C++项目报告

————桌游《骰子街》的代码实现

学院：中欧工程技术学院

成员：王泓杰、娄宇鑫、王楚涵、李鑫宇、郑益

2022.11

目录

一、 项目概述

《骰子街》是一款简单又有趣的经营类桌游，适合 2-4 人使用，利用骰子和货币来建造具有各种功能的建筑，最先完成四样地标性建筑的玩家获得胜利，单局游戏时长为 20-30 分钟。

我们小组使用 c++编程语言，结合 Gitee 平台创建自己的私有仓库，采用分布式合作的方式，在 Clion 编译器中完成了《骰子街》线上应用实现，包括游戏版本的选择，各种普通建筑以及地标建筑的功能和属性设置；玩家所拥有的建筑，货币等属性；电脑玩家的生成与设定；人机交互页面等基础功能。

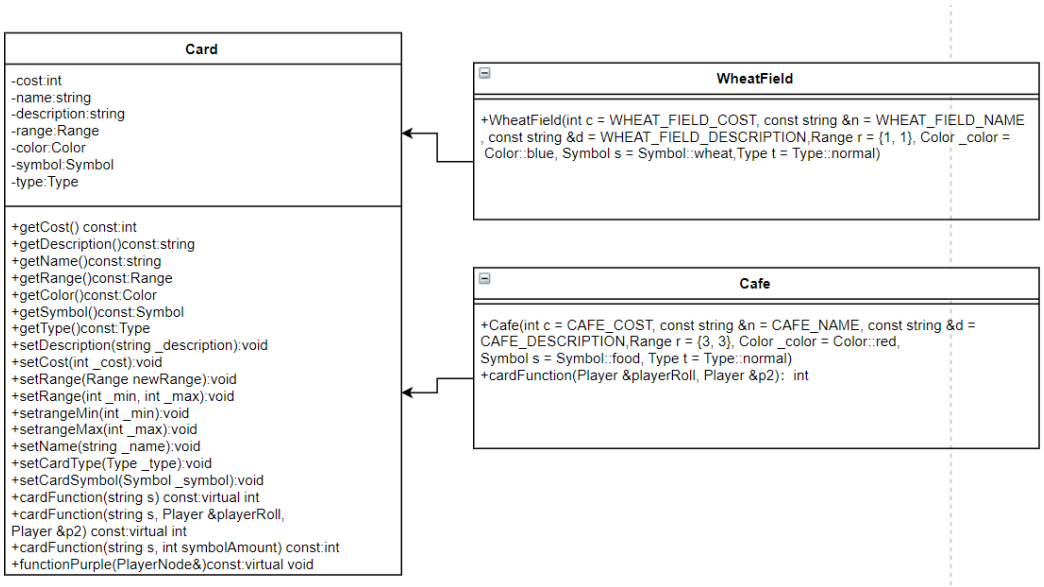
在以上模块的基础上初步实现了《骰子街》的正常运行，玩家可以在程序桌面 Desk 上进行操作进行游戏，满足条件的玩家将由系统自动判定获胜。另外设计了拓展了港口版本的卡包和玩法，玩家可以自由选择玩法，使用不同的功能卡包。

二、 项目流程

1. 小组成员试玩桌游，熟悉玩法和规则
2. 策划游戏运行顺序，构思游戏框架（UML 图），统一函数和变量名称
3. 划分功能模块，讨论大致实现方法并实现分工，统一模块接口
4. 分工完成各自模块功能的实现
5. 整合功能，搭建游戏界面
6. 代码的查漏补缺和优化

