# d

《现代软件工程课程设计》中期报告

|  |  |  |
| --- | --- | --- |
| 小组编号 | 13组 | |
| 小组成员 | 学号 | 姓名 |
| 2022141090079 | 高淞 |
| 2023141460294 | 李林润 |

提交报告时间： 2025 年 11 月 19 日

**目 录**

1 建模 1

1.1 场景建模 1

1.2 类建模 2

1.3 功能建模/行为建模 2

2 已完成工作 2

2.1 红包游戏 2

2.1.1 已实现功能 2

2.1.2 界面截图及说明 2

2.1.3 代码结构及说明 3

2.2 统计代码量 4

2.2.1 已实现功能及测试结果 4

2.2.2 界面截图及说明 4

2.2.3 代码结构及说明 4

2.3 小鸭扮靓 4

2.3.1 已实现功能 4

2.3.2 界面截图及说明 4

2.3.3 代码结构及说明 4

2.4 AI对话 5

2.4.1 已实现功能 5

2.4.2 界面截图及说明 5

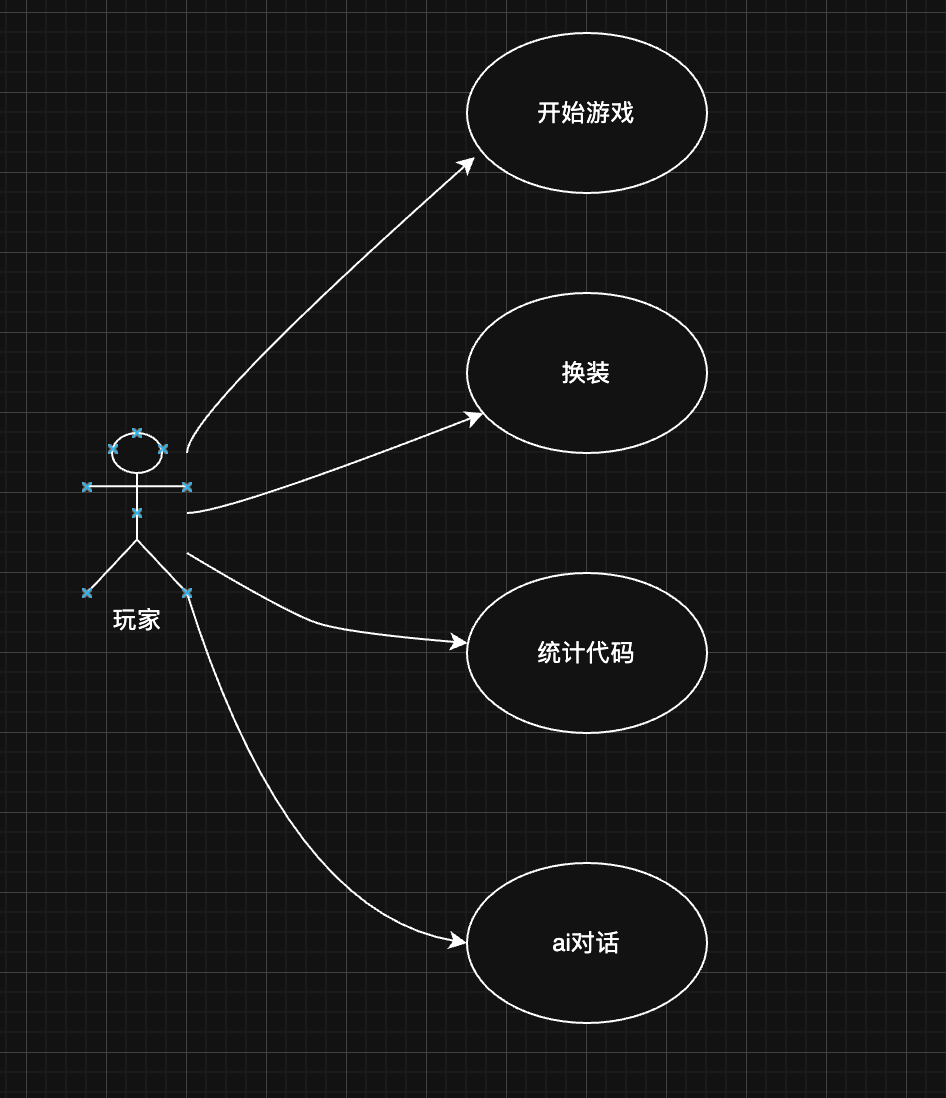
2.4.3 代码结构及说明 5

# 1 建模

【以小鸭扮靓为例，给出use case、class diagram、activity diagram / sequence diagram】

## 1.1 场景建模

用例：小鸭扮靓

1.1.1 功能划分

1.1.2 需求分配表

|  |  |  |  |
| --- | --- | --- | --- |
| 需求编号 | 功能名称 | 优先级 | 说明 |
| TF-01 | 功能菜单 | 1 | 用户点击唐老鸭出现功能选择框 |
| TF-02 | 开始游戏 | 1 | 用户在功能框输入start开始游戏 |
| TF-03 | 换装 | 1 | 用户在功能框输入wear换装 |
| TF-04 | 统计代码 | 1 | 用户在功能框输入count统计代码 |
| TF-05 | ai对话 | 1 | 用户在功能框输入talk开始ai对话 |

