



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

規格嚴格



功夫到家

面向对象的软件构造导论

实验一：飞机大战功能分析

2023春

哈尔滨工业大学（深圳）



各班课程表

➤ 3、4班上课时间

3周	星期5	7-8节
4周	星期5	7-8节
5周	星期5	7-8节
7周	星期1	5-6节
8周	星期2	7-8节
8周	星期5	7-8节
9周	星期5	5-8节

➤ 7、8班上课时间

3周	星期5	5-6节
4周	星期5	5-6节
5周	星期5	5-6节
7周	星期4	1-2节
8周	星期2	9-10节
8周	星期5	5-6节
9周	星期4	5-8节

➤ 地点：T2608

***如有课程冲突需要调整的请与我联系**



各班课程表

➤ 1、2班上课时间

3周	星期3	3-4节
4周	星期3	3-4节
5周	星期3	5-6节
7周	星期2	7-8节
8周	星期3	3-4节
8周	星期5	1-2节
9周	星期4	1-4节

➤ 5、6班上课时间

3周	星期4	5-6节
4周	星期4	5-6节
5周	星期4	5-6节
7周	星期1	3-4节
8周	星期1	3-4节
8周	星期4	5-6节
9周	星期5	9-12节

➤ 地点：T2608

***如有课程冲突需要调整的请与我联系**



目录

01 本学期实验总体安排

02 实验一说明

03 作业提交



目录

01

本学期实验总体安排

02

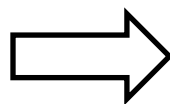
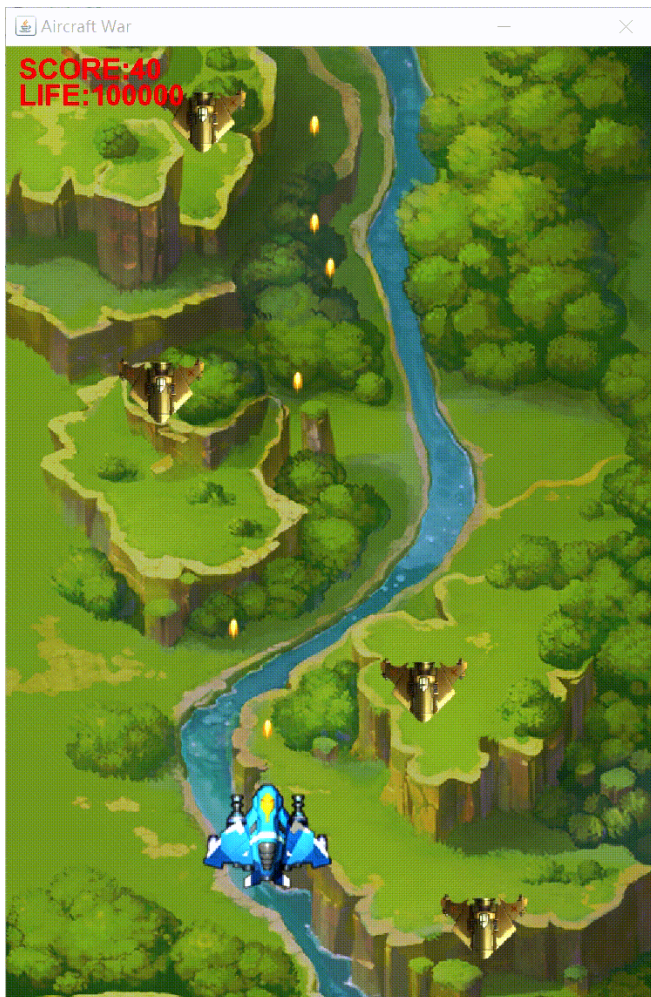
实验一说明