三、 功能模块

1.Card

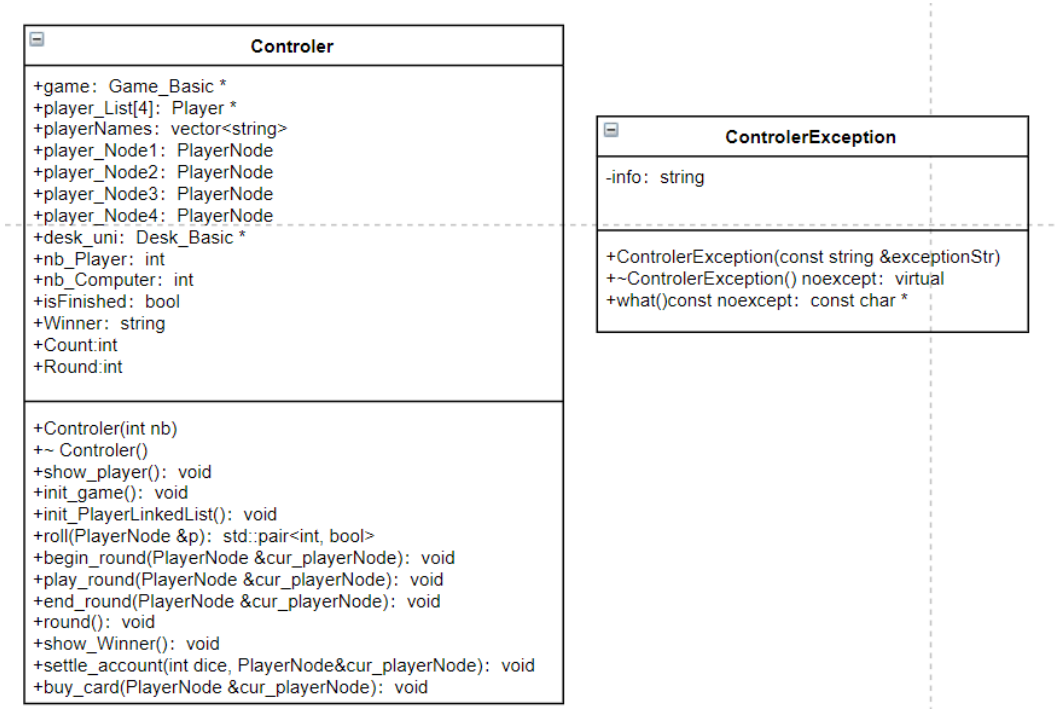


Card 父类中定义了所有卡共有的的基本属性变量：名字，价值，颜色（蓝卡、红卡、绿卡、紫卡），种类（普通卡，特殊卡，工厂卡），符号（农业，畜牧，商业，餐饮，工业等），功能描述，Range 结构体（骰子生效范围）等。

Card 的方法定义了每张卡的属性获取函数（get），属性设置函数（set）和功能实现函数（virtual cardFunction（普通卡功能），virtual functionPurple（紫卡功能））。使用每张卡的功能时都使用同名接口 cardFunction 函数，利用传递的参数类型的不同，分别调用重载过的函数。紫卡等功能区别较大的卡使用虚函数在每张子类卡中重新定义。

