哈尔滨工业大学（深圳）

面向对象的软件构造导论
实验指导书

实验一 飞机大战功能分析

2023 春

目录

1. 实验目的

1. 理解面向对象的基本思想
2. 结合实例，理解面向对象分析和设计的方法
3. 掌握 UML 类图的绘制方法

2. 实验环境

1. Windows 10
2. IntelliJ IDEA 2022.3.2
3. Java 11

3. 实验内容（2 学时）

1. 分析飞机大战系统功能;
2. 导入飞机大战模板程序;
3. 根据面向对象设计原则，设计普通和精英敌机类、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，重构代码，为系统增加精英敌机类和三种道具类，以及它们的父类。

4. 实验步骤

4.1 分析飞机大战系统功能

根据下发的飞机大战任务书，分析系统中的类和对象，初步确定系统的功能模块。

飞机大战任务书

1. 角色设定

名称：英雄机		预期应用设计模式
实例数量	**1**	**单例模式**
种类	**1 种：英雄机**	

移动方式	跟随玩家鼠标移动	
生存	通过生命值（血）生存，被敌机子弹击中损失部分生命值，被敌机碰撞则全部损失	
火力	自动发射子弹，如有火力道具加成则改变弹道	策略模式
消灭	生命值为 0 时判定游戏失败	

名称：敌机		预期应用设计模式
实例数量	不限	工厂模式
种类	3 种：普通敌机、精英敌机、Boss 敌机	
移动方式	普通敌机、精英敌机：以一定频率在界面随机位置出现并向屏幕下方移动	
	Boss 敌机悬浮于界面上方左右移动，直至被消灭	
生存	通过生命值生存，被英雄机子弹击中损失部分生命值，生命值为 0 时坠毁	
火力	普通敌机不发射子弹，精英敌机和 Boss 机按照一定轨迹、频率模式发射子弹	策略模式
消灭	敌机坠毁时，总分增加相应分数，随机掉落某种道具	

名称：道具		预期应用设计模式
实例数量	不限	工厂模式
种类	3 种：火力道具、炸弹道具、加血道具	
出现方式	敌机坠毁后，以较低概率随机出现某种道具	
使用方式	英雄机碰撞后自动触发	
移动方式	以一定速度向屏幕下方移动	
效果	火力道具：英雄机由直射切换为散射弹	

	道并持续一段时间，结束后恢复原状态	
	炸弹道具：清除界面上除 boss 机外的所有敌机和敌机子弹，英雄机可获得坠毁的所有敌机的分数。	观察者模式
	加血道具：可使英雄机恢复一定血量，但不能超过英雄机初始的最大血量	
消灭	与英雄机碰撞或移动至界面底部则消失	

名称：子弹		预期应用设计模式
实例数量	不限	
种类	2 种：英雄机子弹（黄色）、敌机子弹（红色）	
速度	子弹飞行速度	
角度	子弹初始飞行方向	
移动方式	以一定速度向某个方向匀速直线运动	
消灭	击中英雄机、敌机，或飞出界面外自动消亡	

名称：排行榜		预期应用设计模式
排名方式	根据游戏难度分类，按照总分排名	数据访问对象模式
记录信息	游戏难度、名次、玩家名、得分、记录时间	
更新方式	游戏结束后显示该难度的得分排行榜界面，排名正确，可要求记录或取消记录玩家该局得分，并可删除玩家历史得分（需有确认对话框）。	

2. 游戏规则

（1） 用户进入游戏界面后，可选择某种**游戏难度**：简单 / 普通 / 困难。其中，普通和困难模式在得分达到设定阈值后，Boss 机出现。

（2） 英雄机、精英机和 Boss 机采用**不同弹道**发射子弹，如 Boss 机可散射。

（3） 敌机或英雄机被**子弹击中**后（子弹消失），生命值减少，生命值为 0 时**坠毁**。

（4） 英雄机生命值为 0 或与任意敌机相撞，英雄机坠毁，游戏结束。

（5） 精英机和 Boss 机坠毁后会以较低概率**随机掉落某种道具**（也可不产生道具）；不同类型敌机坠毁后，英雄机增加相应的分数。英雄机碰撞道具后，道具自动生效。

（6） 普通和困难模式，每一局随着游戏时长增加，程序会逐渐增加游戏难度，如增加敌机血量、速度，增加精英机出现概率、缩短敌机产生周期等。并且得分每超过一次阈值，则产生一次 Boss 机，困难模式每次召唤提升 boss 机血量。