03

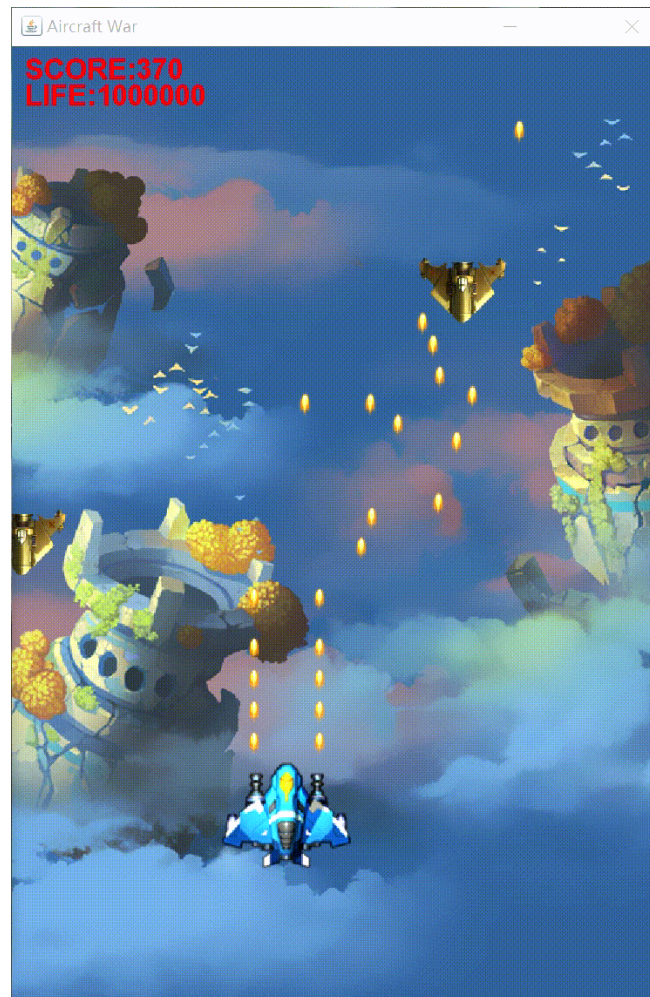
作业提交

本学期实验总体安排

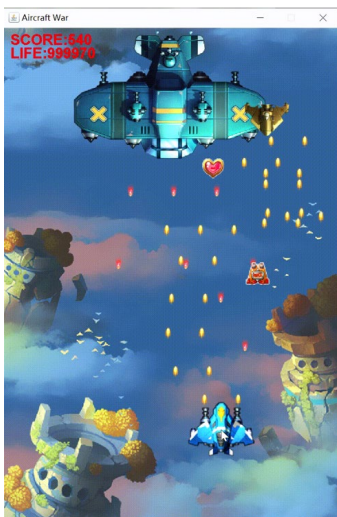
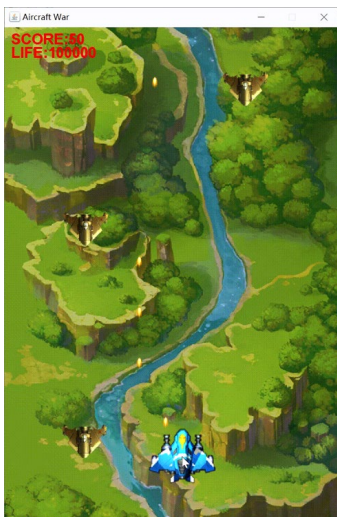
我们的
开始



我们的
目标



本学期实验总体安排



游戏主界面

英雄机移动
英雄机子弹直射
碰撞检测
统计得分和生命值

使用**单例模式**创建英雄机
使用**工厂模式**创建敌机和道具

使用**策略模式**实现不同弹道发射
使用**数据访问对象模式**实现得分排行榜

采用**观察者模式**实现炸弹道具生效
采用**模板模式**实现三种游戏难度

初始版本

01

创建**精英敌机**，精英敌机子弹**直射**
精英敌机坠毁后随机掉落三种**道具**
加血道具生效

02

03

添加**JUnit单元测试**
创建**Boss敌机**，Boss敌机子弹**散射**
Boss敌机坠毁后随机掉落三种**道具**

04

05

使用**Swing**实现游戏难度选择界面和排行榜界面
使用**多线程**实现音效的开启/关闭
使用**多线程**实现火力道具

06



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	模板模式 观察者模式
分数	4	6	4	6	6	14 (6+8)
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

本学期实验总体安排

02

实验一说明

03

作业提交



实验目的

- 深入理解面向对象的基本思想；
- 结合实例，深入理解面向对象分析和设计的方法；
- 掌握UML类图的绘制方法。



实验任务

1. 分析飞机大战系统功能;
2. 导入飞机大战模板程序;
3. 分析并设计所有的敌机类、道具类和子弹类, 并使用PlantUML插件绘制相应的类图及继承关系;
4. 在模板程序基础上, 为游戏增加精英敌机和三种道具。

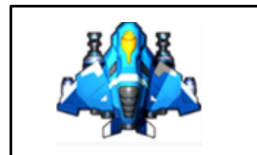


实验步骤

1 飞机大战系统功能分析

角色设定

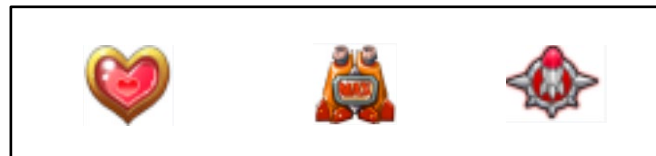
● 英雄机



● 敌机



● 道具



● 子弹





实验步骤

1 飞机大战系统功能分析



游戏规则

- 游戏难度：简单、普通、困难
- 子弹发射、弹道变化
- 道具生成、获取和使用
- 生命值和得分计算、总分排行榜



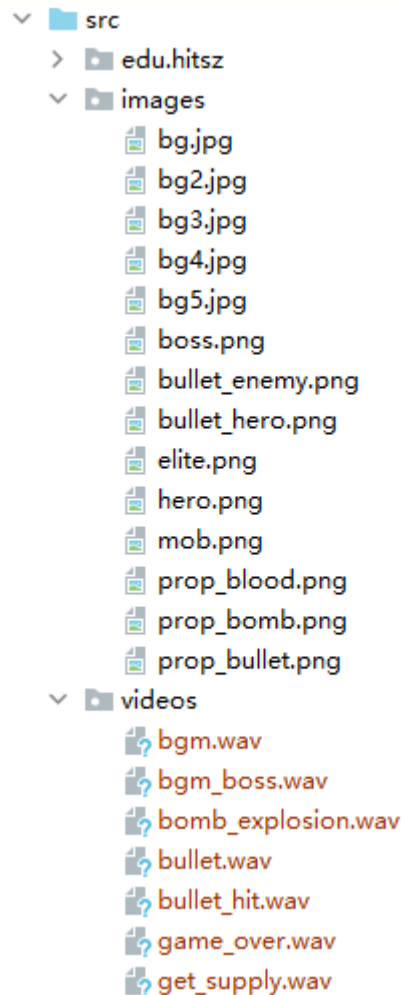
实验步骤

1 飞机大战系统功能分析

界面和音效

游戏地图

音效开启和关闭



需求细则请参考实验指导书!



