

# 设计文档 Plant Flowers

## 北京理工大学计算机学院 《Java 程序设计》课程设计

### 1 程序的运行环境、安装步骤

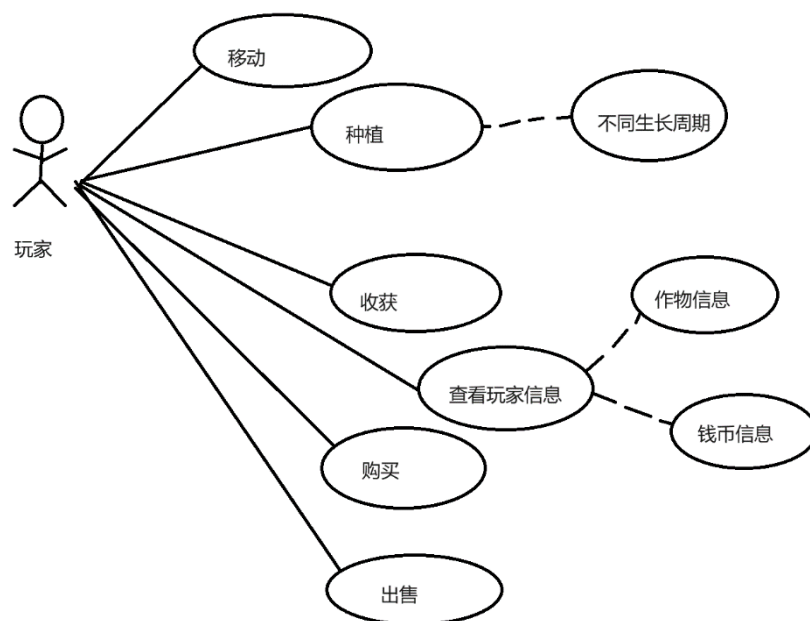
- (1) 运行环境: JDK 19.0.2
- (2) 程序的组成部份: 指可以运行的程序包容 demo.jar 文件, 不需要其他的支持文件。
- (3) 安装步骤:
  - 1) 安装 JRE 17。
  - 2) 将程序 jar 文件复制到计算机上
  - 3) 打开命令提示符窗口, 输入以下命令: `java -jar .\demo.jar`