（7） 游戏过程中记录**英雄机得分**，在界面左上角显示。**游戏结束**后显示玩家当局分数，显示当前游戏难度的历史**得分排行榜**。并询问玩家是否保存本次记录。如保存，则插入新的游戏记录，需要玩家输入相关信息（如玩家姓名），玩家也可删除历史记录。

3. 界面和音效需求

（1） **游戏界面**：风格清新，简单易操作。

（2） **音效**：可打开或关闭

- ✧ 游戏背景音乐
- ✧ Boss 机出现
- ✧ 敌机被子弹击中
- ✧ 三种道具生效

若音效开启，游戏中循环播放游戏背景音乐，游戏结束后停止播放。子弹击中敌机、炸弹爆炸、道具生效、游戏结束时有相应的音效。Boss 敌机出场时循环播放其背景音乐，坠毁后或游戏结束后停止播放。

4.2 导入飞机大战模板程序

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

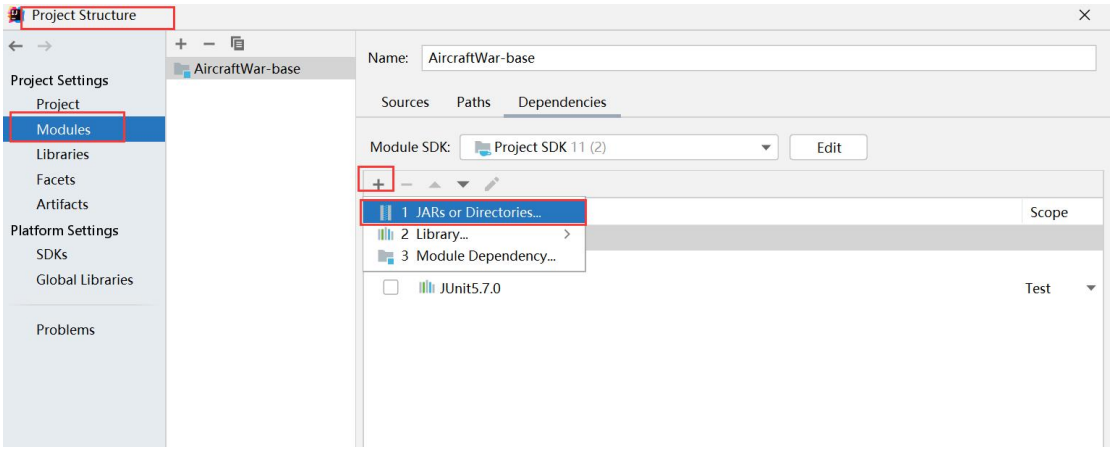
- ✓ 游戏主界面
- ✓ 英雄机、普通敌机移动
- ✓ 英雄机子弹发射

- ✓ 碰撞检测
- ✓ 统计并显示得分和英雄机生命值

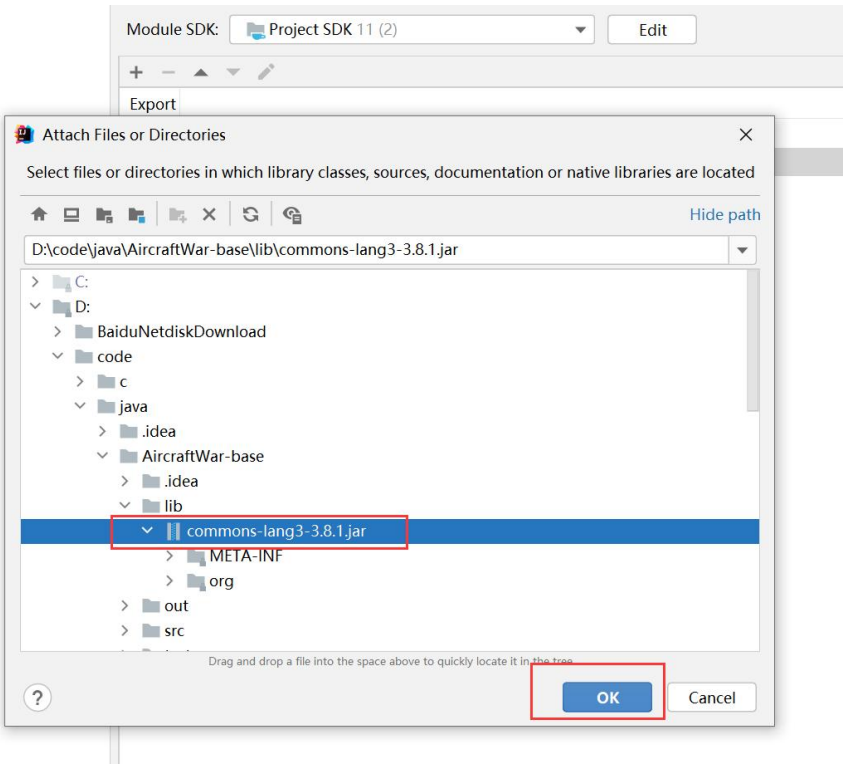
Java 集成开发环境 IntelliJ IDEA 的安装步骤请参考实验前下发的实验环境搭建说明文档。