实验步骤

2

模板程序导入

本实验提供了飞机大战的模板代码 **AircraftWar-base.zip**，已实现如下功能：

- ☐ 游戏主界面
- ☐ 英雄机、普通敌机移动
- ☐ 英雄机子弹直射
- ☐ 碰撞检测
- ☐ 统计并显示游戏得分和英雄机生命值

安装Java集成开发环境**IntelliJ IDEA**后可导入该项目运行。



实验步骤

2

模板程序导入

- 申请JetBrains免费个人许可证
- 安装并激活IntelliJ IDEA Ultimate
- 安装JDK11
- 导入项目

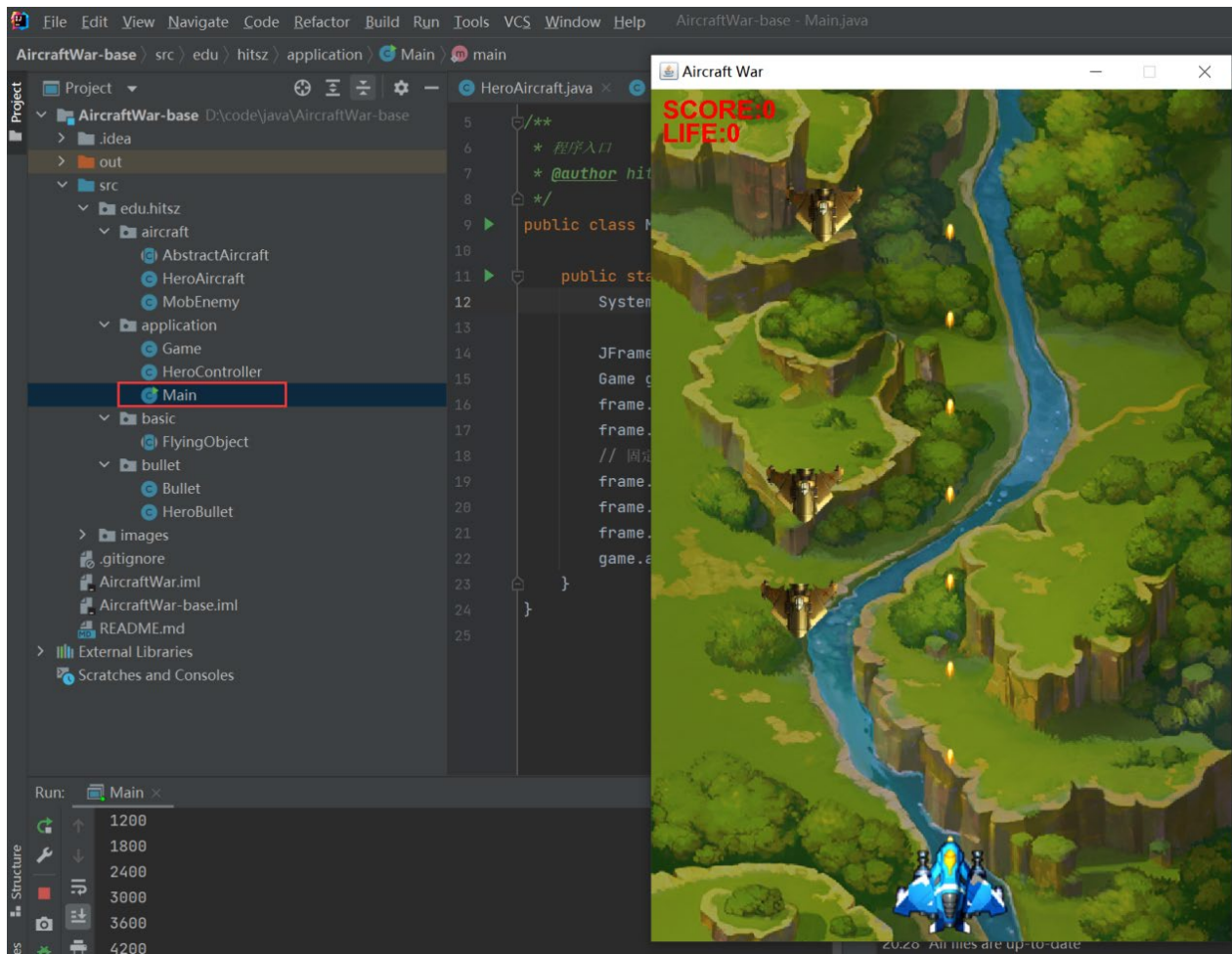


实验步骤

2

模板程序导入

- 申请JetBrains免费个人许可证
- 安装并激活IntelliJ IDEA Ultimate
- 安装JDK11
- 导入项目
- 运行Main.java