### 2 程序开发平台

- (1) 代码行数 约 1200 行
- (2) 开发环境: IntelliJ 2022.2 + JDK 19.0.2

### 3 程序功能说明:

- (1) 绘制 UML 图如下



(2) 重点功能说明如下。

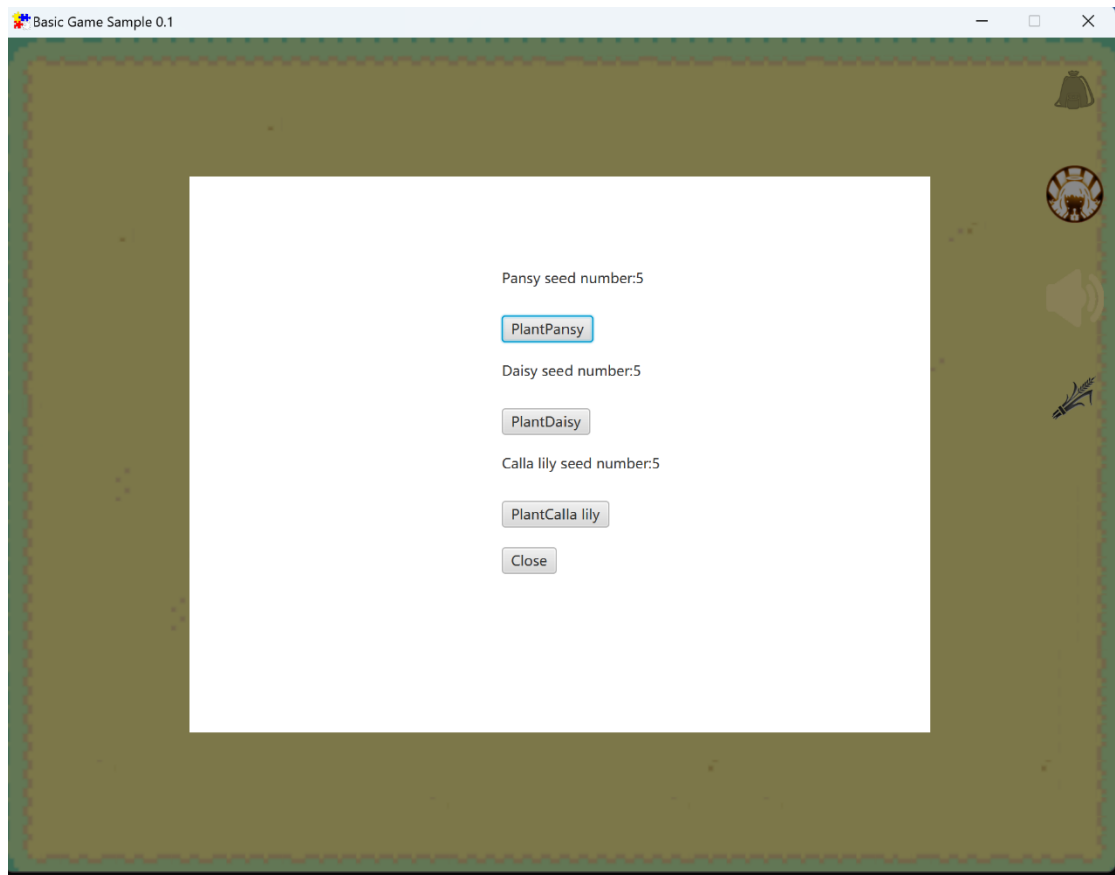
(i) 移动

键盘按住 A 键，玩家形象左移；按住 D 键，玩家形象右移；按住 W 键，玩家形象上移；按住 S 键，玩家形象下移。

(ii) 种植



玩家控制移动到可种植的土地范围内，可种植的问号 ui 显现。



点击问号 ui，弹出子窗体，询问是否种植与种植的花卉种类，点击一种花卉种植。（如果花卉种子数目为零，仍继续种植，会弹出错误子窗体）



点击种植按钮后，对应问号 **ui** 的土地上该种花卉被种植并开始生长。

(iii) 收获



花卉生长完毕成熟后，上方表明可收获的叹号 **ui** 出现。

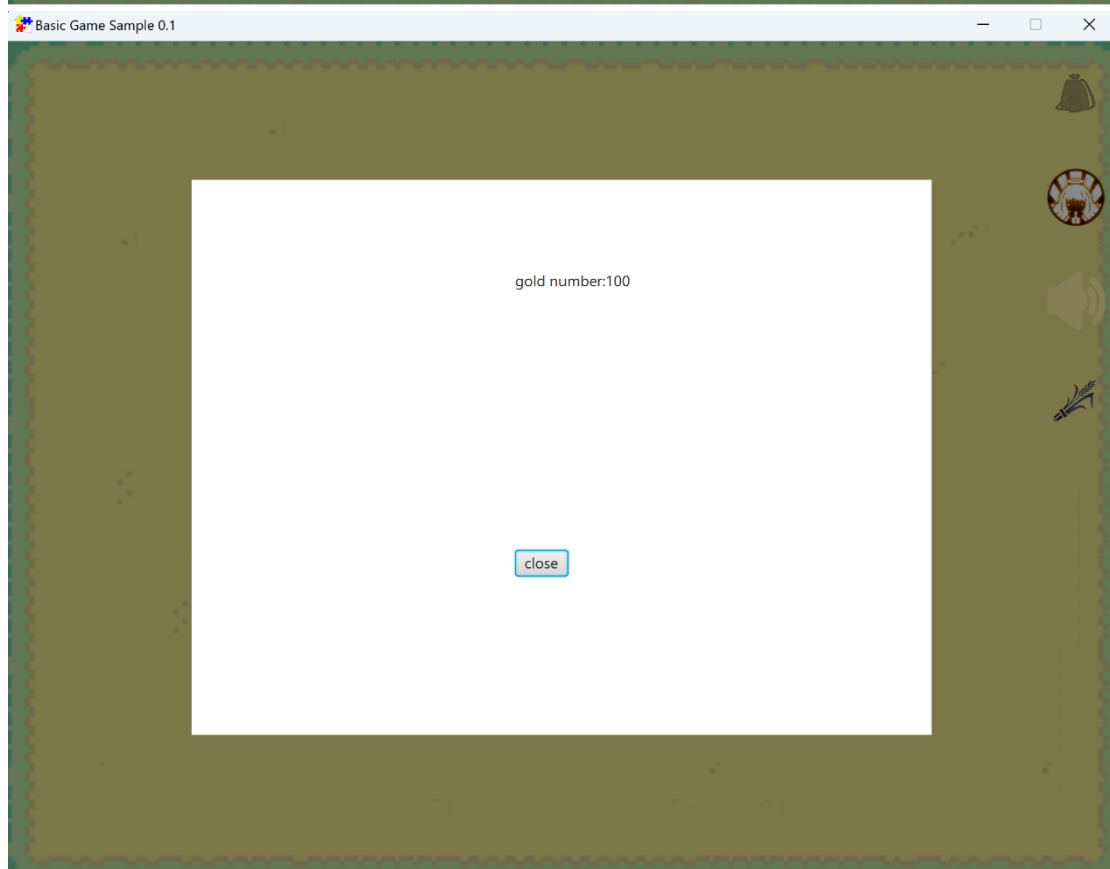
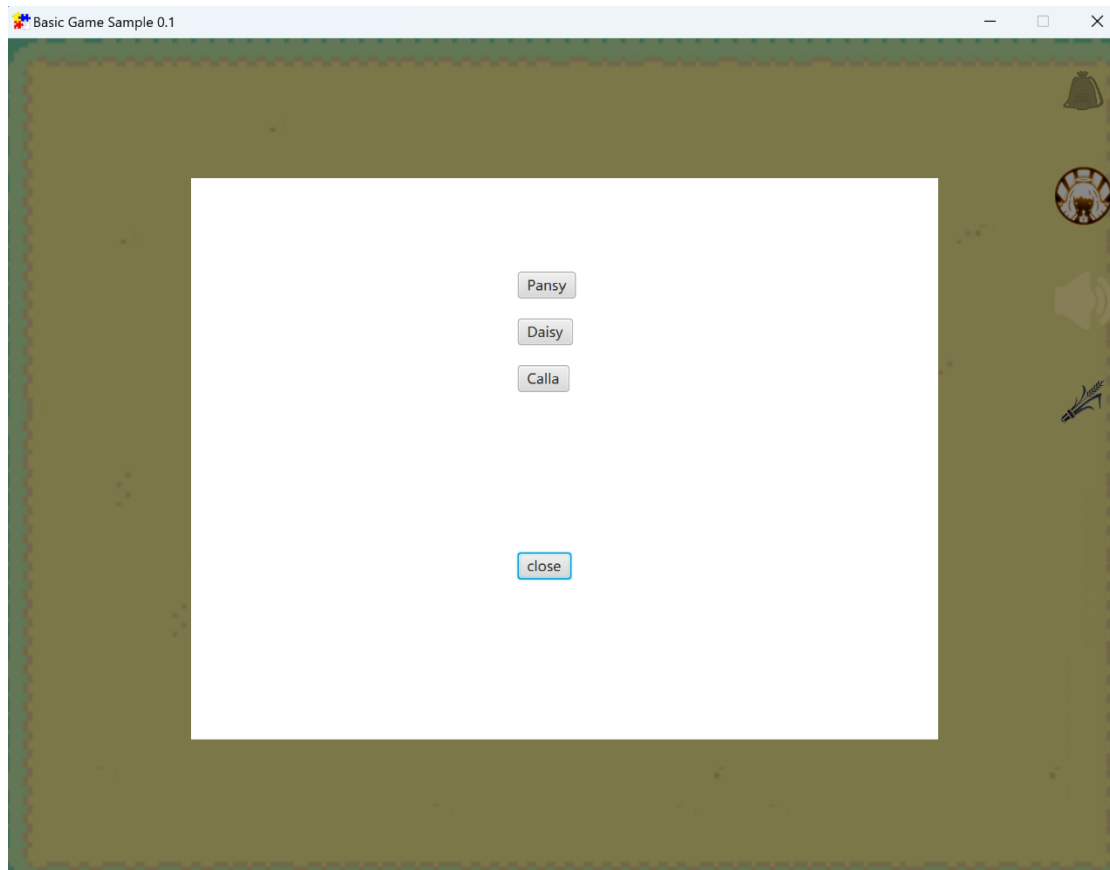