安装后可导入模板程序 AircraftWar-base 项目运行：
添加 commons-lang3-3.8.1.jar，运行 Main.java，模板代码已完成基本运行框架，并已创建英雄机和普通敌机。

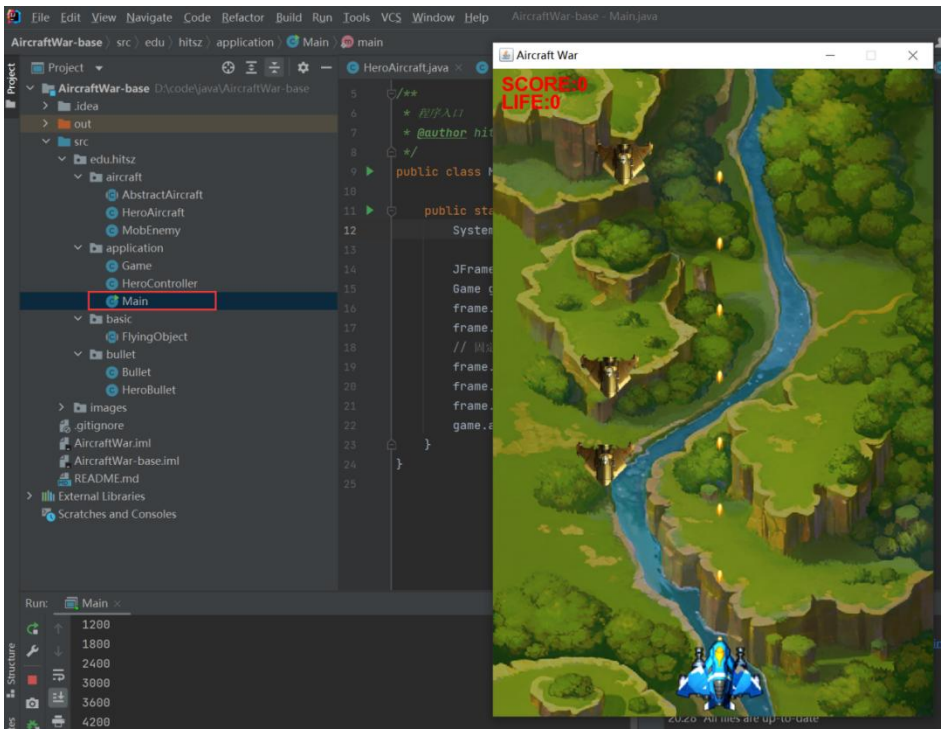
点击“File”，选择“Project Structure”->“Modules”-> “Dependencies”：



添加 commons-lang3-3.8.1.jar：



运行 src\edu\hitsz\application\Main.java:

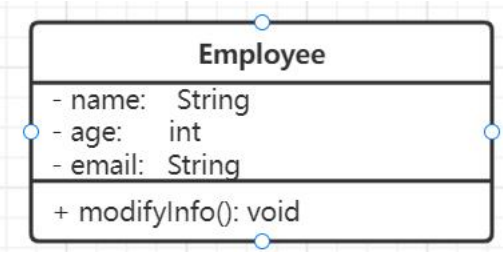


4.3 根据面向对象设计原则，设计普通和精英敌机类、道具类和子弹类

游戏中有 3 种类型的敌机：普通敌机、精英敌机、Boss 敌机，3 种类型的道具：火力道具、炸弹道具、加血道具，还有 2 种类型的子弹：英雄机子弹、敌机子弹。请根据面向对象设计原则，分析和设计游戏中的普通和精英敌机类（Boss 机暂不实现）、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、普通和精英敌机、三种道具、两种子弹及它们所继承的父类**。