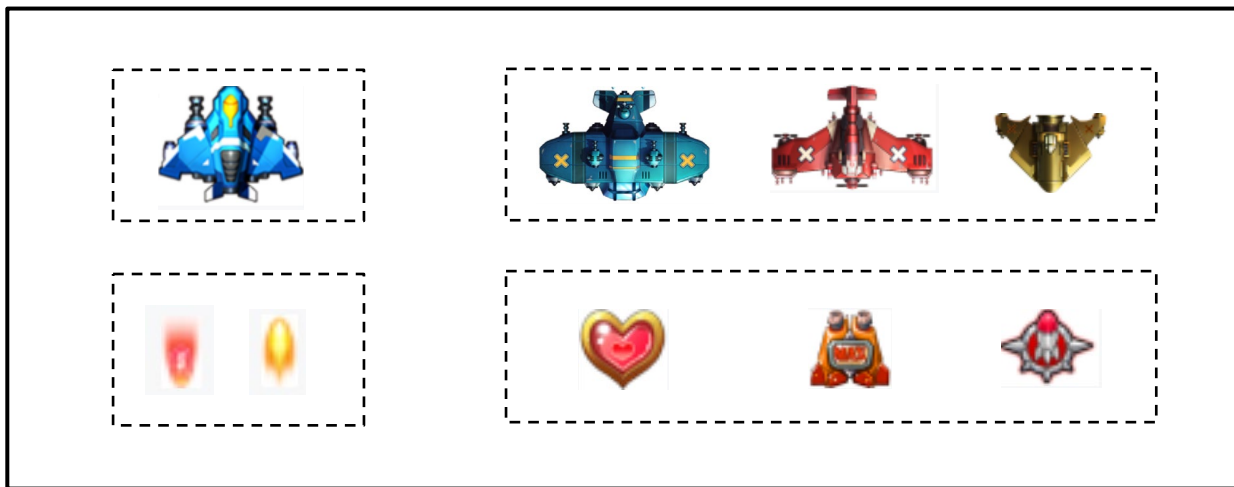


实验步骤

3

绘制UML类图

请根据面向对象设计原则，分析和设计游戏中的所有敌机类、道具类和子弹类，并使用PlantUML插件绘制相应的UML类图及继承关系，类图中需包括英雄机、所有敌机、道具、子弹及它们所继承的父类。



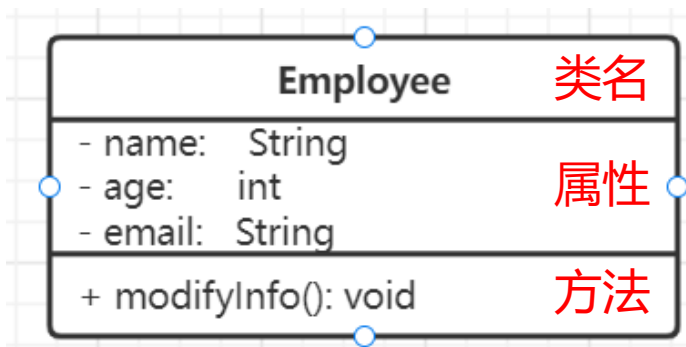


实验步骤

3

绘制UML类图

类图 (Class Diagram)：用来显示系统中的类、接口以及它们之间的静态结构和关系的一种**静态模型**。



```
public class Employee {  
    private String name;  
    private int age;  
    private String email;  
  
    public void modifyInfo() {  
        .....  
    }  
}
```



实验步骤

3

绘制UML类图

类与类之间的关系: 关联、泛化、实现和依赖。
其中关联又分为一般关联和聚合关系，组合关系。

01 ↑

泛化关系

继承关系，子类继承父类的所有行为和属性。如：老虎和动物

02 ↑

实现关系

类与接口的关系，表示类是接口所有特征和行为的实现者。如：鸟和飞行。

03 ↓

依赖关系

一种**使用关系**，一个类的实现需要其他类的协助。如：驾驶员和汽车。

04 ↓

一般关联

对象之间的一种**引用关系**，用于表示一类对象与另一类对象之间的联系。如：老师和学生。

05 ↓

聚合关系

整体与部分的关系，且部分可以离开整体而单独存在。如：汽车和轮胎。

06 ↓

组合关系

整体与部分的关系，但部分不能离开整体而单独存在。如：公司和部门。

实验步骤

3

绘制UML类图

The screenshot shows the IntelliJ IDEA IDE with the project 'AircraftWar-base304 - AbstractAircraft' open. The 'Project' tool window on the left displays the project structure, including the 'aircraft' package. A context menu is open over the 'aircraft' package, with the 'Diagrams' option highlighted. The 'Show Diagram...' option is selected, which will generate a UML class diagram for the selected package.

Below the context menu, a partial UML class diagram is visible, showing the following classes and their attributes/operations:

- FlyingObject**:
 - Attributes: locationX (int), locationY (int), speedX (int), speedY (int), image (BufferedImage), width (int), height (int), isValid (boolean).
 - Operations: FlyingObject(), FlyingObject(int, int, int), forward(), crash(FlyingObject) (boolean), getLocationX() (int), getLocationY() (int), setSpeedY() (int), getImage() (BufferedImage), getWidth() (int), getHeight() (int), notValid() (boolean), vanish() (void).
- AbstractAircraft**:
 - Attributes: maxHp (int), hp (int).
 - Operations: AbstractAircraft(int, int, int, int), decreaseHp(int) (void), getHp() (int), shoot() List<AbstractBullet>.
- HeroAircraft**:
 - Attributes: shootNum (int), power (int), direction (int).
 - Operations: HeroAircraft(int, int, int, int), forward() (void), shoot() List<AbstractBullet>.
- MobEnemy**:
 - Operations: MobEnemy(int, int, int, int), forward() (void), shoot() List<AbstractBullet>.

The diagram shows a generalization relationship (solid line with an open arrow) from HeroAircraft and MobEnemy to AbstractAircraft. A red circle highlights the 'Show Diagram...' option in the context menu.