点击叹号 ui，弹出收获子窗体。

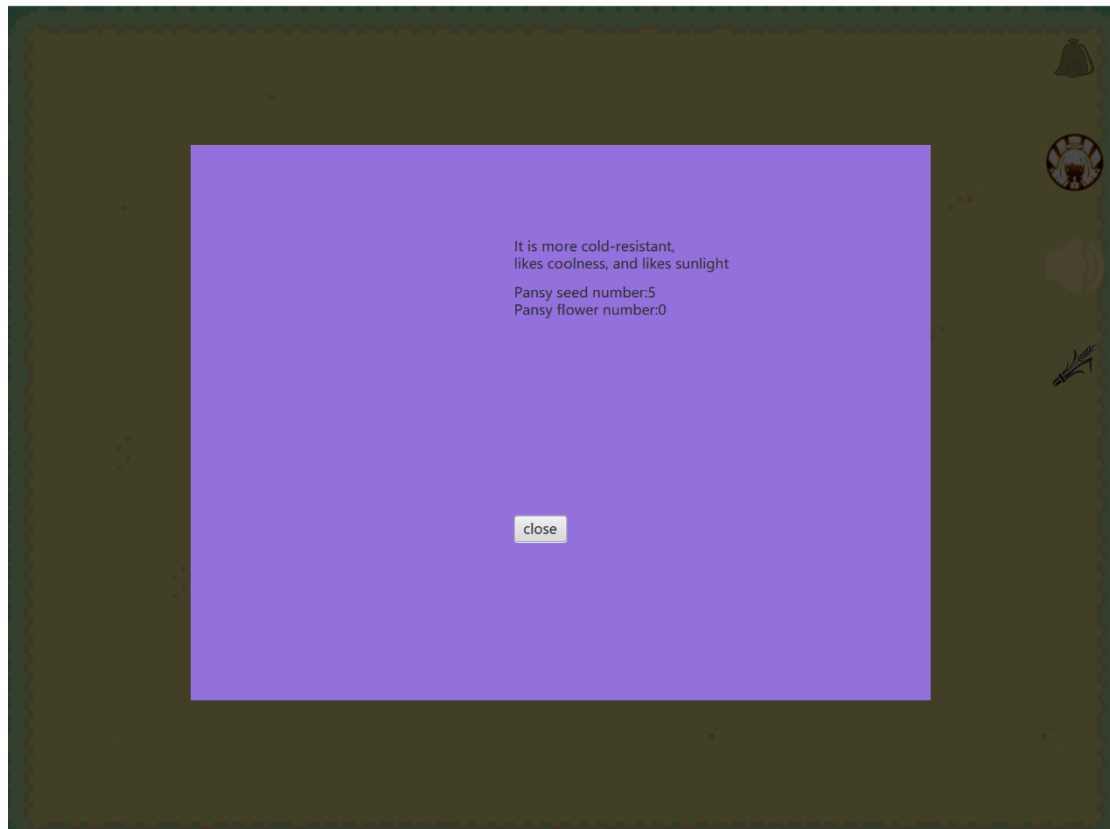


点击收获，对应花卉被收获，背包内对应花卉数目增加相应数量。

(iiii) 查看信息

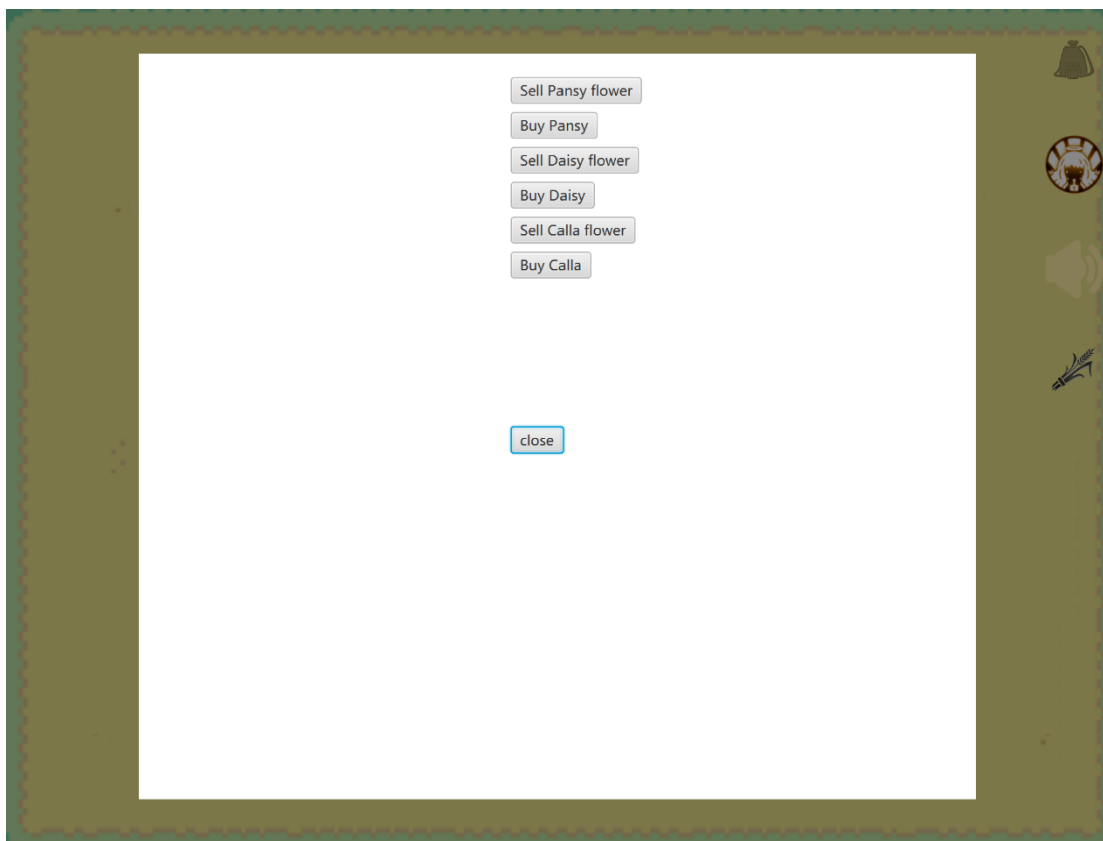


分别点击右上角背包 ui 按钮和人物 ui 按钮，弹出背包子窗体和人物身份子窗体。



背包子窗体内点击不同按钮，弹出含有不同花卉信息和背包内对应种子和花朵数目的子窗体。

(iiii) 交易

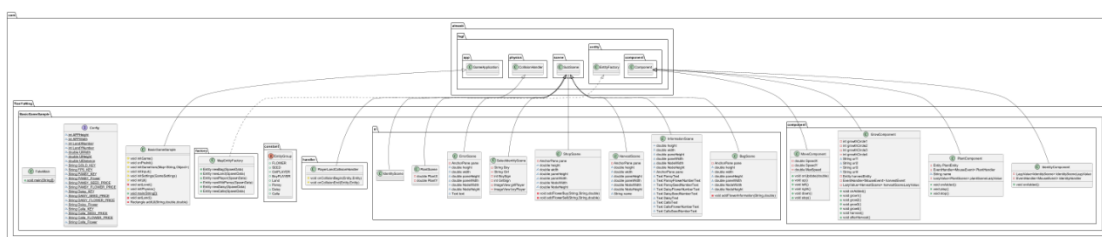


点击右上角交易 ui，弹出交易子窗体。点击不同按钮，对应不同种花卉的交易，所拥有钱币的数目和花卉、花卉种子的数目信息会发生对应改变。（如果拥有的花卉数目为零仍点击卖出该种花卉按钮或者拥有的钱币数目不足购买仍点击购买按钮，会弹出错误子窗体）。