1.1.3 功能描述

a.开始游戏

**功能编号:TF-01**

**用例名称:开始游戏**

**用例描述:玩家打开游戏界面点击指定区域后, 在弹出窗口输入‘start’开始红包游戏**

**前置条件: 用户输入正确**

**后置条件:弹出提示窗后开始游戏**

**参与成员:用户, 系统**

**输入数据: 输入‘start’**

**输出数据:开始红包游戏**

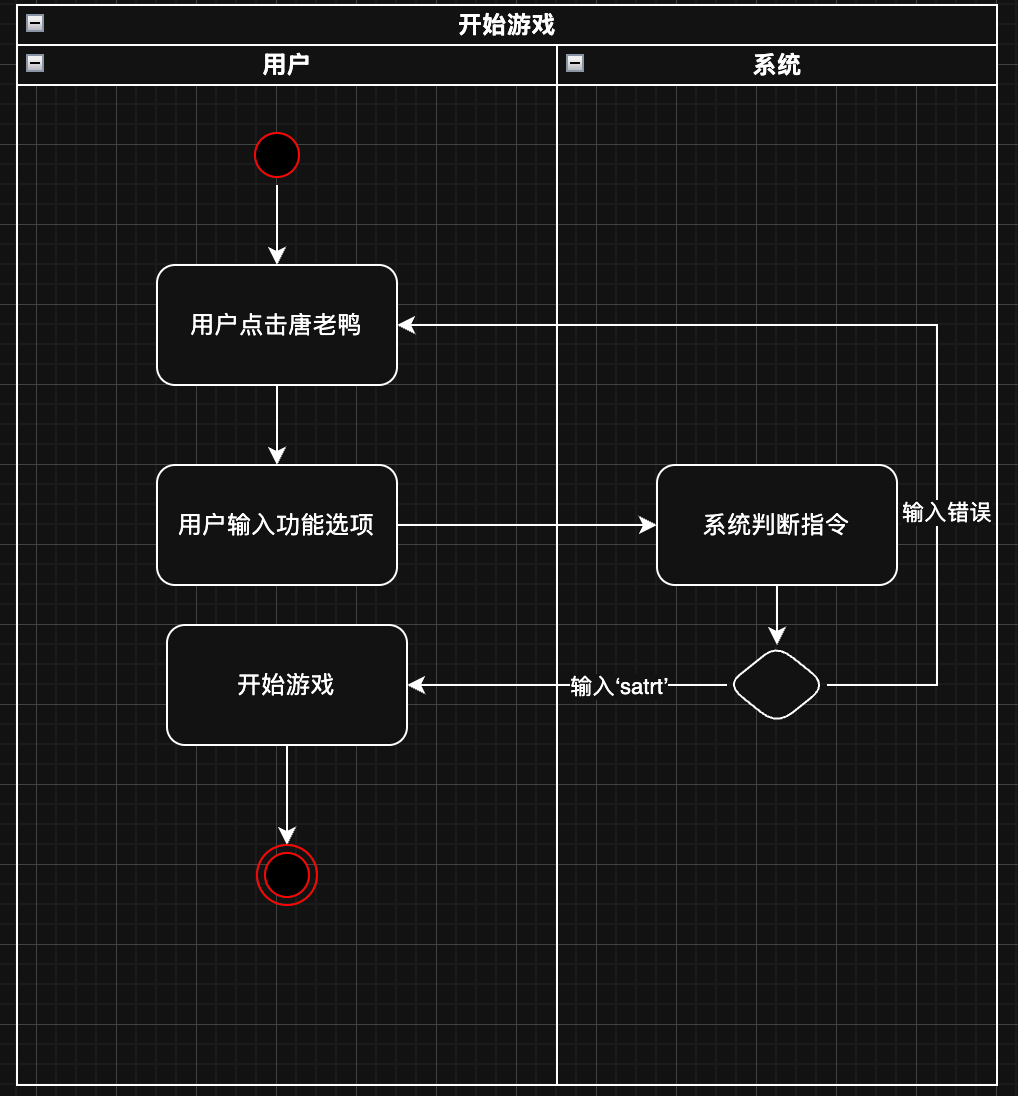
**基本事件流:**

1. **用户点击唐老鸭**
2. **用户在弹出窗口的输入框输入‘start’并确认**
3. **提醒用户开始游戏**
4. **开始游戏**

**异常以及备选流:**

**用户输入错误指令: 提示报错并返回主界面**

**其他说明: 暂无**



b.换装

**功能编号:TF-02**

**用例名称:换装**

**用例描述:用户选择换装后弹出对应窗口, 自定义天气和时间后更换唐小鸭的装扮**