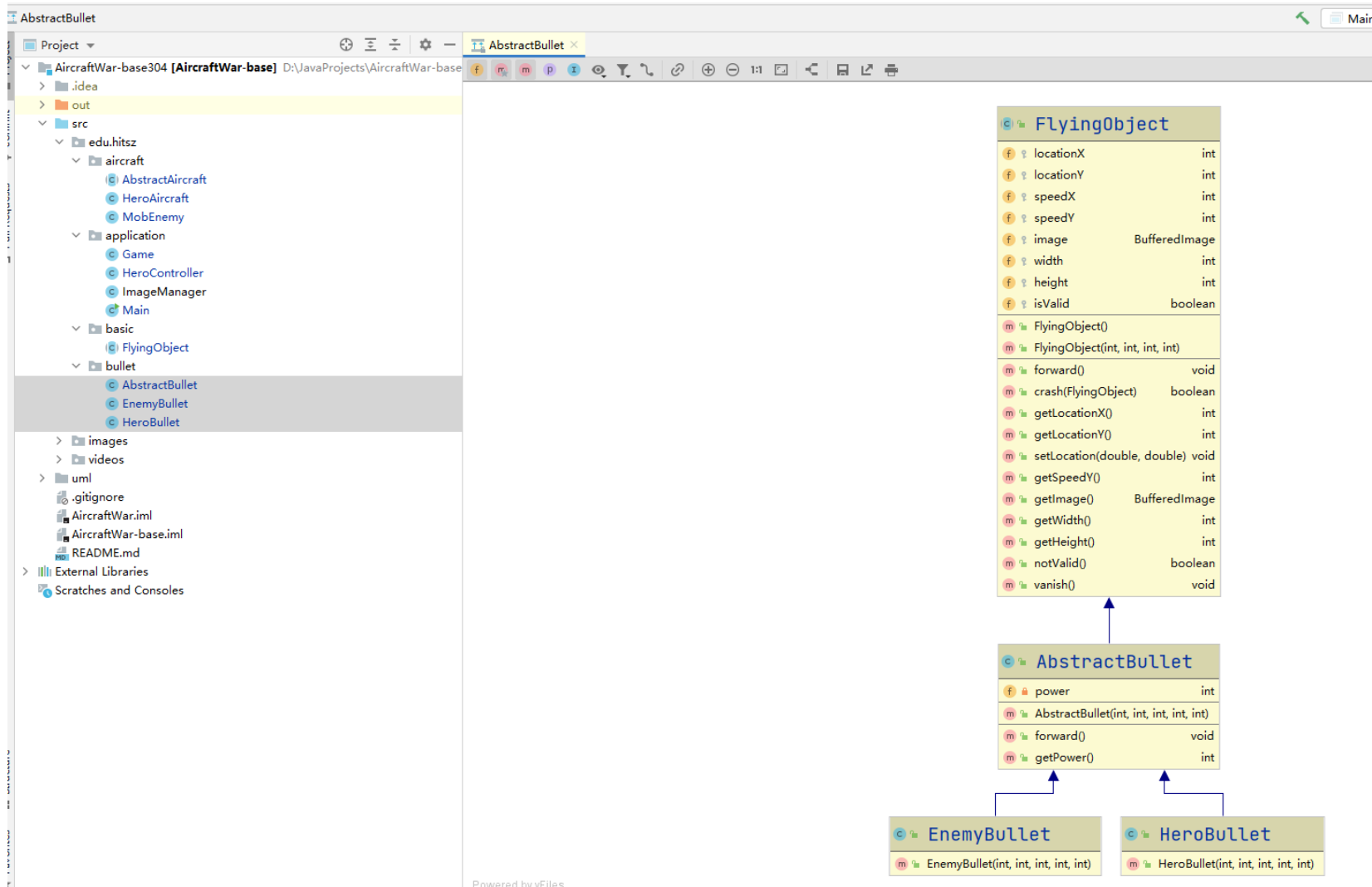
这是什么关系?