## 4 程序面向对象实现技术方案

程序共容纳一个 jar 包。

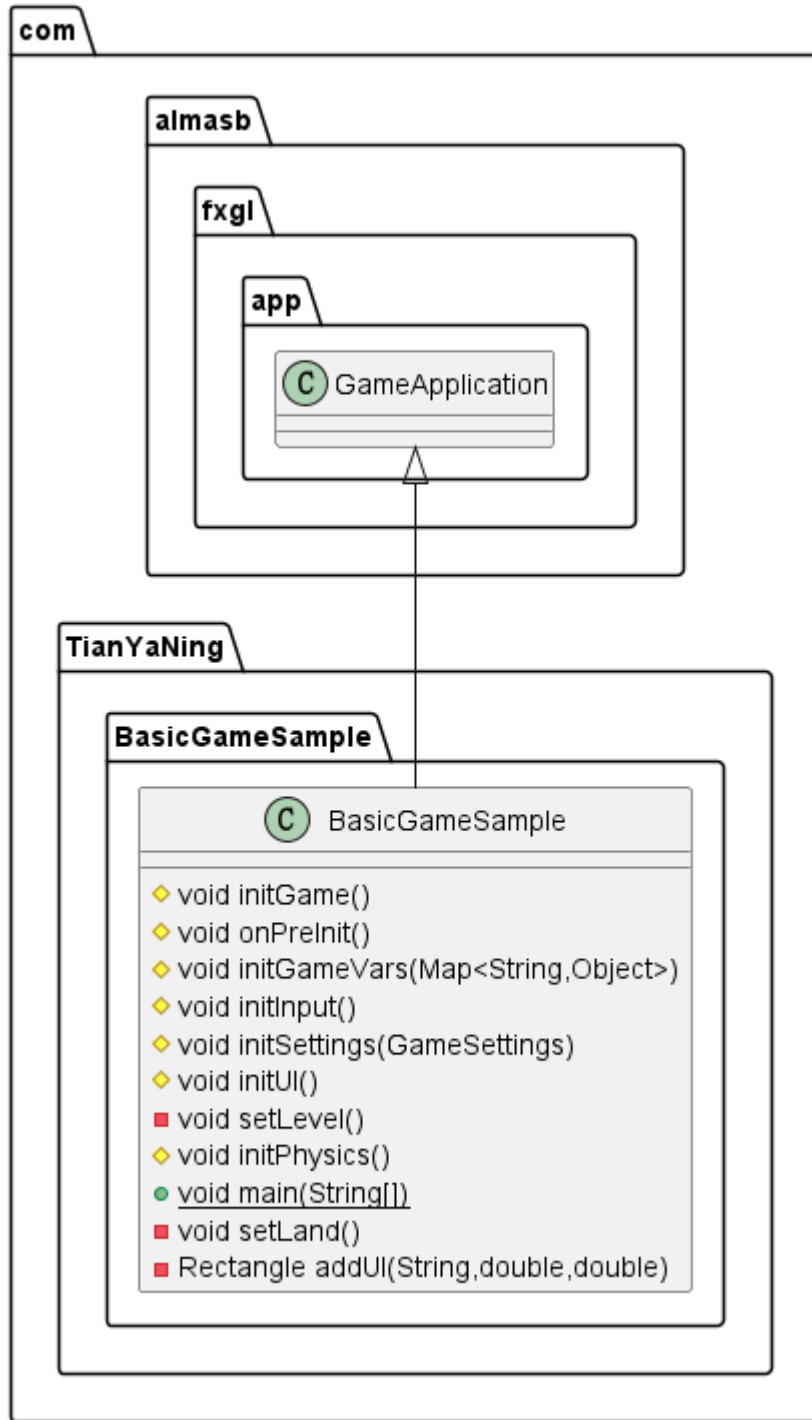
程序整体由 PlantUML 插件生成的 UML 图如下。



其中，各个包中程序的 UML 图及解释如下。

### （1）BasicGame

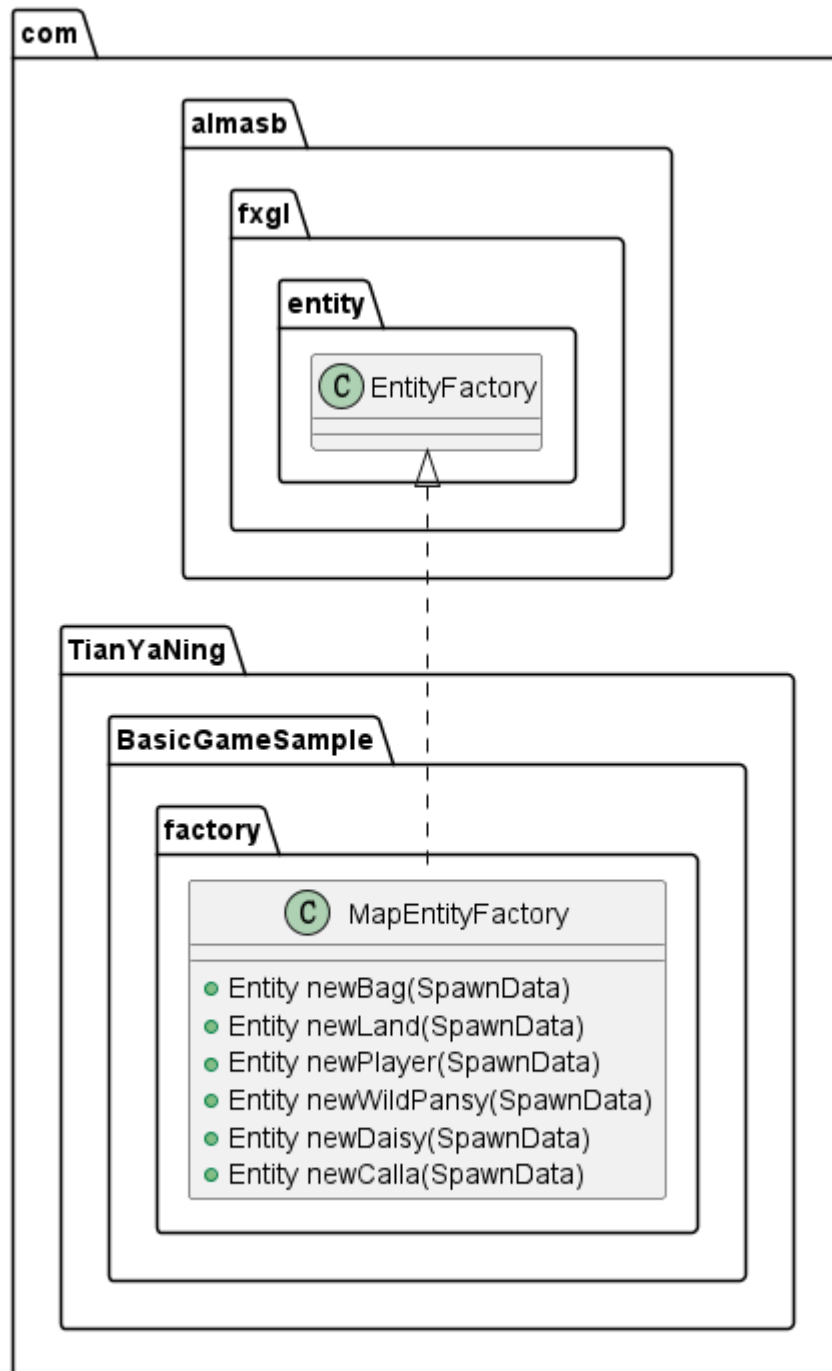




此为包中的主类，BasicGameSample 继承自 fxgl 包中的 GameApplication，用于显示游戏主窗体；其中的 setlevel()方法用于设置地图，setLand()方法用于在地图上加入 Land 实体，addUI()方法用于加入 ui 标志。

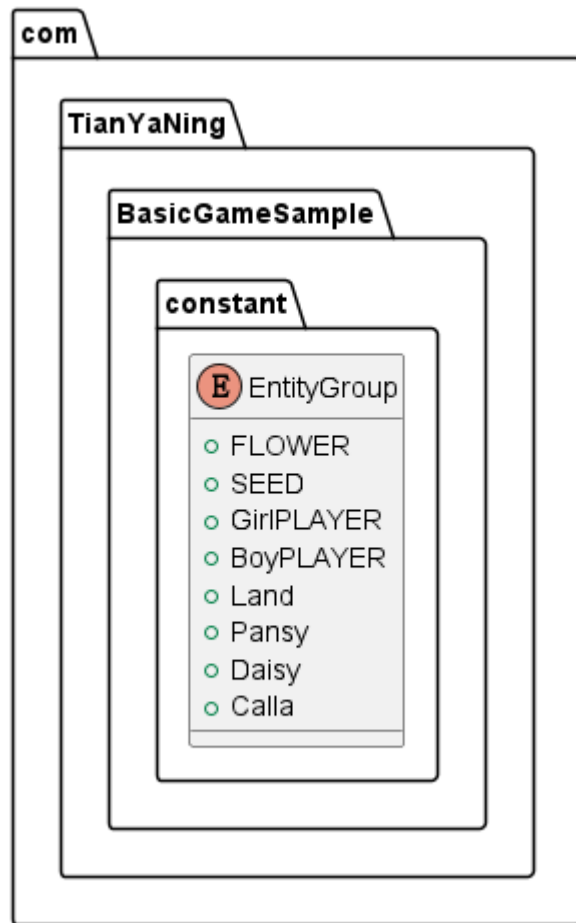
重写的方法中，特别说明 initGameVars()方法，其中的全局变量储存玩家的信息，包括钱币数量、种子和花卉价格、种子和花卉数目等。