4.3.1 UML 类图

类图（Class Diagram）：类图是面向对象系统建模中最常用和最重要的图，是定义其它图的基础。类图主要是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。在 UML 中，类使用包含类名、类的属性和操作且带有分隔线的长方形来表示，如定义一个 Employee 类，它包含属性 name、age 和 email，以及操作 modifyInfo()。



对应的 Java 代码片段如下：

```
public class Employee {
    private String name;
    private int age;
    private String email;

    public void modifyInfo() {
          ......
    }
}
```

4.3.2 类与类之间的关系

在画类图的时候，理清类和类之间的关系是重点。类的关系有关联(Association)、泛化(Generalization)、实现（Realization）和依赖(Dependency)。其中关联又分为一般关联关系和聚合关系(Aggregation)，组合关系(Composition)。

- **关联（Association）**

它是类与类之间最常用的一种关系，它是一种结构化关系，用于表示一类对象与另一类对象之间有联系。

① **一般关联**

【含义】：是一种拥有的关系，它使一个类知道另一个类的属性和方法；可以是双向的，也可以是单向的。如老师和学生、顾客和商品等。

【代码体现】：成员变量

【箭头及指向】：带普通箭头的实心线，指向被拥有者。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。

② **聚合（Aggregation）**

【含义】：是整体与部分的关系，且部分可以离开整体而单独存在。如汽车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。

【代码体现】：成员变量

【箭头及指向】：带空心菱形的实心线，菱形指向整体。 

③ **组合（Composition）**

【含义】：是整体与部分的关系，但部分不能离开整体而单独存在。如公司和部门是整体和部分的关系，没有公司就不存在部门。组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

【代码体现】：成员变量

【箭头及指向】：带实心菱形的实线，菱形指向整体 

- **泛化（Generalization）**

【含义】：也就是继承关系，用于描述父类与子类之间的关系，父类又称基类或超类，子类又称作派生类。表示一般与特殊的关系，它指定了子类如何特化父类的所有待征和行为。例如：老虎是动物的一种，即有老虎的特性也有动物的共性。

【箭头指向】：带三角箭头的实线，箭头指向父类 

- **实现（Realization）**

【含义】：是一种类与接口的关系，表示类是接口所有特征和行为的实现者。例如：定义了一个交通工具接口 Vehicle，包含一个抽象操作 move()，在类 Ship 和类 Car 中都实现了该 move()操作，不过具体的实现细节将会不一样。Ship 和 Car 类与接口 Vehicle 是实现关系。