**前置条件:用户在功能选择框填入‘wear’**

**后置条件:用户选择时间和天气**

**参与成员:用户、系统**

**输入数据:‘wear’、天气、时间**

**输出数据:唐小鸭装扮**

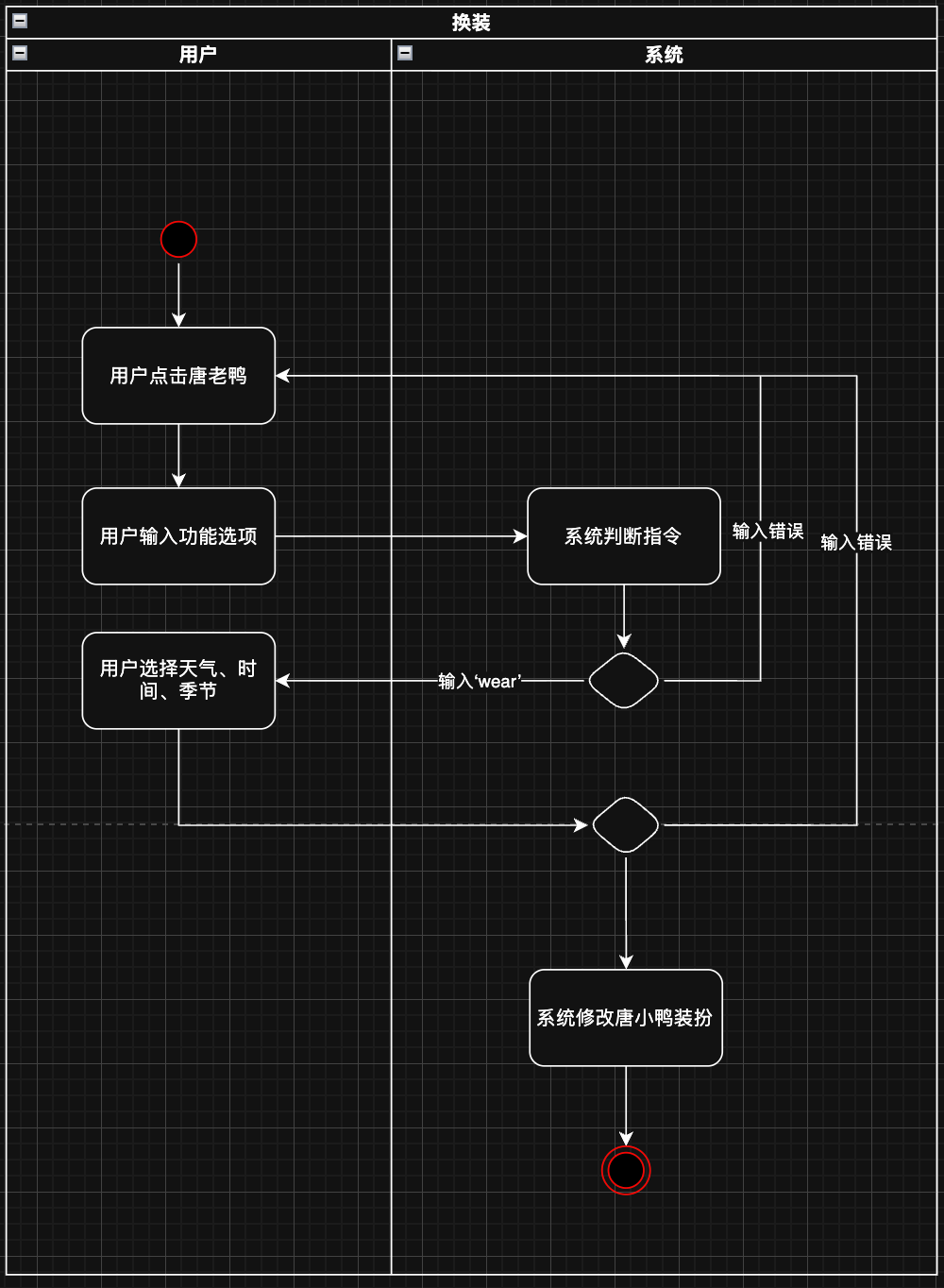
**基本事件流:**

1. **用户在功能选择界面输入‘wear’**
2. **用户自定义时间、天气**
3. **用户点击确认**

**异常以及备选流:**

**用户输入错误指令: 提示报错并返回主界面**

**其他说明: 暂无**



c.统计代码

**功能编号:TF-03**

**用例名称:统计代码**

**用例描述:用户选择统计代码功能, 在对应界面自定义选项后输出**

**前置条件:用户正确输入功能选择命令**

**后置条件:进入统计代码界面**

**参与成员:用户、系统**

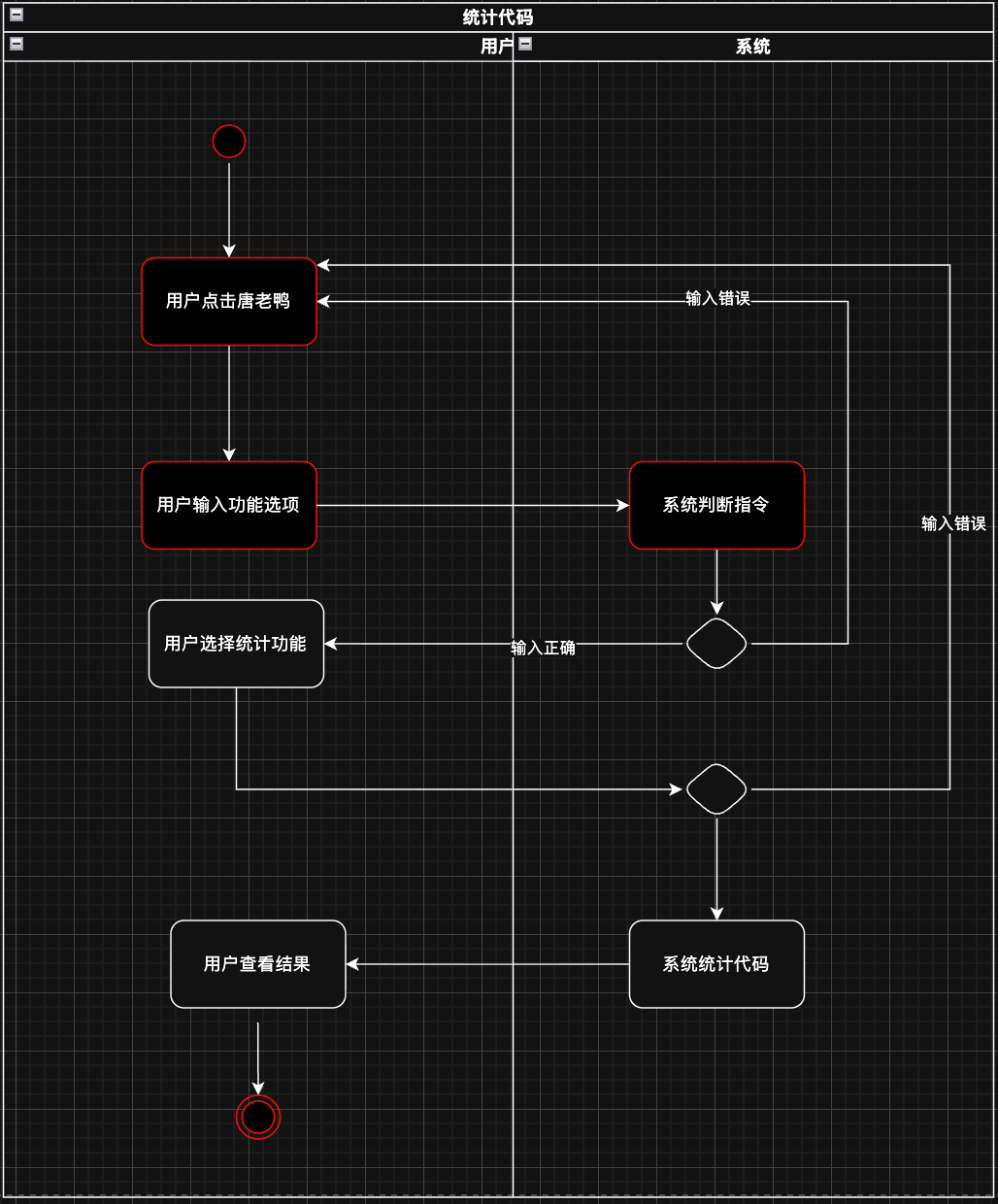
**输入数据:‘count’、自定义统计选项**

**输出数据:统计结果**

**基本事件流:**

1. **用户点击唐老鸭**