## (2) factory



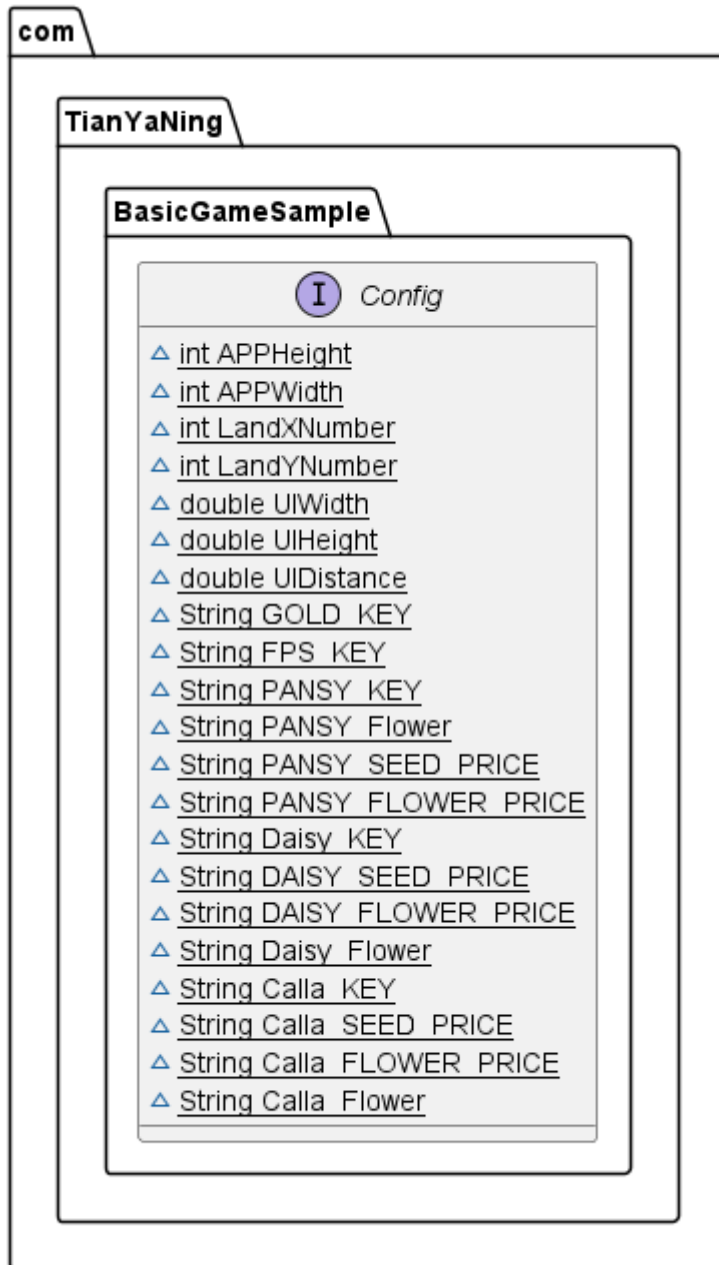
**Factory** 中包含 FXGL 游戏中的主要实体及其信息（包括组合的组件、对应的图片等），用于构建游戏中的实体。实体主要包含玩家实体、土地实体、花卉实体三类，三者进行交互。

### （3）entityGroup



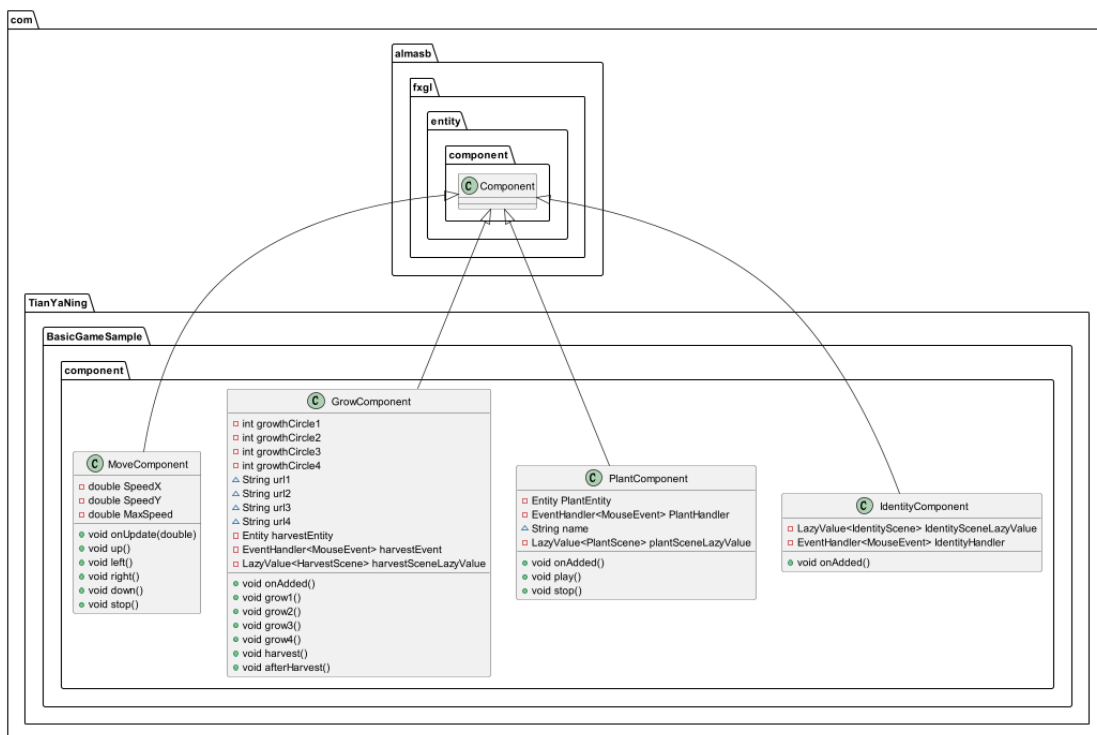
EntityGroup 枚举类中按 FXGL 要求保存了 entity 的 type。