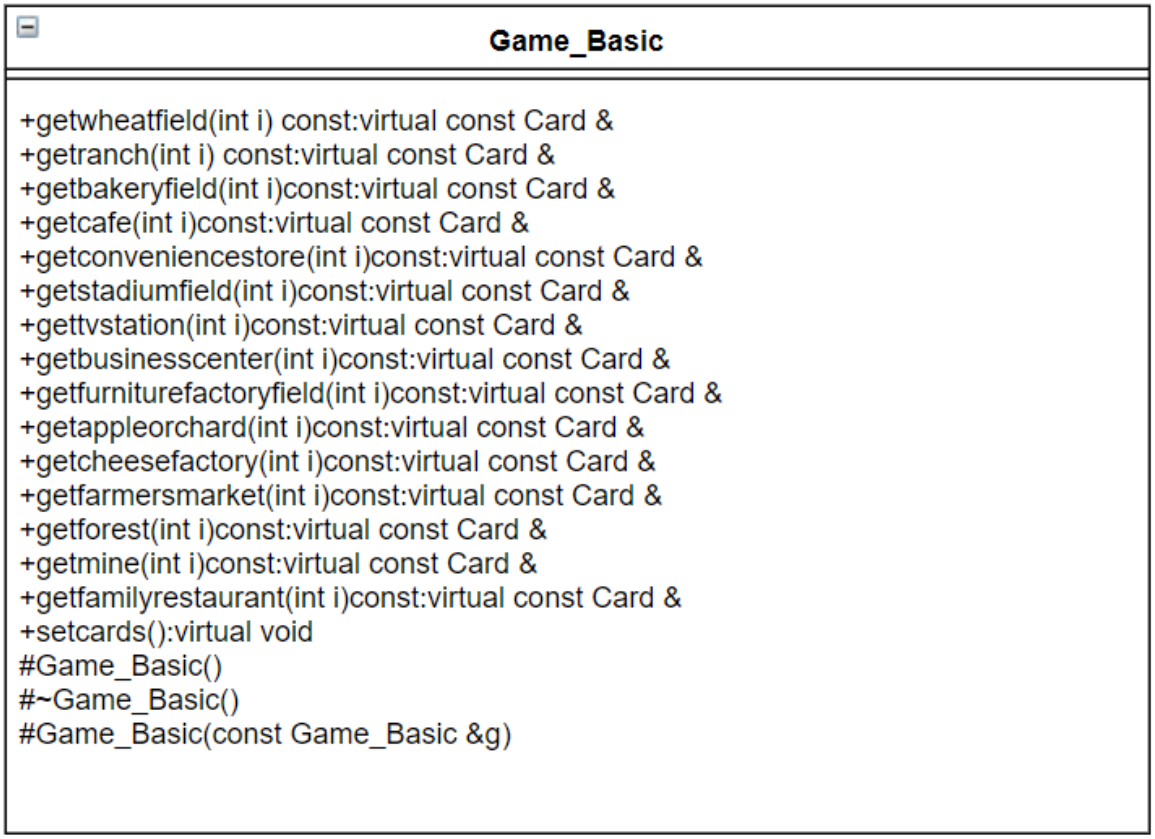
所有具体功能卡（例如麦田 WheatField，农场 Ranch 等）作为 Card 的子类（如右图）调用 Card 的构造函数设置属性，WheatField 类包含重写的父类虚函数（例如功能生效函数 cardFunction）。

2.Controler



Controler 作为整个游戏的“管理者”，其所拥有的各个指针成员变量负责了 Player, Desk, Game 类的生命周期，其所包含的所有函数都是对整个游戏的操纵，例如：初始化游戏内容，settle_account 结算金币，round()进行游玩，买卡等。Init_game():负责初始化各成员指针变量，其中调用了 Version 类的静态成员函数。Init_PlayerLinkedList()：利用链表数据结构将 4 位玩家结点收尾相连，这样在负责进行游玩的 round()函数中，只需一直对链表体->next（直到游戏结束）就可以完成玩家游玩顺序的循环