2. **用户输入‘count’**
3. **用户在统计代码功能框自定义选项**
4. **开始统计得到结果**

**异常以及备选流:**

**用户输入错误指令: 提示报错并返回主界面**

**其他说明:暂无**

d.ai对话

**功能编号:TF-04**

**用例名称:ai对话**

**用例描述:用户在功能选择页面输入‘talk’后弹出聊天框**

**前置条件:用户输入正确指令**

**后置条件:用户在聊天框输入内容并发送**

**参与成员:用户、系统**

**输入数据:‘talk’、聊天内容**

**输出数据:系统回复**

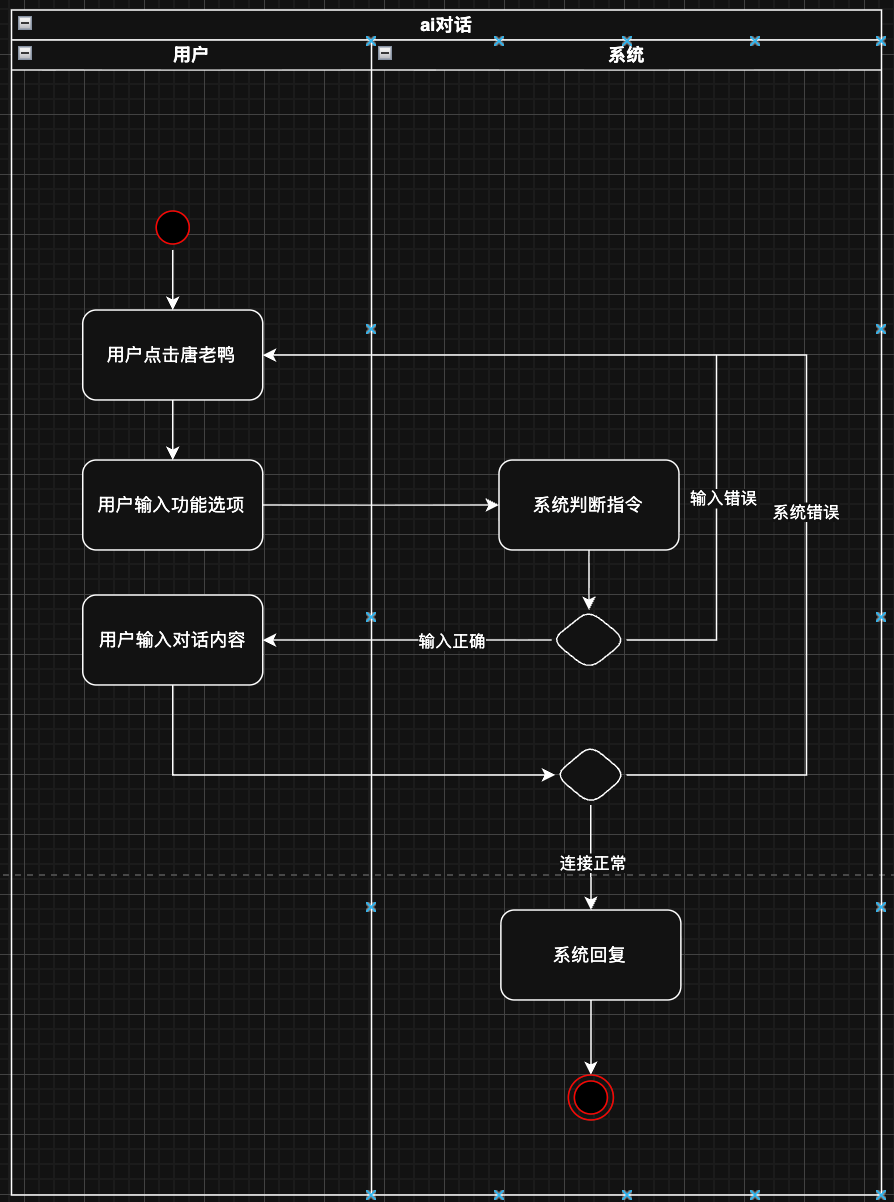
**基本事件流:**

1. **用户点击唐老鸭**
2. **用户输入‘talk’**
3. **用户发送聊天内容**
4. **系统回复**