- A、泛化
- B、实现
- C、依赖
- D、关联

实验步骤

3

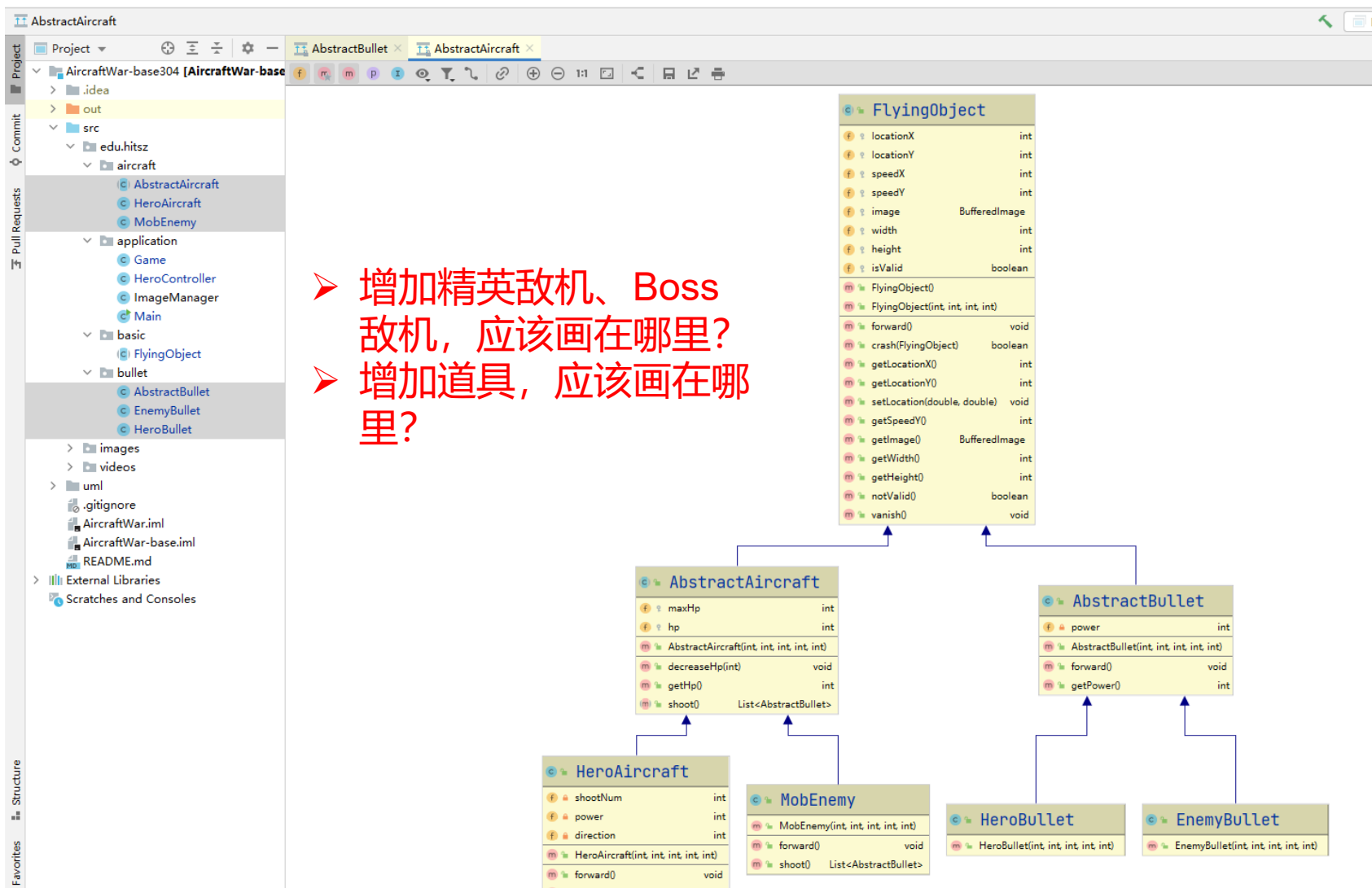
绘制UML类图



实验步骤

3

绘制UML类图



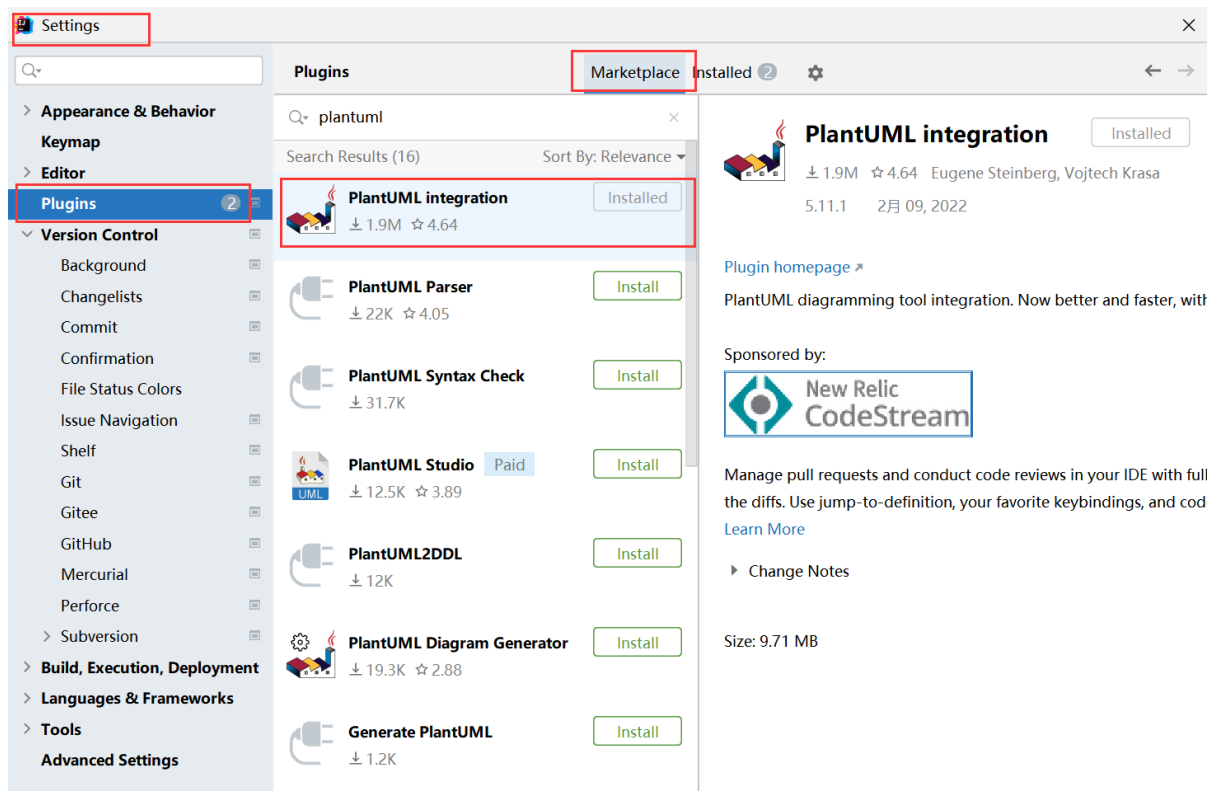


实验步骤

3

绘制UML类图

PlantUML是一个开源项目，支持快速绘制**类图**、时序图、用例图、等UML结构图和行为图。



实验步骤

3

绘制UML类图

AircraftWar-base - Inheritance.puml

```
25 + notValid():boolean
26 + vanish():void
27 }
28 abstract class AbstractAircraft
29 {
30     # maxHp:int
31     # hp:int
32     + AbstractAircraft(int locationX, int locationY, int speedX, int speedY, int hp)
33     + decreaseHp(int decrease):void
34     + getHp():int
35     + {abstract} shoot():List<BaseBullet>
36 }
37
38
39 class HeroAircraft {
40     - shootNum:int
41     - power:int
42     - direction:int
43     + HeroAircraft(int locationX, int locationY, int speedX, int speedY, int hp)
44     + forward():void
45     + shoot():List<BaseBullet>
46 }
47
48 AbstractAircraft <|-- HeroAircraft
49
50 class MobEnemy {
51     + MobEnemy(int locationX, int locationY, int speedX, int speedY, int hp)
52     + forward():void
53     + shoot():List<BaseBullet>
54 }
55
56 AbstractAircraft <|-- MobEnemy
```