(4) Config



为方便修改程序代码和使用全局变量，Config 接口保存需使用的全局变量名称和各类主窗体信息。

#### (5) Component



程序各类功能的实现主要依靠组合 component。

Component 包中的各类均继承自 Component，可被实体组合实现功能。

#### (i) MoveComponent

玩家实体组合此移动组件，实现移动功能。组件中包含上下左右移动和停止方法。

#### (ii) GrowComponent

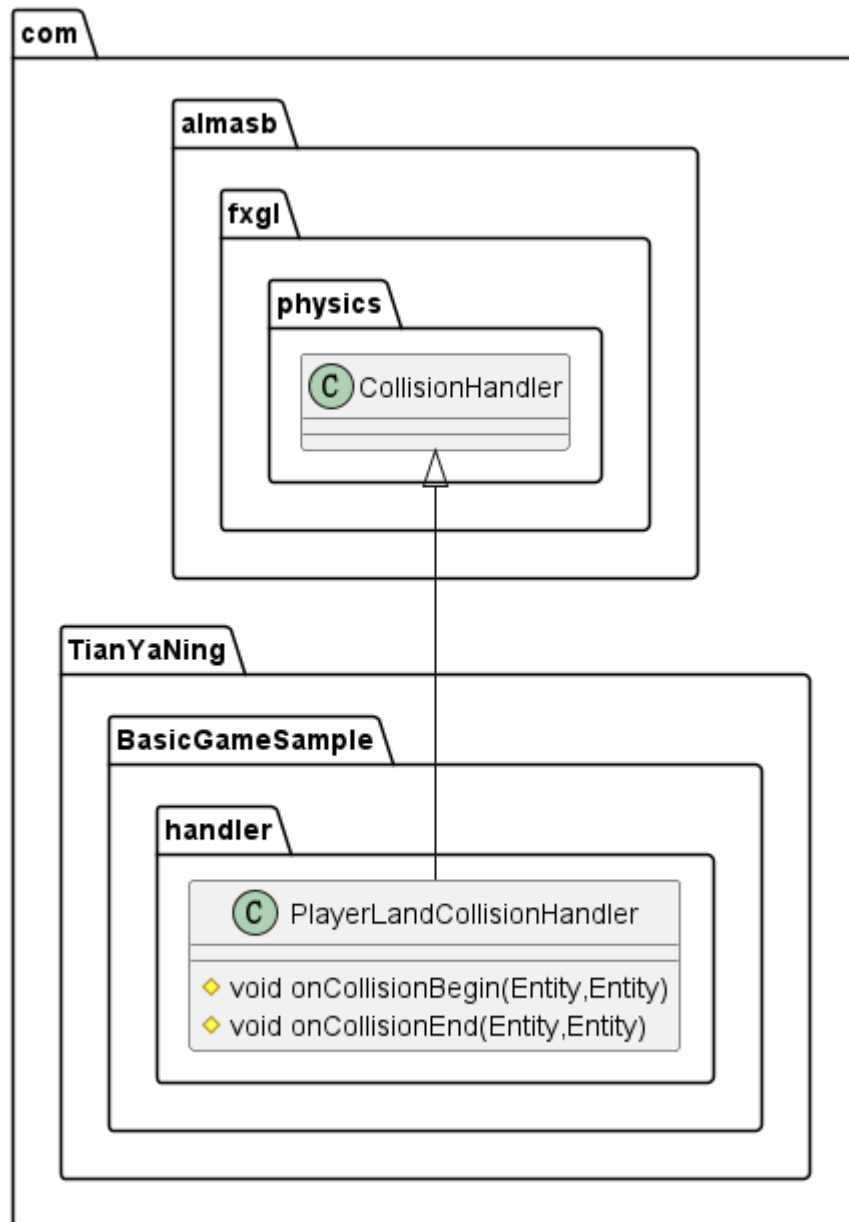
花卉实体组合此生长组件，实现生长功能。不同花卉组件传入不同信息调用不同的构造方法，实现不同生长周期花卉的生长。

收获功能也被部分包含于此组件中。

#### (iii) PlantComponent

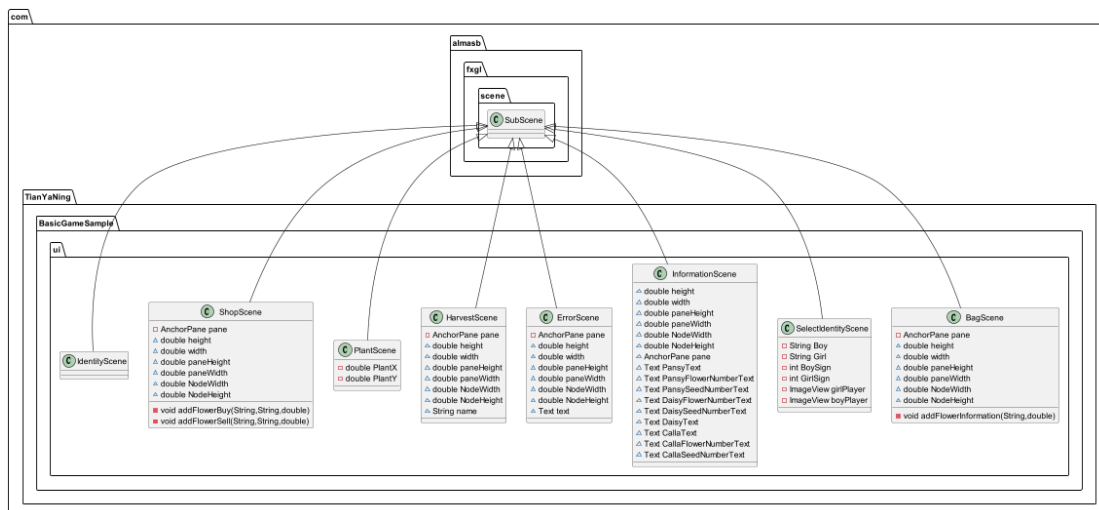
土地组件组合此种植组件，实现种植功能。当土地与玩家发生碰撞时，显现种植 ui。

#### (6) handler



此 handler 包中的类继承自 fxgl 中的 CollisionHandler 类，用于实现不同实体间的碰撞。  
此 PlayerLandCollisionHandler 实现了玩家和土地的碰撞，用于种植功能的实现。

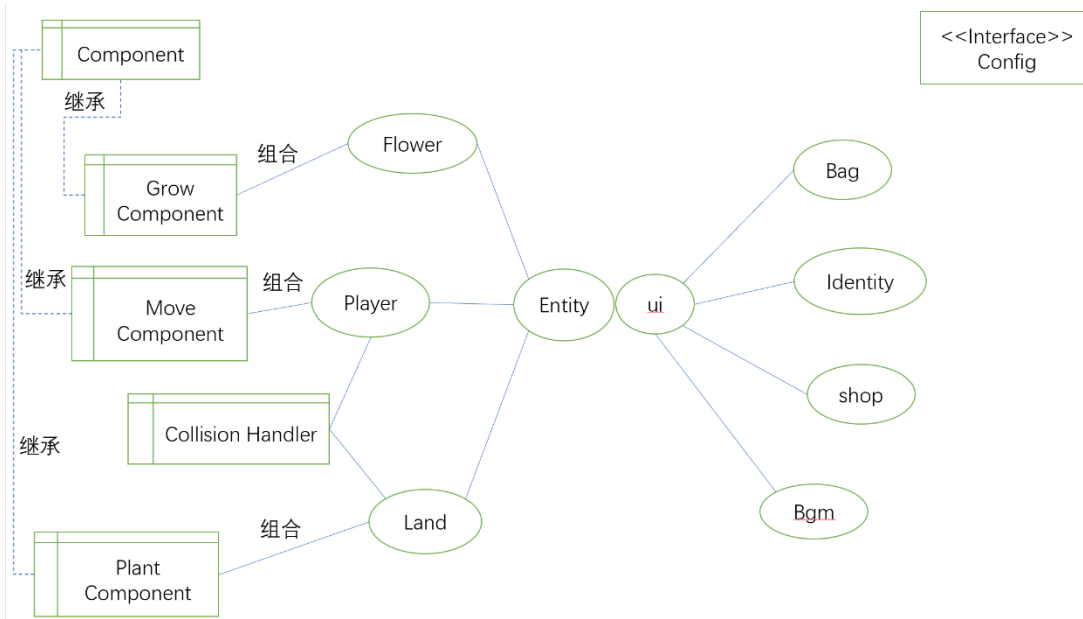
(7) ui/Subscene



Subscene 包中包含的类均继承自 fxgl 中的 Subscene 类,用于实现子窗体的构建。其中,强调一下各窗体显示的数目信息均与 fxgl 全局变量池中的变量进行绑定,能够随之变化。