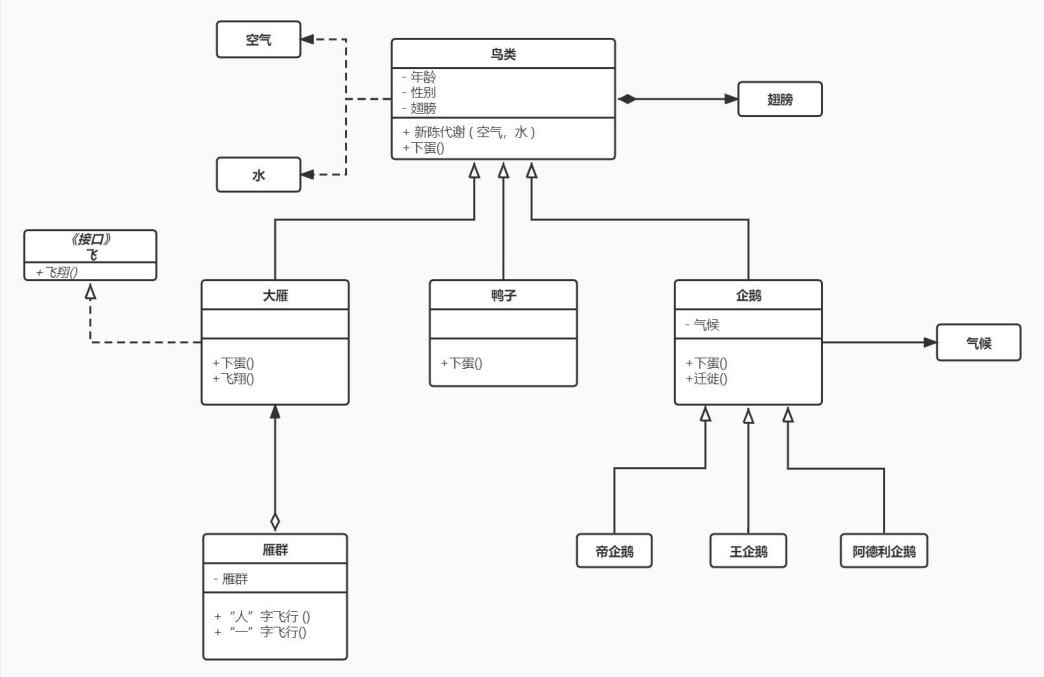
【箭头指向】：带三角箭头的虚线，箭头指向接口 

- **依赖（Dependency）**

【含义】：是一种使用关系，特定事物的改变有可能会影响到使用该事物的其他事物，在需要表示一个事物使用另一个事物时使用依赖关系。例如：驾驶员开车，在 Driver 类的 drive()方法中将 Car 类型的对象 car 作为一个参数传递，以便在 drive()方法中能够调用 car 的 move()方法，且驾驶员的 drive()方法依赖车的 move()方法，因此类 Driver 依赖类 Car。

【代码表现】：局部变量、方法的参数或者对静态方法的调用

【箭头及指向】：带箭头的虚线，指向被使用者 



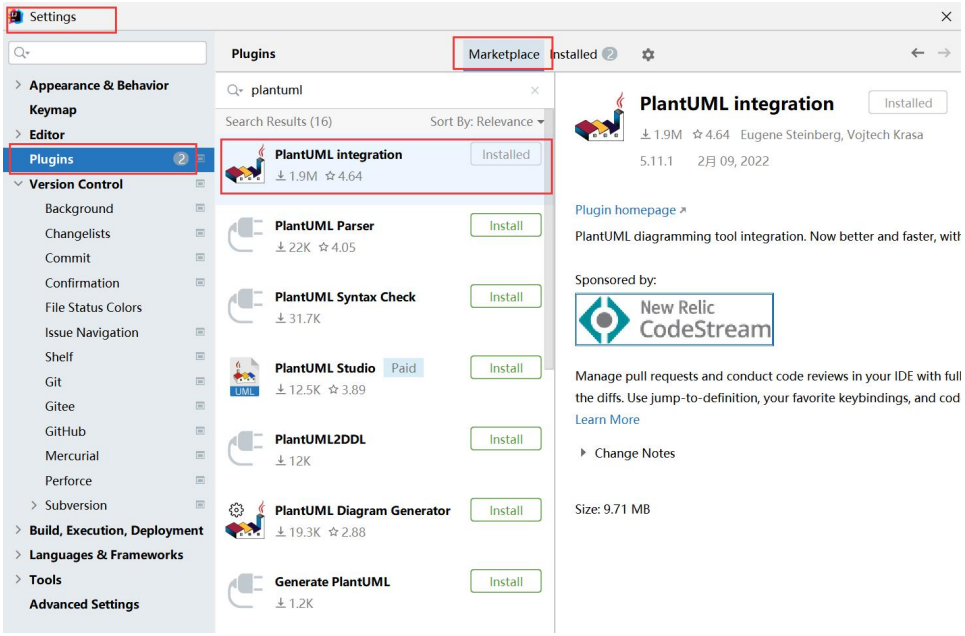
4.3.3 PlantUML 插件

PlantUML 是一个开源项目，支持快速绘制时序图、用例图、类图、对象图、活动图、组件图、部署图、状态图和定时图。Plantuml 本质上是也算一门可以快速画图的设计语言,通过简单的描述语言绘制，非常符合程序员的习惯。