注意UML类图绘制的规范性：
类、抽象类、接口、成员变量、
方法、可见性、返回值、关系的
符号等

UML Class Diagram:

- AbstractAircraft** (Abstract Class)
 - Attributes: maxHp:int, hp:int
 - Operations: AbstractAircraft(int locationX, int locationY, int speedX, int speedY, int hp), decreaseHp(int decrease):void, getHp():int, shoot():List<BaseBullet>
- HeroAircraft** (Class)
 - Attributes: shootNum:int, power:int, direction:int
 - Operations: HeroAircraft(int locationX, int locationY, int speedX, int speedY, int hp), forward():void, shoot():List<BaseBullet>
- MobEnemy** (Class)
 - Operations: MobEnemy(int locationX, int locationY, int speedX, int speedY, int hp), forward():void, shoot():List<BaseBullet>

Relationships:

- HeroAircraft inherits from AbstractAircraft.
- MobEnemy inherits from AbstractAircraft.



实验步骤

3

绘制UML类图

PlantUML类图的语法和功能:

```
interface 接口

abstract class 抽象类

class 类
{
    + 公有成员变量: String
    - 私有成员变量: int
    # 受保护的成员变量: Date
    + {static} 静态成员变量: String

    + 构造函数(int 参数1)
    + 普通方法(int 参数1, String 参数2): void
    + {abstract} 抽象方法(): int
    + {static} 静态方法(int 参数1, String 参数2): String
}
```



请参考官网说明: [类图的语法和功能 \(plantuml.com\)](http://plantuml.com)



实验步骤

3

绘制UML类图

PlantUML类图的语法和功能:

```
abstract class 类A
class 类B
类A <|-- 类B
interface 类C
class 类D
类C <|.. 类D
class 类E
class 类F
类E ..> 类F
class 类G
class 类H
类G --> 类H
class 类I
class 类J
类I o--> 类J
class 类K
class 类L
类K *--> 类L
```

泛化

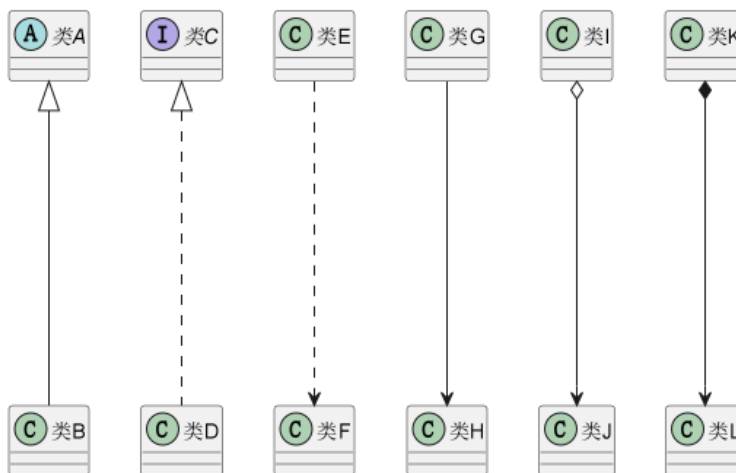
实现

依赖

一般关联

聚合

组合



请参考官网说明: [类图的语法和功能 \(plantuml.com\)](http://plantuml.com)



实验步骤

4

重构代码

请根据你所设计的UML类图，重构代码，为游戏添加**精英敌机**和**三种道具**。

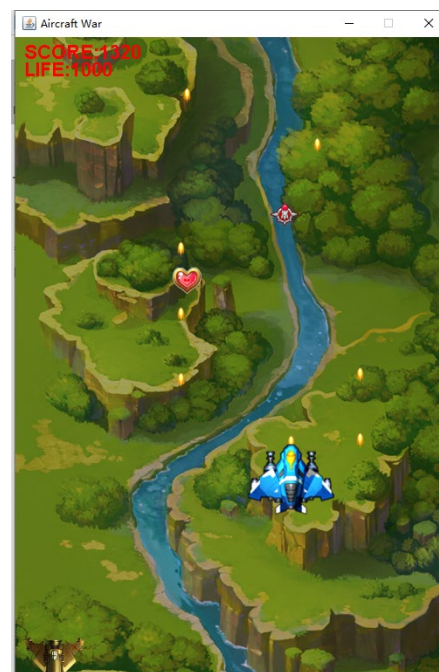
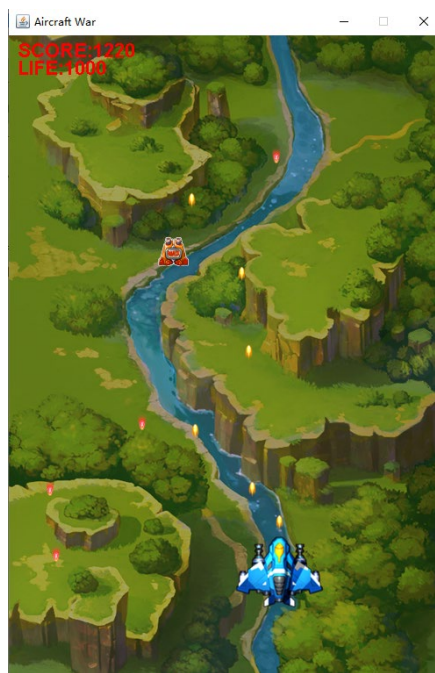
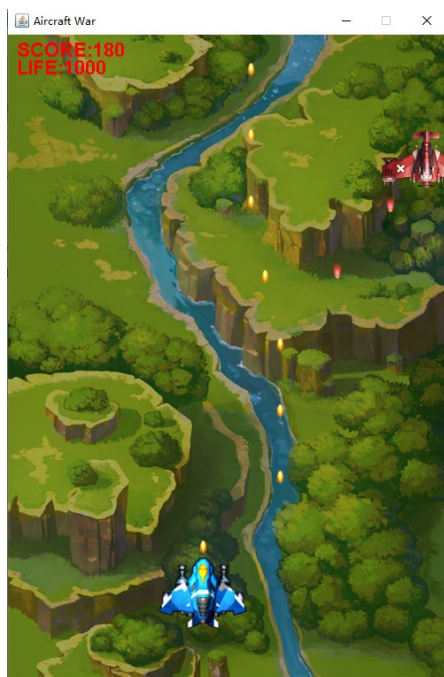
本实验UML图需包含Boss机，但代码暂不实现Boss机。