以上即为 jar 包中各包的组织与功能。

简略的示意图如下。



## 5 技术亮点、关键点及其解决方案

### ● 本程序的亮点:

程序整体实现了模拟经营游戏功能,功能完善。  
程序利用 FXGL 游戏开发框架,开发代码简洁明确。  
程序结构清晰,功能设计方案简洁明了。  
游戏画面明丽自然,配乐相合,玩家体验感良好。

### ● 本程序的技术关键点

使用 FXGL 游戏开发开源框架进行编写  
设计游戏基本功能的技术实现方案  
提升玩家与游戏交互体验


使用 GitHub 来管理源码

- 遇到的技术难点及对应的解决方案：

(1) 文件加载图片问题

**问题描述：**当使用 FXGL 载入图片时，编译器警告无法在 `assets/texture` 文件夹中找到导入的图片

[社区首页](#) > [问答首页](#) > [无法从FXGL中的资源加载纹理](#)

**问** 无法从FXGL中的资源加载纹理  腾讯云小微

EN >

Stack Overflow用户 提问于 2021-01-30 01:25:46

回答 1

查看 167

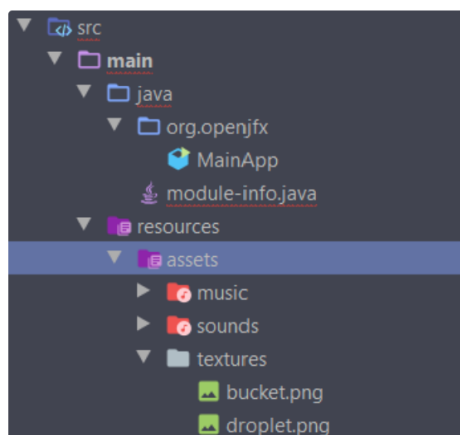
关注 0

票数 0

我想为一个属于我的大学的项目制作一个2D游戏。所以我决定使用JavaFX和FXGL库。我从FXGL Git repository获得了一个test basic项目，并尝试运行它。当我运行这个项目时，FXGL警告说它无法加载纹理 `bucket.png`。以下是完整的日志消息：

```
19:54:36.901 [FXGL Background Thread 1 ] WARN  FXGLAssetLoaderServi - Failed to load texture bucket.png Error: java.lang.IllegalArgumentException: Asset "/assets/textures/bucket.png" was not found!
```

下面是我的项目结构：



**最终的解决方案：**将 `module-info` 中的 `module` 改为 `open module`

github 开发文档中首页介绍页面中包含此介绍，惭愧，刚开始阅读的时候先跑去读后面的内容，没意识到，查找到问题花费了挺长时间

## 模块化

如果您希望开发模块化应用程序，这里有一个完整的示例 `module-info.java`：

```
open module app.name {
    requires com.almasb.fxgl.all;
}
```

(2) 线程运行问题

**问题描述：**当 `new` 一个子场景时，显示某个线程的运行出错，后经检查，发现是创建此 `Subscene` 时 `fx` 工具箱还未初始化。



```

20:41:04.441 [JavaFX Application Thread] FATAL FXGLApplication - Uncaught Exception:
java.lang.RuntimeException Create breakpoint : Initialization failed
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.FXGLApplication$InitAppTask.failed(FXGLApplication.kt:343)
    at javafx.graphics/javafx.concurrent.Task.setState(Task.java:710)
    at javafx.graphics/javafx.concurrent.Task$TaskCallable.lambda$call$2(Task.java:1456)
    at javafx.graphics/com.sun.javafx.application.PlatformImpl.lambda$runLater$10(PlatformImpl.java:457) <1 个内部行>
    at javafx.graphics/com.sun.javafx.application.PlatformImpl.lambda$runLater$11(PlatformImpl.java:456)
    at javafx.graphics/com.sun.glass.ui.InvokeLaterDispatcher$Future.run(InvokeLaterDispatcher.java:96)
    at javafx.base/com.sun.glass.ui.win.WinApplication._runLoop(Native Method)
    at javafx.graphics/com.sun.glass.ui.win.WinApplication.lambda$runLoop$3(WinApplication.java:184) <1 个内部行>
Caused by: java.lang.IllegalStateException Create breakpoint : Not on FX application thread; currentThread = FXGL Background Thread 4
    at javafx.graphics/com.sun.javafx.tk.Toolkit.checkFxUserThread(Toolkit.java:297)
    at javafx.graphics/com.sun.javafx.tk.quantum.QuantumToolkit.checkFxUserThread(QuantumToolkit.java:458)
    at javafx.graphics/javafx.scene.Parent$3.onProposedChange(Parent.java:474)
    at javafx.base/com.sun.javafx.collections.VetoableListDecorator.add(VetoableListDecorator.java:205)
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.MainWindow.pushState(MainWindow.kt:165)
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.FXGLApplication$GameApplicationService.pushSubScene(FXGLApplication.kt:711)
    at com.TianYaNing.BasicGameSample/com.TianYaNing.BasicGameSample.ui.IdentityScene.showIdentity(IdentityScene.java:50)
    at com.TianYaNing.BasicGameSample/com.TianYaNing.BasicGameSample.BasicGameSample.initGame(BasicGameSample.java:98)
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.FXGLApplication$InitAppTask.initGame(FXGLApplication.kt:339)
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.FXGLApplication$InitAppTask.call(FXGLApplication.kt:319)
    at com.almasb.fxgl.all@17.2/com.almasb.fxgl.app.FXGLApplication$InitAppTask.call(FXGLApplication.kt:311)
    at javafx.graphics/javafx.concurrent.Task$TaskCallable.call(Task.java:1426) <5 个内部行>
    ... 1 more

20:41:04.443 [JavaFX Application Thread] FATAL FXGLApplication - Application will now exit
20:41:05.245 [FXGL Background Thread 2 ] INFO UpdaterService - Your current version: 17.2
20:41:05.245 [FXGL Background Thread 2 ] INFO UpdaterService - Latest stable version: 17.3

```

## 最终的解决方案:

### 使用 lazyValue

模块 java.desktop  
软件包 javax.swing

#### Interface UIDefaults.LazyValue

所有已知实现类:  
UIDefaults.LazyInputMap, UIDefaults.ProxyLazyValue

Enclosing class:  
UIDefaults

```

public static interface UIDefaults.LazyValue

```

此类允许在默认表中存储一个条目，该条目在第一次使用getXXX(key)方法之一进行查找之前不会构建。延迟值对于构造昂贵或很少检索的默认值很有用。一个在第一时间LazyValue被取回它的“真实值”是通过调用计算LazyValue.createValue()与真实值被用来代替LazyValue在UIDefaults表。对相同键的后续查找返回实际值。下面是一个示例LazyValue即构造一个border：

```

Object borderLazyValue = new UIDefaults.LazyValue() {
    Object createValue(UIDefaults table) {
        return new BorderFactory.createLoweredBevelBorder();
    }
};

uidefaultsTable.put("MyBorder", borderLazyValue);

```

另请参见:  
UIDefaults.get(java.lang.Object)

1 个用法

```

private LazyValue<HarvestScene> harvestSceneLazyValue = new LazyValue<>(() -> {
    return new HarvestScene(entity);
});

```

0 个用法

```

@Override
public void onAdded() {
    harvestEvent = new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent mouseEvent) {
            FXGL.getSceneService().pushSubScene(harvestSceneLazyValue.get());
        }
    };
}