(1) Intellij IDEA 安装插件

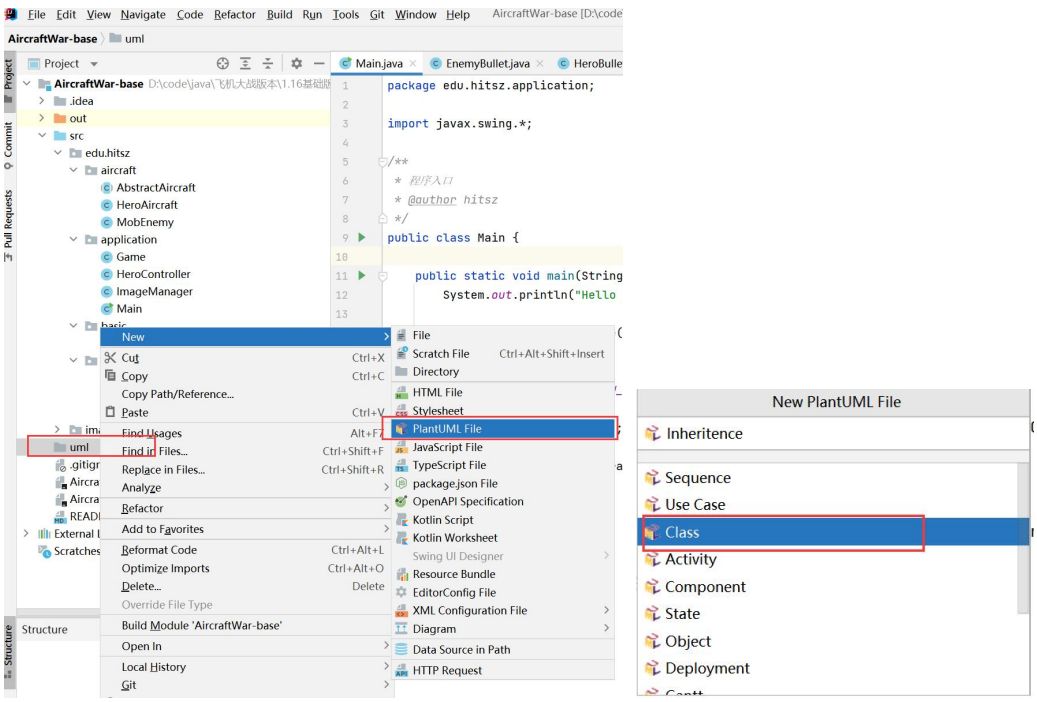
Plantuml 提供了 Intellij IDEA 插件，通过 IDEA 的插件管理搜索 PlantUml 并集成安装后，重启 IDEA 即可。

点击 File --> Settings --> Plugins --> Marketplace，搜索 PlantUml。



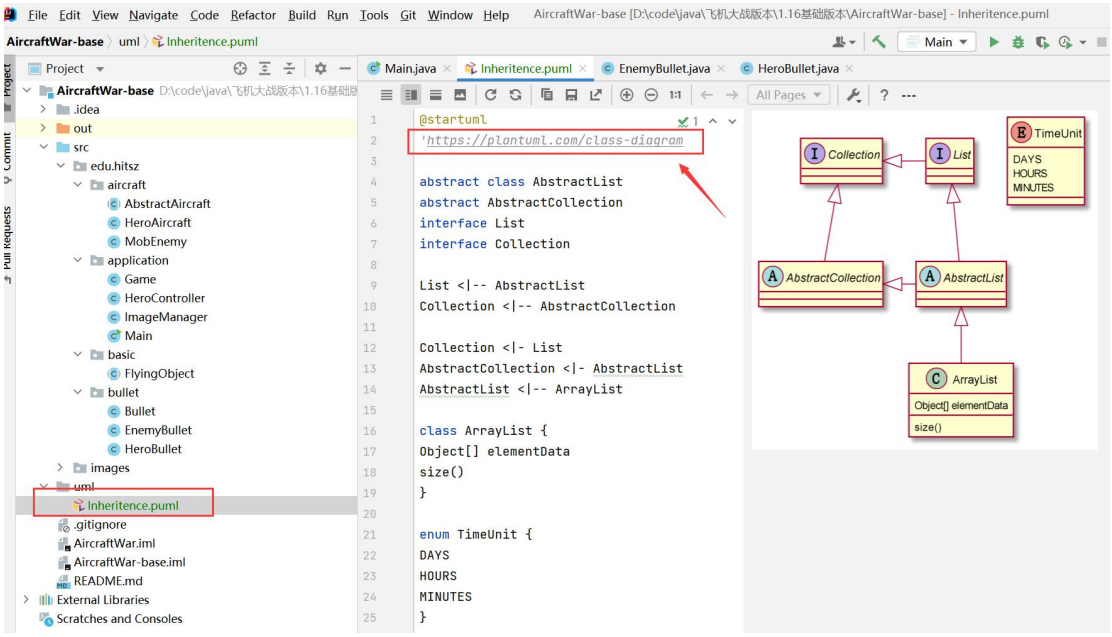
(2) PlantUML 绘制类图示例

在项目下新建 uml 文件夹，右键选择 New-->PlantUML File 创建文件，选择 Class 类图。puml 文件可以像代码一样在 IDEA 中进行管理，即通过 IDEA 可以完成日常设计，开发过程中的大多数工作。