本次实验的目标（1）

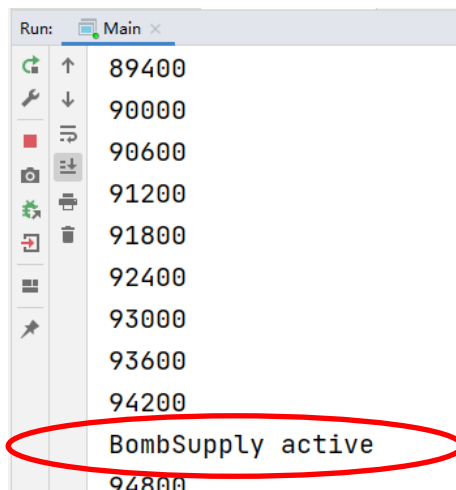
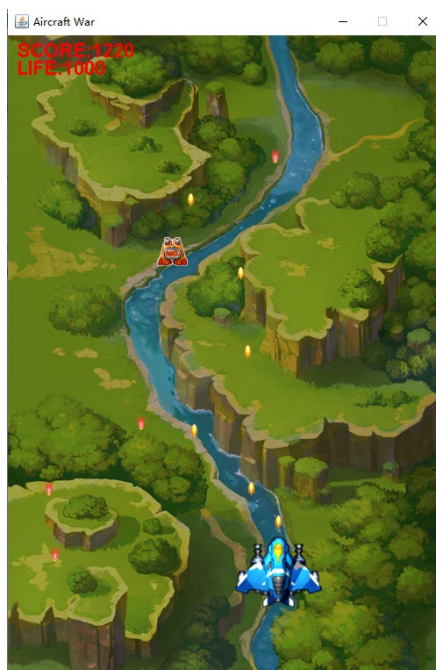
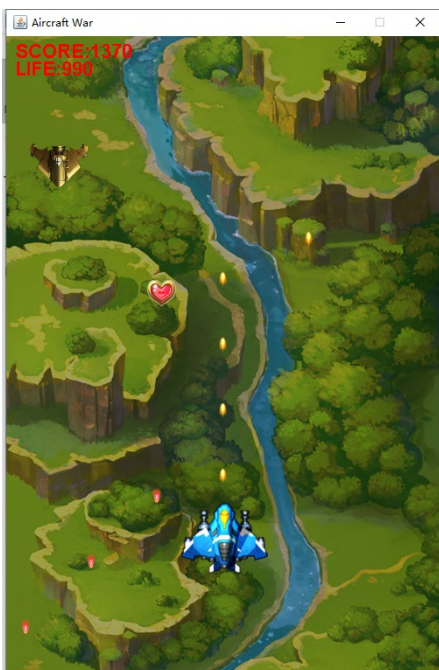
- ✓ 每隔一定周期**随机**产生一架**普通敌机**或**精英敌机**；
- ✓ **精英**敌机按设定周期**直射**子弹；
- ✓ **精英**敌机坠毁后**随机**产生某种向下飞行的**道具**（或不产生）；





本次实验的目标（2）

- ✓ ☐ 英雄机碰撞道具后，道具自动生效；
- ✓ 加血道具可使英雄机恢复一定血量，但不超过初始值；
- ✓ 火力道具和炸弹道具暂不实现具体功能，生效时只需在控制台打印“FireSupply active!” “BombSupply active!” 语句即可。





目录

01 本学期实验总体安排

02 实验一说明

03 作业提交



作业提交

- **提交内容：**项目代码（含UML类图）

包括：

- ① 使用PlantUML插件绘制的UML类图及继承关系；
- ② 正常运行的代码。

- **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>
- 推荐浏览器：Chrome
- 初始用户名、密码均为学号，登录后**请及时修改密码**

注意：上传后请自行下载确认是否正确提交。



小补充

➤ 几个有助于看懂框架代码的功能

1、 show diagram用代码生成UML图

2、 find usages找到所有使用某变量或方法的代码

3、 Ctrl+enter可以找到某个变量或者方法的出处

➤ 修改代码好用的功能Refactor



**同学们
请开始实验吧！**