- 模板模式代码示例（银行业务办理）

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 {

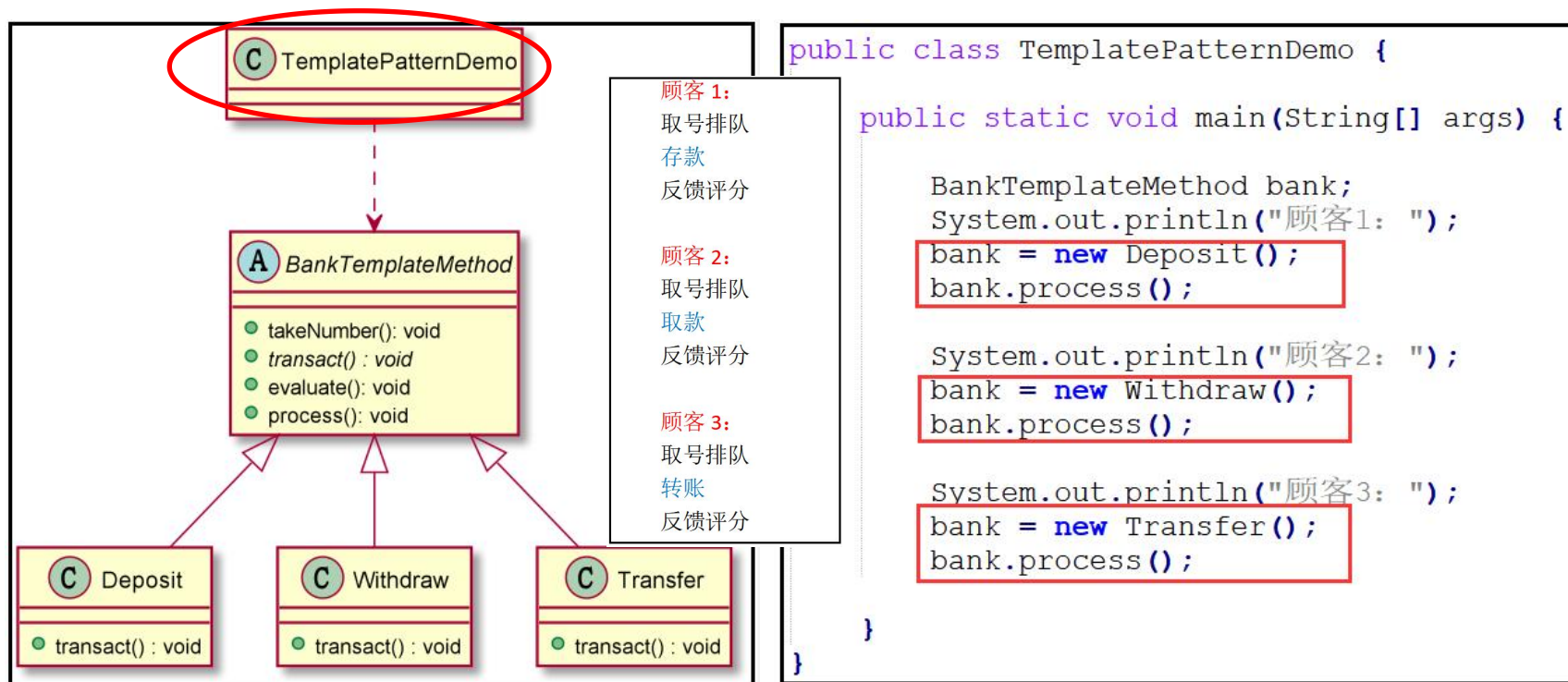
    @Override
    public void transact() {
        System.out.println("取款");
    }