**异常以及备选流:**

**用户输入错误指令: 提示报错并返回主界面**

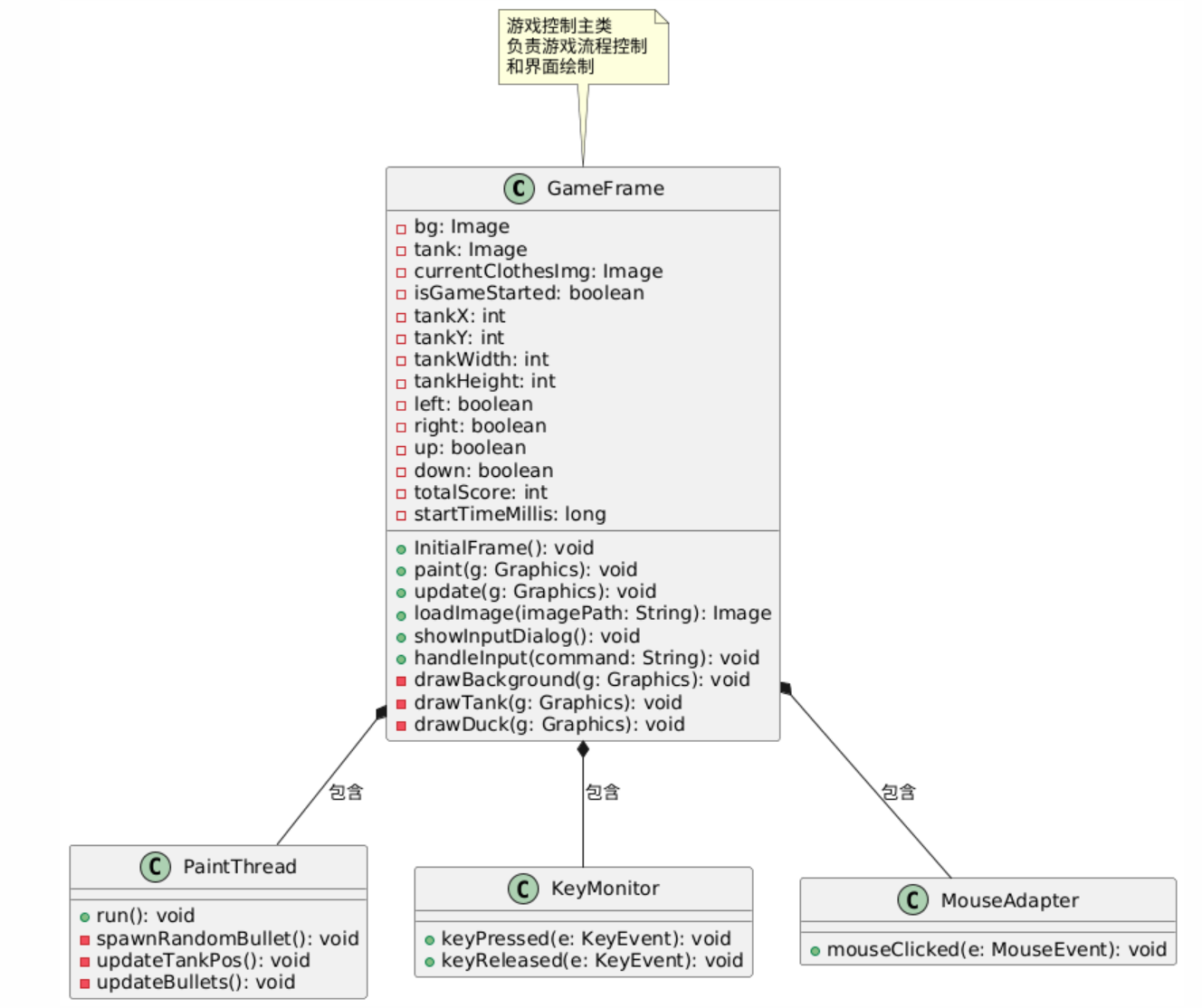
**其他说明:暂无**

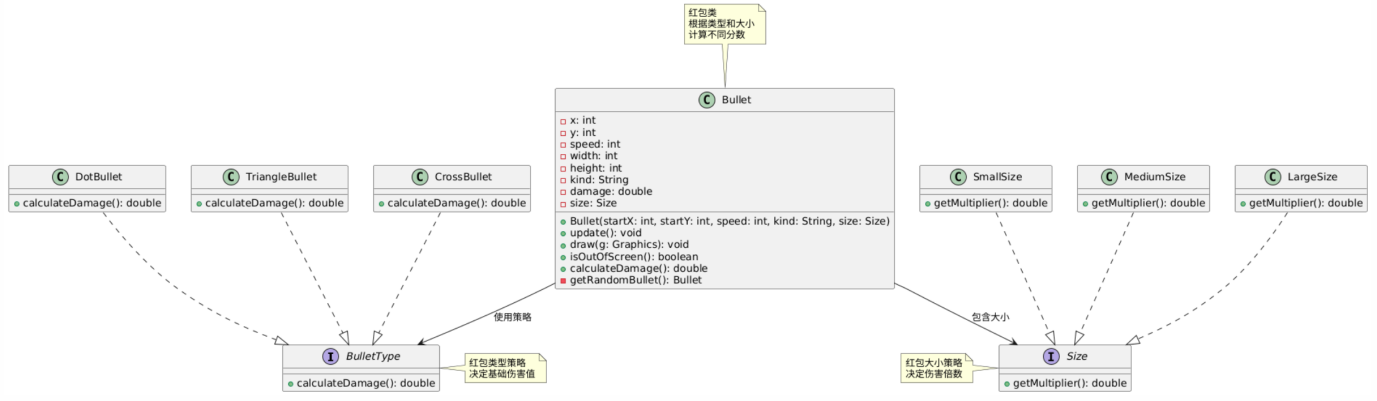


## 1.2 类建模

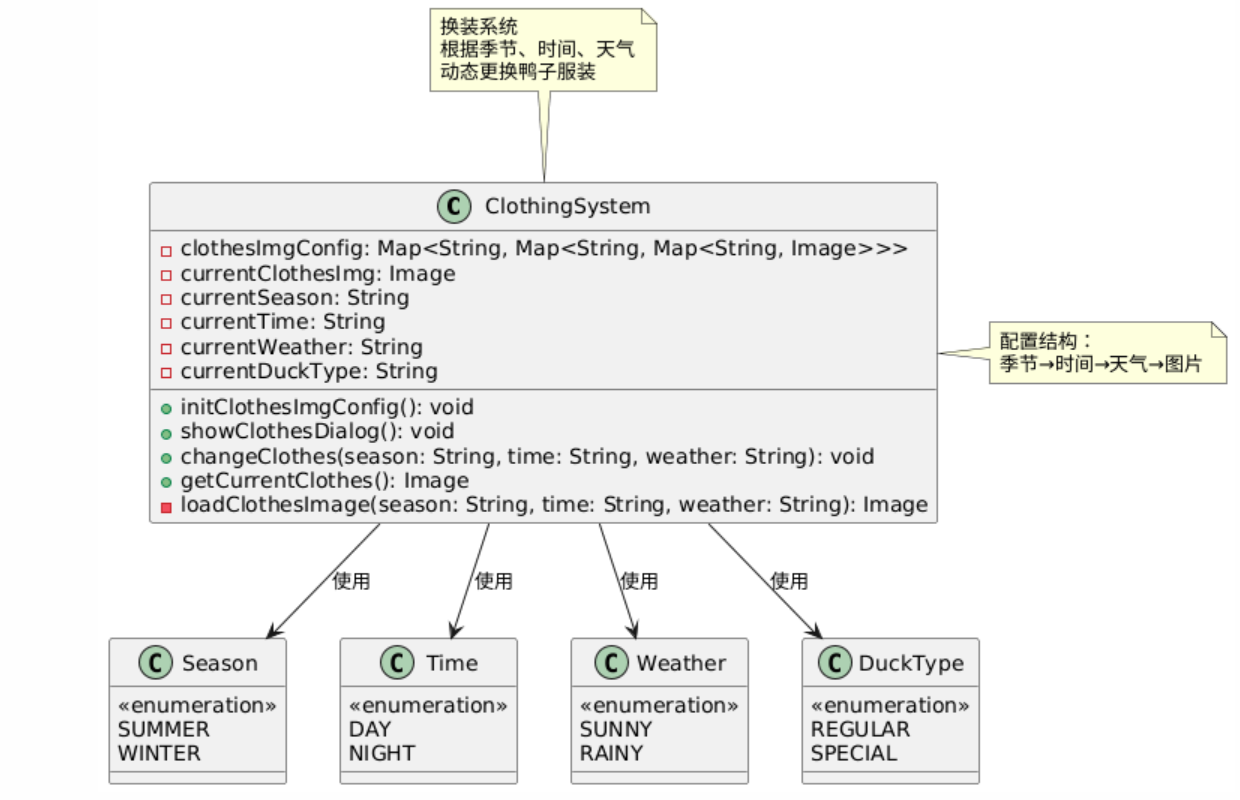
【基于1.1的用例，给出类模型。】

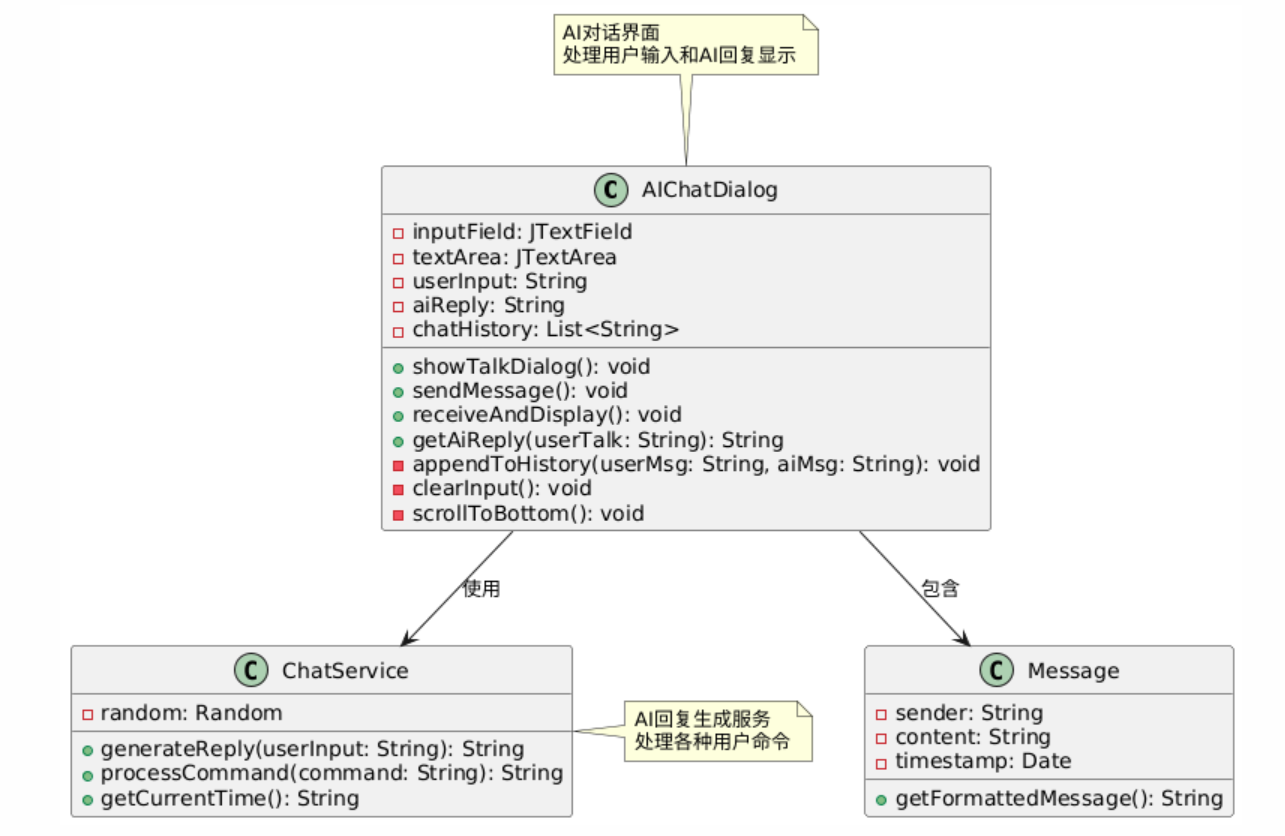
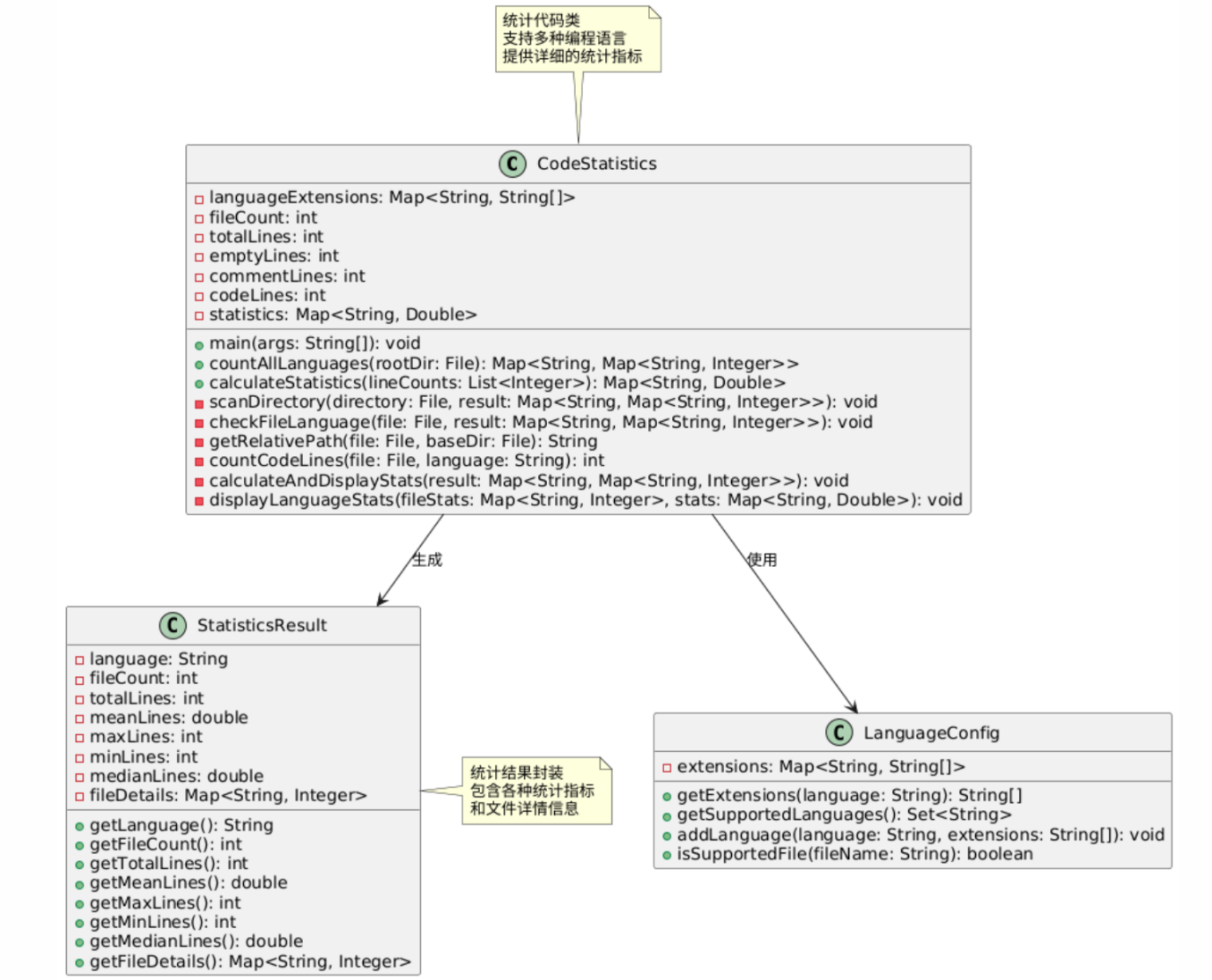
1.2.1类属性描述和操纵定义

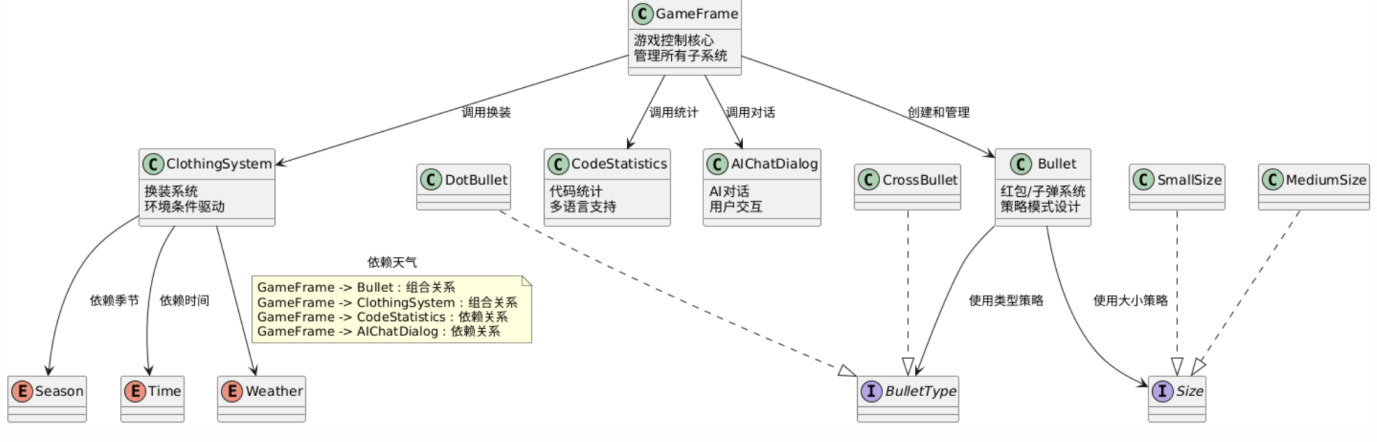
1. 游戏控制类
   1. 类属性: 背景图, 坦克图, 鸭子图
   2. 类方法: 绘制背景, 绘制坦克, 绘制鸭子, 鼠标监听, 键盘监听
   3. 类图
2. 红包类
   1. 类属性: 红包大小, 红包种类, 红包分数
   2. 类方法: 计算分数, 随机生成红包
   3. 类图