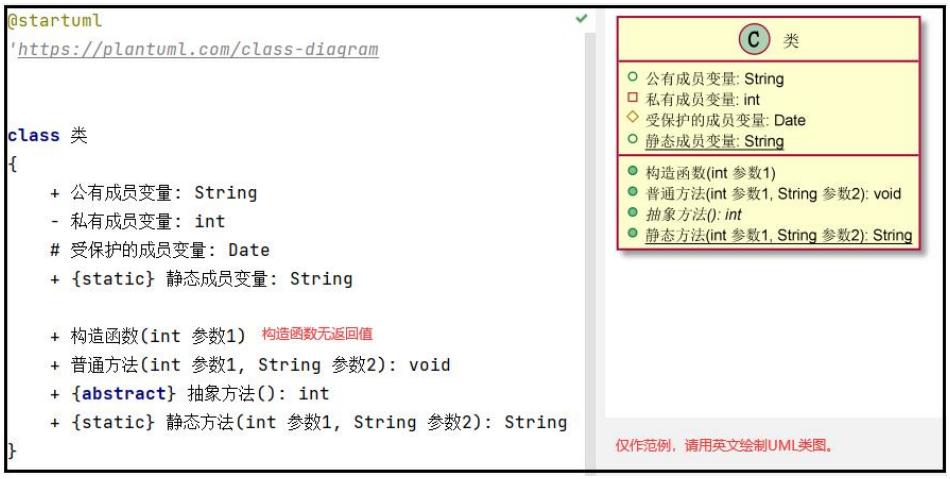


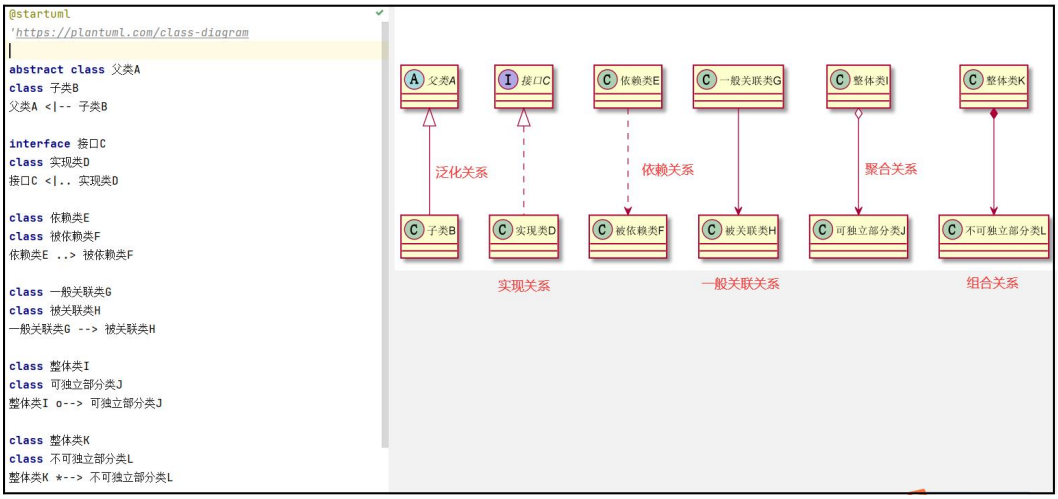
(3) plantUML 类图的语法和功能

请参考官网说明：[类图的语法和功能 (plantuml.com)]



请注意 UML 类绘制的规范性，包括类、抽象类、成员变量、方法、可见性、返回值、联系的符号等，按照指导书 4.3.1 和 4.3.2 节的要求绘制 UML 类图。