```

### (3) 多线程使用问题

**问题描述:** 在实现生长功能时想要用多线程实现不同生长周期植物的变化，但是一直无

法显示。

9 个用法

```
public class GrowComponent extends Component {  
    1 个用法  
    private final long growthCircle1=1;  
    1 个用法  
    private final int growthCircle2=1;  
    1 个用法  
    private LazyValue<Thread> growLazyValue=new LazyValue<>(()->{  
        Runnable runnable =new Runnable() {  
            @SneakyThrows  
            @Override  
            public void run() {  
                TimeUnit.SECONDS.sleep(growthCircle1);  
                grow1();  
                TimeUnit.SECONDS.sleep(growthCircle2);  
                grow2();  
            }  
        };  
        return new Thread(runnable);  
    });
```

0 个用法

```
public void onAdded() {  
    Thread thread =growLazyValue.get();  
    thread.start();  
}
```

1 个用法

```
u:\java\bin\java.exe -XX  
11:29:45.723 [JavaFX Application Thread] INFO Engine - FXGL-17.2 (20.08.2022 16:50) on WINDOWS (J:17.0.8 FX:17.0.6)  
11:29:45.724 [JavaFX Application Thread] INFO Engine - Source code and latest versions at: https://github.com/AlmasB/FXGL  
11:29:45.724 [JavaFX Application Thread] INFO Engine - Ask questions and discuss at: https://github.com/AlmasB/FXGL/discussions  
11:29:45.724 [JavaFX Application Thread] INFO Engine - Join the FXGL chat at: https://gitter.im/AlmasB/FXGL  
11:29:46.384 [FXGL Background Thread 1 ] INFO FXGLApplication - FXGL initialization took: 0.336 sec  
11:29:46.513 [FXGL Background Thread 4 ] WARN TMXLevelLoader - TiledMap generated from 1.10.2. Supported version: 1.9.0. Some features may not be parsed full  
11:29:46.629 [FXGL Background Thread 4 ] INFO FXGLApplication - Game initialization took: 0.178 sec  
11:29:48.634 [FXGL Background Thread 2 ] INFO UpdaterService - Your current version: 17.2  
11:29:48.634 [FXGL Background Thread 2 ] INFO UpdaterService - Latest stable version: 17.3  
11:29:52.200 [Thread-4 ] FATAL FXGLApplication - Uncaught Exception:  
java.lang.IllegalStateException Create breakpoint: : Not on FX application thread; currentThread = Thread-4  
    at javafx.graphics/com.sun.javafx.tk.Toolkit.checkFxUserThread(Toolkit.java:297)  
    at javafx.graphics/com.sun.javafx.tk.quantum.QuantumToolkit.checkFxUserThread(QuantumToolkit.java:458)  
    at javafx.graphics/javafx.scene.Parent$3.onProposedChange(Parent.java:474)  
    at javafx.base/com.sun.javafx.collections.VetoableListDecorator.clear(VetoableListDecorator.java:293)  
    at com.almasb.fxgl.entity@17.2/com.almasb.fxgl.entity.components.ViewComponent.clearChildren(ViewComponent.kt:241)  
    at com.TianYaNing.BasicGameSample/com.TianYaNing.BasicGameSample.component.GrowComponent.grow1(GrowComponent.java:45)  
    at com.TianYaNing.BasicGameSample/com.TianYaNing.BasicGameSample.component.GrowComponent$1.run(GrowComponent.java:27) <1 个内部行>  
  
11:29:52.200 [Thread-4 ] FATAL FXGLApplication - Application will now exit  
  
Exception: java.lang.IllegalStateException thrown from the UncaughtExceptionHandler in thread "Thread-4"
```

最终的解决方案：查阅 FXGL 开发文档，发现了合适的替代实现，改为使用其中的

getGameTimer 功能来实现生长功能，代码更加简洁。

In games you often want to run an action after some period of time or at some interval. For example, you can use the following to schedule some action to run only once after 1 second. The action will run on the JavaFX Thread.

```
getGameTimer().runOnceAfter(() -> {  
    // code to run once after 1 second  
}, Duration.seconds(1));
```

Note the above is equivalent to FXGL DSL:

```
import static com.almasb.fxgl.dsl.FXGL.*;  
  
runOnce(() -> {  
    // code to run once after 1 second  
}, Duration.seconds(1))
```

Similar DSL functions are available for other timer actions. If you want something to run constantly, you can use:

```
getGameTimer().runAtInterval(() -> {  
    // code to run every 300 ms  
}, Duration.millis(300));
```

```
getGameTimer().runOnceAfter(()->{  
    grow1();  
},Duration.seconds(growthCircle1));  
getGameTimer().runOnceAfter(()->{  
    grow2();  
},Duration.seconds(v: growthCircle1+growthCircle2));
```

## 6 简要开发过程

- 12 月 1 号 查找 FXGL 资料，阅读开发文档
- 12 月 7 号 用八位元玩家完成游戏形象设计
- 12 月 13 号 用 mapedit 完成地图制作并载入 intellij
- 12 月 14 号 进行登录及形象选择功能开发
- 12 月 15 号 进行种植功能开发
- 12 月 16 号 进行生长功能开发
- 12 月 17 号 进行背包功能开发
- 12 月 18 日，进行商店功能开发。
- 12 月 19 日，进行界面 ui 装载
- 12 月 20 日，进行窗体构建
- 12 月 24 号 功能开发完成，进入测试阶段
- 12 月 25 号 对程序进行集成测试
- 12 月 28 号 程序开发工作完毕，编写及整理文档

## 7 个人小结及建议

本学期学习 Java 语言程序设计课程，感悟良多。

首先是体会到学习一门语言的关键在于自己亲自实践上手写。之前没有什么接触 Java 语言的经验，我最开始学习的时候主要是以做笔记的方式学习记忆知识，结果发现笔记记得

很多很累还没什么效果，不如自己亲自上手写几行代码演练一下记忆得清晰，而且这样对各个内容的理解也更深入。

然后就是发现使用 **Java** 的关键在于读开发文档和源代码。因为结课设计我想要用 **FXGL** 开发框架写一个小游戏。最开始我是从网上查各种资料起步的，先去看了 **b** 站上几个介绍游戏开发的视频和 **CSDN** 上的经验分享，没有先看开发文档，结果最初明明写的代码都对，图片无论如何都加载不出来。**IntelliJ** 的提示就只是找不到该图片，感觉特别疑惑迷茫，自己放的文件夹位置明明也是按 **fxgl** 的架构来的，最后找来找去，终于找到开发文档首页介绍，如果使用模块化编程，需要把 **module-info** 中的 **module** 改为 **public module**，改完之后，一切运行顺利。之后出现的其他问题，我也发现基本可以用开发文档中和 **github** 上开发者给出的建议进行解决，这也让我很深地体会到开发文档和源码的重要性。

最后，进行结课设计的开发也给了我很多的乐趣。虽然，在结课设计过程中，因为我之前没怎么接触过 **Java** 编程，学习 **FXGL** 花费了很多时间（比我想象得多得多的时间），整个编程过程很累，有时候遇到问题，进度推不下去的时候还特别痛苦想要放弃，最后做出的游戏程序虽然基本实现了当初构想的功能，但是还不是特别的精致和完美（感觉尤其是各个子窗体，还可以再改善很多，但是实在没有时间了）。但是，在这个过程中，看到自己设想的功能一点点实现，问题一个个解决，一个个小程序组合起来实现整个功能，让我感到特别的满足，让我真正地体会到了程序开发的乐趣。

总而言之，非常感谢这一学期老师干货满满的教导，非常感谢这门课程，也非常感谢这一学期虽然经常觉得一点听不懂课跟不上进度但是仍然课下坚持编程的自己，在 12 月末忙碌的考试周敲下这段话，当做送给新年的一份温暖和一份礼物！