1. 换装类
   1. 类属性: 天气、时间、季节、鸭子类型
   2. 类方法: 换装
   3. 类图



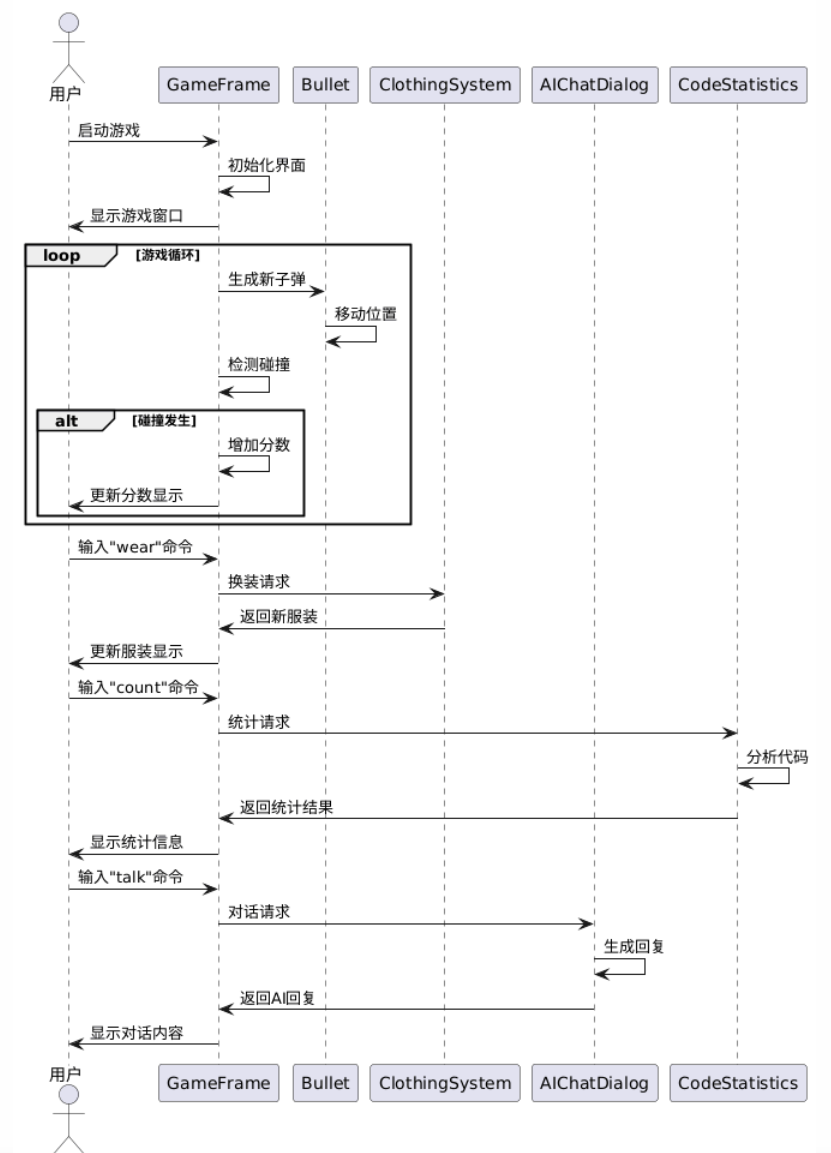
1. 统计代码类
   1. 类属性: 语言扩展名映射（C、C++、Java、Python、C#）、文件总数、总 行数、空行数、注释行数、有效代码行数、统计指标（均值、最大值、最小值、中位数等）、扫描结果、当前扫描目录
   2. 类方法: 递归扫描目录、检查文件语言类型、统计代码行数、计算统计结果
   3. 类图
2. ai对话类
   1. 类属性: 输入内容、回复内容
   2. 类方法: 发送内容, 接受并显示
   3. 类图