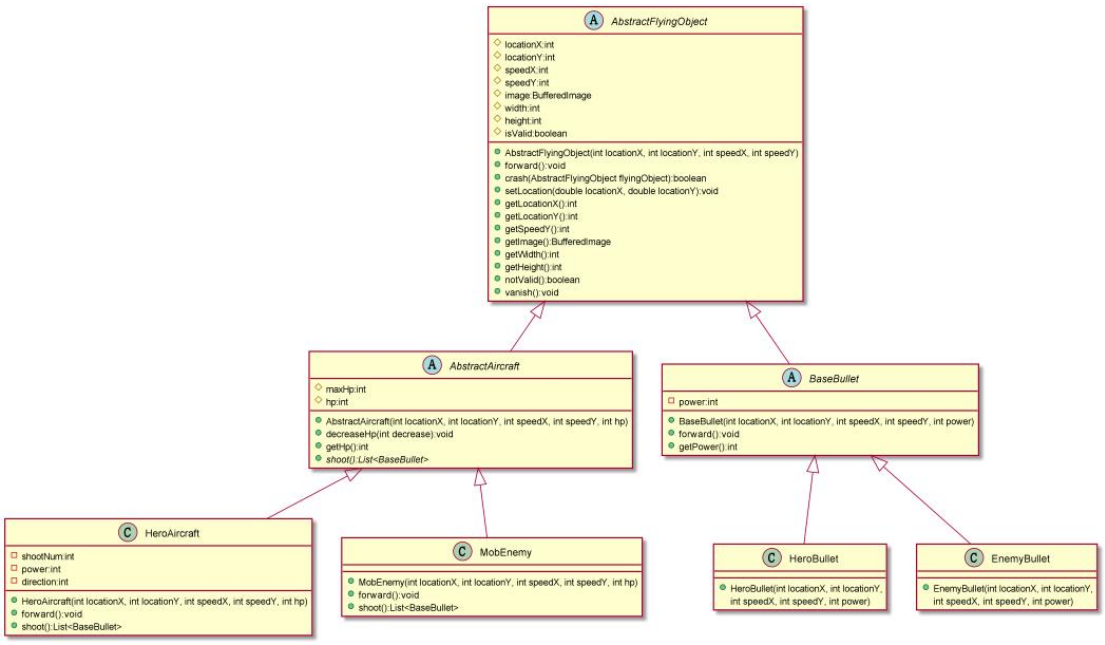
(4) 飞机大战模板程序类图示例

Inheritence.puml

```
@startuml
'https://plantuml.com/class-diagram

abstract class AbstractFlyingObject
{
    # locationX:int
    # locationY:int
    # speedX:int
    # speedY:int
    # image:BufferedImage
    # width:int
    # height:int
    # isValid:boolean

    + AbstractFlyingObject(int locationX, int locationY, int speedX, int speedY)
    + forward():void
```

```
        + crash(AbstractFlyingObject flyingObject):boolean
        + setLocation(double locationX, double locationY):void
        + getLocationX():int
        + getLocationY():int
        + getSpeedY():int
        + getImage():BufferedImage
        + getWidth():int
        + getHeight():int
        + notValid():boolean
        + vanish():void
}
abstract class AbstractAircraft
{
        # maxHp:int
        # hp:int
        + AbstractAircraft(int locationX, int locationY, int speedX, int speedY, int hp)
        + decreaseHp(int decrease):void
        + getHp():int
        + {abstract} shoot():List<BaseBullet>
  }


class HeroAircraft {
        - shootNum:int
        - power:int
        - direction:int
        + HeroAircraft(int locationX, int locationY, int speedX, int speedY, int hp)
        + forward():void
        + shoot():List<BaseBullet>
}

AbstractAircraft <|-- HeroAircraft

class MobEnemy {
        + MobEnemy(int locationX, int locationY, int speedX, int speedY, int hp)
        + forward():void
        + shoot():List<BaseBullet>
}
AbstractAircraft <|-- MobEnemy

abstract class BaseBullet
{
        - power:int
        + BaseBullet(int locationX, int locationY, int speedX, int speedY, int power)
        + forward():void
      + getPower():int
}

class HeroBullet {
        + HeroBullet(int locationX, int locationY,
          int speedX, int speedY, int power)
}

class EnemyBullet {
```

```
        + EnemyBullet(int locationX, int locationY,
          int speedX, int speedY, int power)
}

BaseBullet <|-- HeroBullet
BaseBullet <|-- EnemyBullet

AbstractFlyingObject <|-- AbstractAircraft
AbstractFlyingObject <|-- BaseBullet

@enduml
```



4.4 重构代码，为系统增加精英敌机类和三种道具类

请根据设计的 UML 类图，重构代码，在项目中添加精英敌机类和三种道具类，以及它们的父类。只需完成本次实现的基本要求，功能细节将在后续实验中逐步完善。本实验暂不实现 Boss 机，UML 类图和代码中无需包含 Boss 机。

本次实验提交版本需完成以下功能：

- ✓ 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）；
- ✓ 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无需具体实现，生效时只需在控制台打印“FireSupply

active!”“BombSupply active!”语句即可。

5. 实验要求

✧ **提交内容：项目代码（含 UML 类图，整个项目压缩成 zip 包）**

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。