}
```

6 重构代码，实现模板模式

● 模板模式代码示例（银行业务办理）

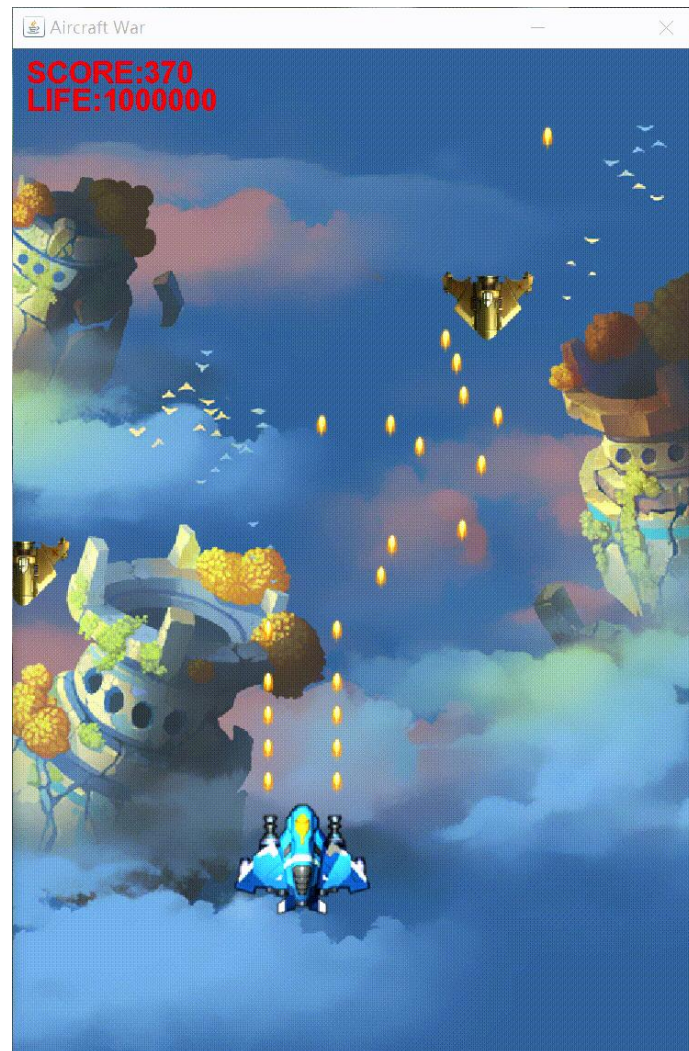
③ 使用BankTemplateMethod 的模板方法 process() 来演示模板模式。





本次实验的目标

- ✓ 使用**观察者模式**实现炸弹道具;
- ✓ **炸弹**生效时清除界面上除boss机外的所有敌机和敌机子弹, **Boss敌机血量减少**;
- ✓ 英雄机可获得坠毁的敌机分数。
- ✓ 使用**模板模式**实现**三种游戏难度**。
- ✓ 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。





本次实验的目标

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.

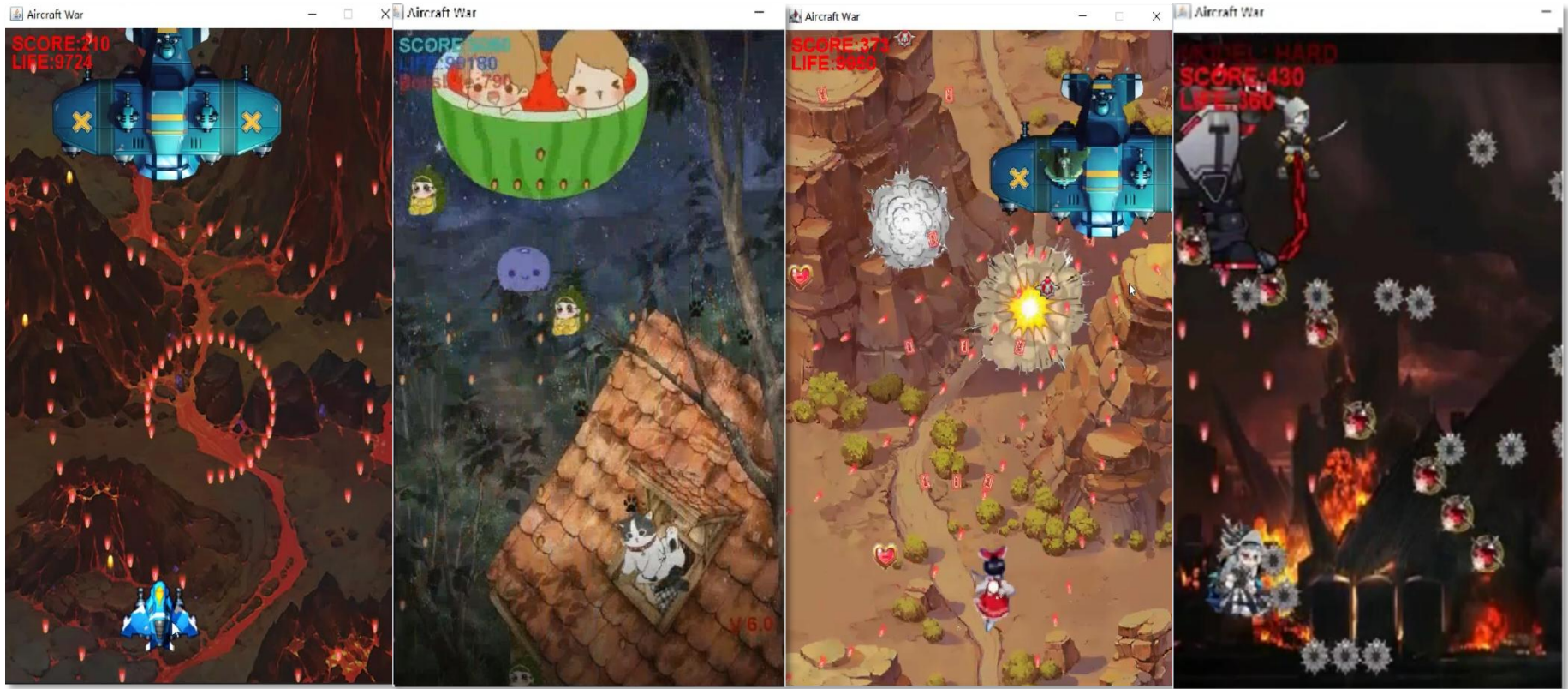
Aircraft War

SCORE:1040
LIFE:840

随游戏时长增加提升难度



作品展示





作业提交

• 提交内容

- ① 整个项目压缩包（整个项目压缩成zip包提交，包含代码、uml图等）
- ② 录制一段游戏的视频（小于2min），展示你的游戏的所有功能点和亮点；
- ③ 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。

登录网址：：<http://grader.tery.top:8000/#/login>



实际上，唯一不变的是变化！！！！

**同学们
请开始实验吧！**