1.2.2类间关系分析图

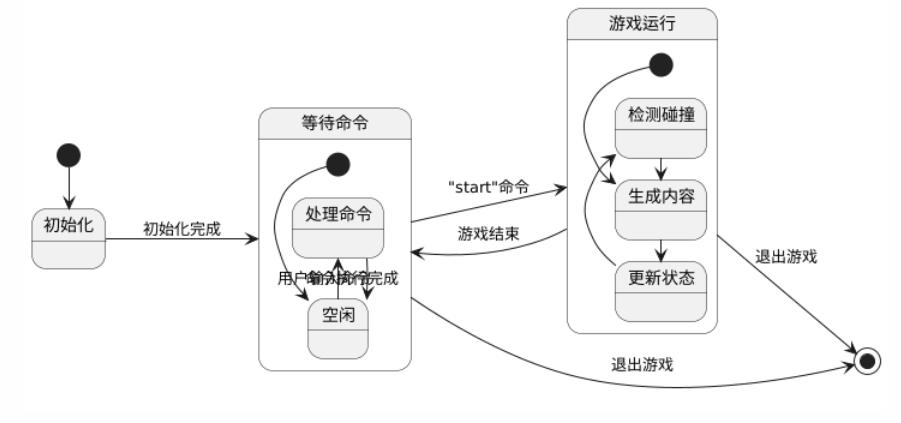
## 1.3 功能建模/行为建模

【基于1.1的用例，给出相应模型。】

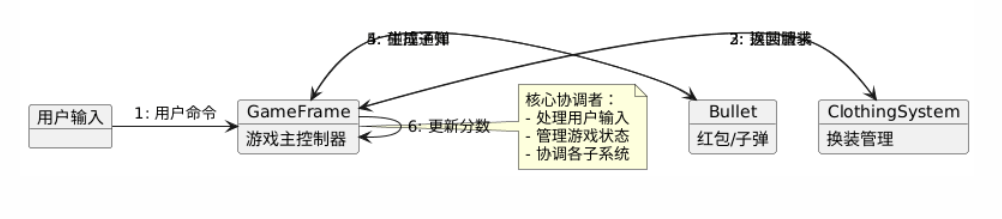
1.3.1时序图



1.3.2 状态图



1.3.3 协作图



# 2 已完成工作

【截止目前为止，

1. 已实现功能的文字描述；

2. 界面截图（每个功能不超过三幅图）；

3. 对应的代码结构及文字说明。代码结构可使用IDE截图，例如，IDEA的Project窗口中的代码结构；也可以画类图。注：截图采用白色背景；不贴源代码。】

## 2.1 红包游戏

### 2.1.1 已实现功能

第六周

坦克大战: 面向对象重构, 实现基本功能: 碰撞检测, 统计时长

将坦克大战改为发红包

第七周

红包设计不同大小形状分数

场景变更: 点击唐老鸭出现文本框, 按照文本框输入增加功能

第八周

鸭子装扮

统计代码量功能扩充

第九周

在统计代码的ui内添加选项

统计代码量功能扩充:C语言,给出均值、最大值、最小值、中位数等数据

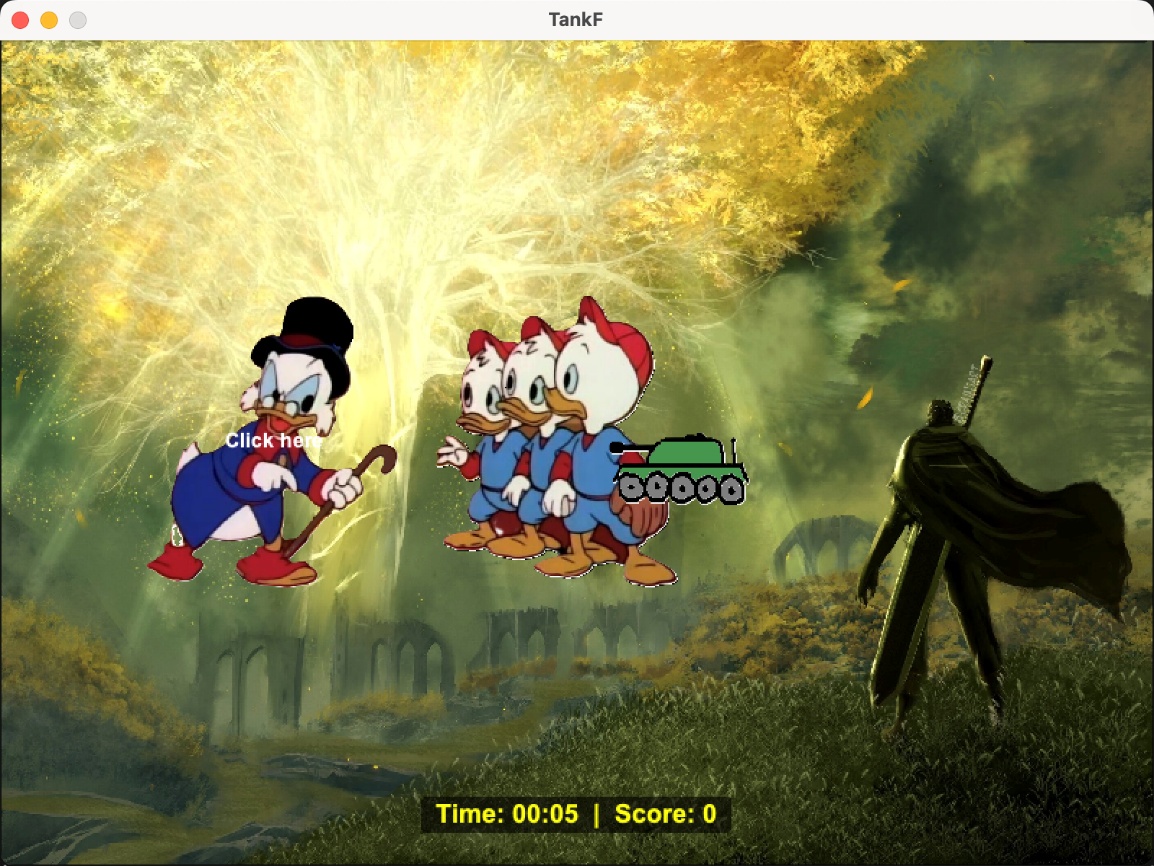
第十周

鸭子不同的飞行/叫声行为的封装和实现,要求满足可扩展可维护

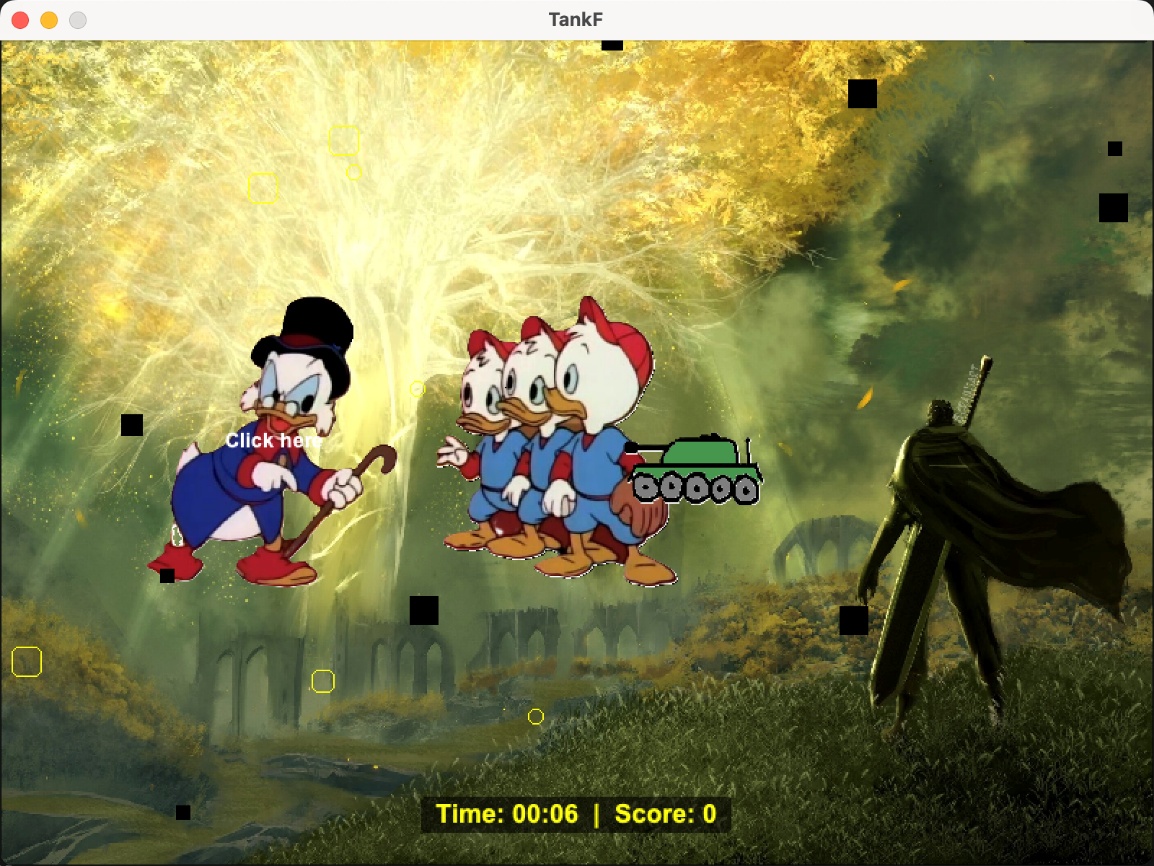
统计代码量: 将结果存入文件内(.csv .json .xlsx)在对话框指定生成指定格式

### 2.1.2 界面截图及说明

a.实现基本界面功能



b.实现将子弹改为大小,样式不同的红包



### 2.1.3 代码结构及说明

1.入口与窗口初始化

main()

初始化 Duck 行为/声音并调用 InitialFrame()

InitialFrame()

Frame 基本设置（title/size/center/resizable）

启动 PaintThread（渲染循环）

注册 KeyMonitor、WindowAdapter、MouseListener

initClothesImgConfig() 调用来构建衣服图片配置表

2.资源加载与配置

字段：多个 Image（bg、tank、clickAreaImg、各季节/时间/天气衣服图）

方法：loadImage(String imagePath) — 通过 Class.getResource 加载图片；若找不到则退出

字段：clothesImgConfig (Map<String, Map<String, Map<String, Image>>>) — 季节->时间->天气->Image 的配置结构

方法：initClothesImgConfig() — 填充 clothesImgConfig

3.UI 输入与命令处理

showInputDialog()

命令提示文本（start/wear/count/talk）+ TextField + Submit 按钮

提交后调用 handleInput(command)

handleInput(String command)

路由：start / wear / count / talk / default 并触发相应对话或逻辑

## 2.2 统计代码量

### 2.2.1 已实现功能及测试结果

统计指定文件夹中所有文件的代码量

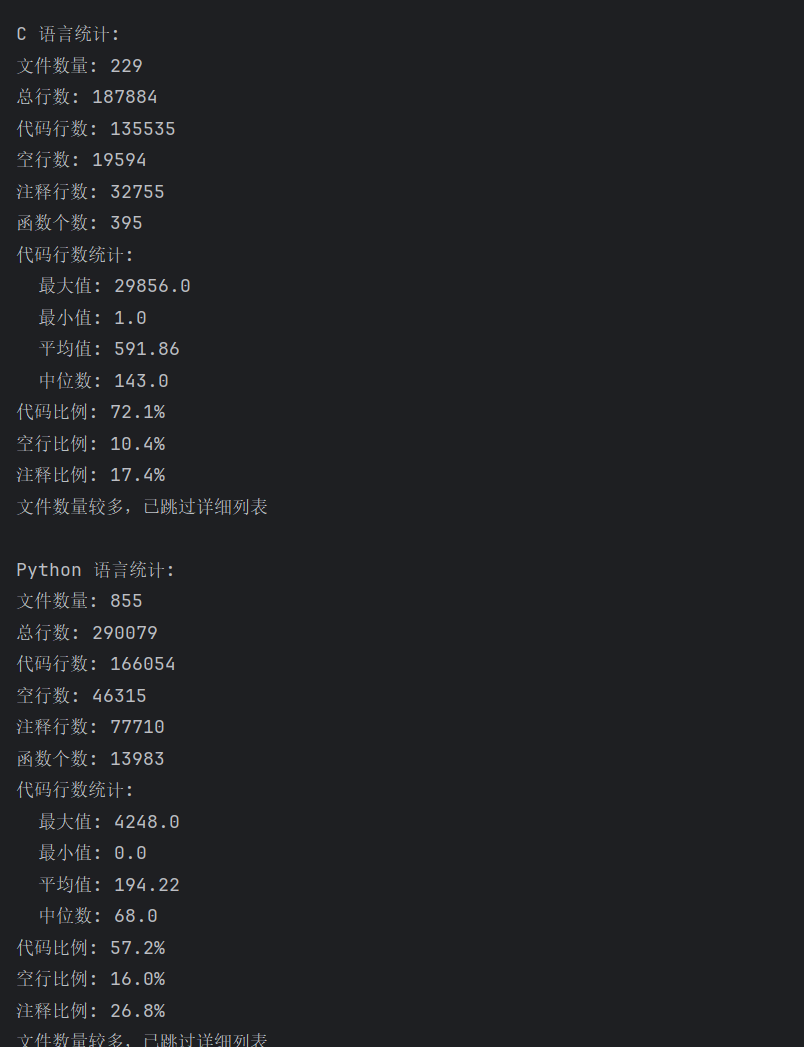
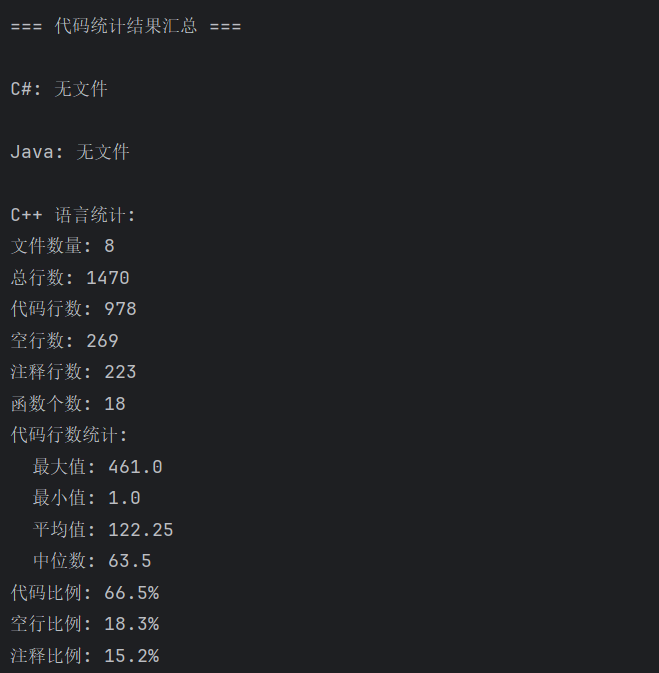
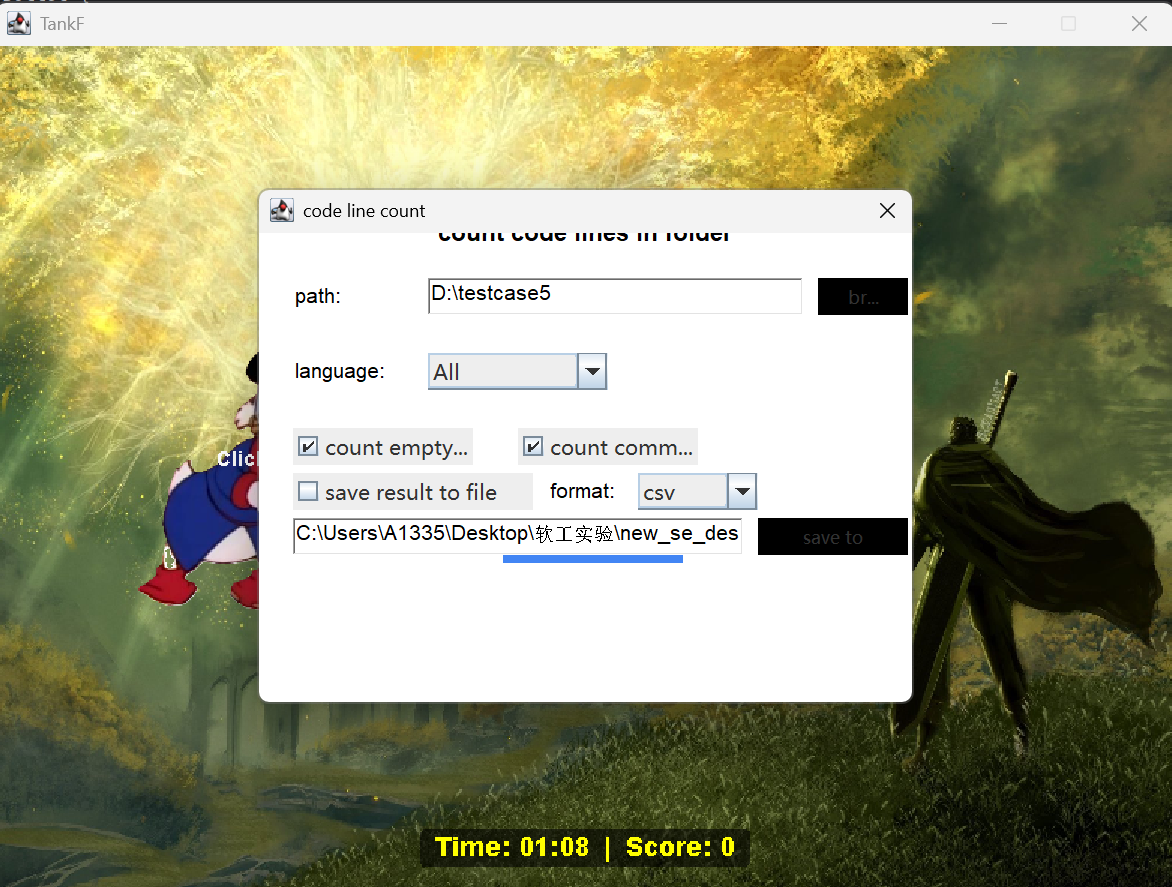
testcase5测试结果：

| **语言** | **源文件数** | **代码行数** | **空行数** | **注释行数** | **函数个数** | **最大值** | **最小值** | **均值** | **中位数** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | 229 | 135535 | 19594 | 32755 | 395 | 29856 | 1 | 591.9 | 143 |
| C++ | 8 | 978 | 269 | 223 | 18 | 461 | 1 | 122.2 | 63.5 |
| python | 855 | 166054 | 46315 | 77710 | 13983 | 4248 | 0 | 194.2 | 68 |

【给出testcase5的测试结果。注：该结果以表格形式给出，如图2.1所示；若不能实现相关UI，则人工在表格中录入数据。】



### 2.2.2 界面截图及说明



### 2.2.3 代码结构及说明

## 1. 入口与初始化

## main()：该方法是程序的入口，首先检查 testcase5 文件夹是否存在。如果存在，开始统计代码量；如果不存在，则输出错误信息。

## countAllLanguages(File rootDir)：该方法接收文件夹路径，递归扫描目录，并统计各编程语言的代码行数。返回一个包含语言和统计数据的结果 Map。

## scanDirectory(File directory, Map<String, Map<String, Integer>> result)：递归扫描目录中的文件。若为子目录，则继续递归；若为文件，则调用 checkFileLanguage() 判断并统计代码行数。

## checkFileLanguage(File file, Map<String, Map<String, Integer>> result)：根据文件扩展名判断文件属于哪种语言（如 C、Java、Python 等）。若匹配，则调用 countCodeLines() 统计有效代码行数，并将结果存入 result。

## countCodeLines(File file, String language)：读取文件内容，统计有效代码行数。跳过空行和注释行（根据语言类型区分注释规则）。

## getRelativePath(File file, File baseDir)：获取文件的相对路径，方便在统计结果中显示。

## 2. 统计结果计算与显示

## calculateAndDisplayStats(Map<String, Map<String, Integer>> result)：计算并显示统计结果。包括文件数、代码行数的统计，以及均值、最大值、最小值、中位数等统计数据。

## calculateStatistics(List<Integer> lineCounts)：对所有文件的行数进行排序，计算均值、最大值、最小值和中位数。

## displayLanguageStats(Map<String, Integer> fileStats, Map<String, Double> stats)：输出每种语言的详细统计数据，显示每个文件的行数及统计摘要。

## 3. 结果存储与导出

## saveToFile(Map<String, Map<String, Integer>> result)：将统计结果保存为指定格式的文件（.csv、.json 或 .xlsx）。

## saveToCsv(Map<String, Map<String, Integer>> result)：将结果保存为 CSV 格式文件。

## saveToJson(Map<String, Map<String, Integer>> result)：将结果保存为 JSON 格式文件。

## saveToXlsx(Map<String, Map<String, Integer>> result)：将结果保存为 Excel 文件（.xlsx 格式）。

## 2.3 小鸭扮靓

### 2.3.1 已实现功能

根据天气,时间,季节的组合选择鸭子装扮

### 2.3.2 界面截图及说明





### 2.3.3 代码结构及说明

衣服选择（wear 命令）

showClothesDialog()

对话框组件：seasonCombo、timeCombo、weatherCombo

确认后从 clothesImgConfig 取图片并设置 currentClothesImg

调用 showTipDialog 显示结果

## 2.4 AI对话

### 2.4.1 已实现功能

本地简单模拟ai对话

todo: 接入api

### 2.4.2 界面截图及说明

### 2.4.3 代码结构及说明

本地 AI 对话（talk 命令）

showTalkDialog()

模态对话框：历史 JTextArea + 输入 JTextField + 发送按钮

发送后异步调用 getAiReply(user\_talk)，先显示占位“正在思考...”，再追加回复

getAiReply(String userTalk)

本地简单规则（匹配问候、时间、help、thanks）+ 随机候选回复

使用 LocalDateTime 格式化当前时间