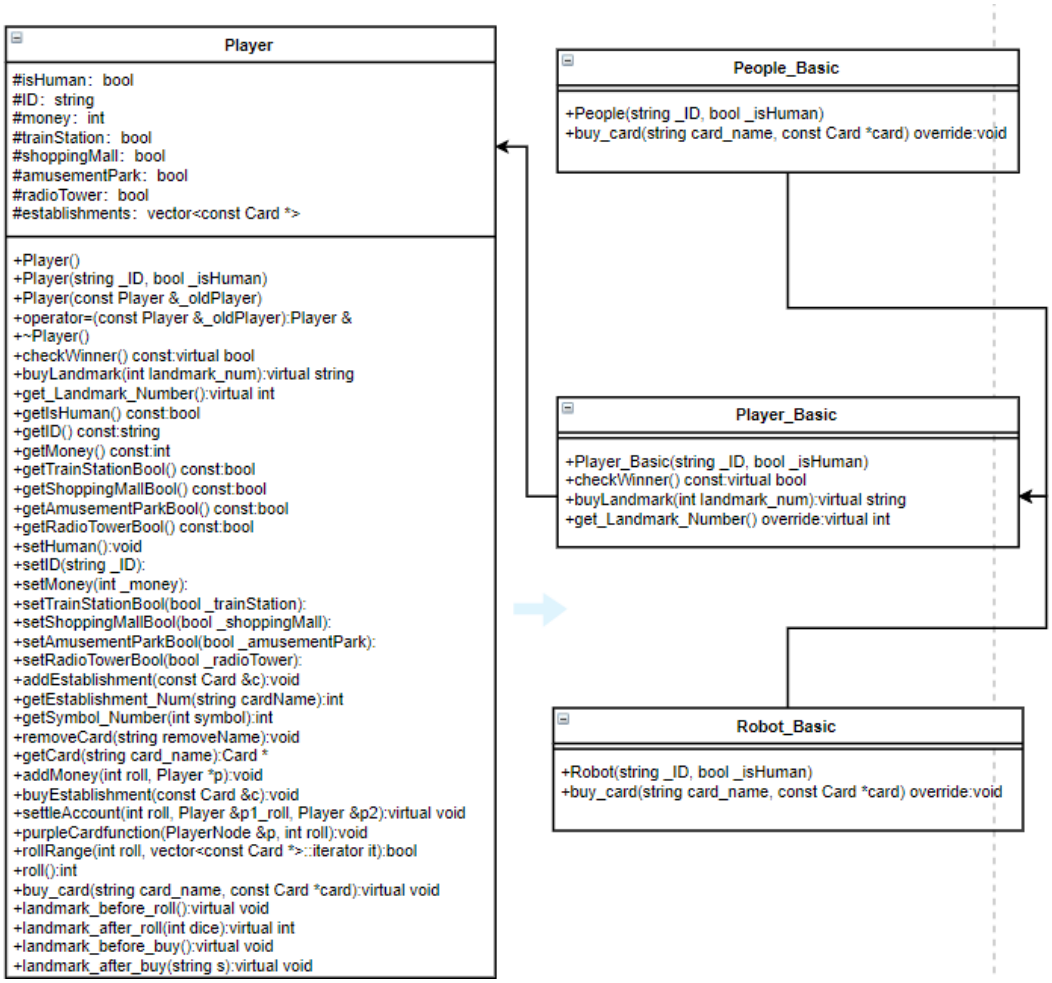
其中 Controler(int n) 是单参构造函数，传入的参数 n 决定了版本的选择，这里会结合 Version 类中的静态成员函数进行版本的设置。



3.Game

Game_Basic 类用来生成、储存和返回卡片。所有卡牌作为其成员变量，并提供 getcard 函数，返回对应的卡牌。setcard()为卡牌生成函数，生成所有卡牌，并写为虚函数。

4.Player



Player 抽象父类定义了玩家的，ID，金币，是否为人类，拥有的卡的数组等基础属性。同时我们将地标类卡定义为 bool 类型存储在 Player 中。我们将地标卡理解为需要玩家花钱解锁的特殊技能，而不是实体的卡，因为：

1.地标卡是玩家开局就拥有的，并不需要从商店购买，只需要付钱解锁。

2.地标卡的功能通常非常特殊，例如影响扔骰子（基础版中的火车站、游乐园、广播塔）、判定各类卡是否结算（港口扩展中的港口地标）等

3.当地标卡涉及到结算金币时，往往与普通卡的结算金币环节无关，而是在一个新的环节，（例如：港口扩展中的飞机场地标，当玩家没有购买新的建筑时增加 10 金币，市政厅地标：当玩家在购买建筑环节前没有金币，则提供 1 金币）。

4.方便新的版本扩展时直接继承新的地标，只需要在 Player 子类定义新的地标 bool 属性。

Player 的方法除了基本的 get，以及 set 函数外，主要设计了判定胜利者（checkWinner），结算金币（settleAccount），购买卡（buy_card），移除卡（removeCard）等功能函数，以及其他的辅助函数。

我们认为机器人玩家和人类玩家行为差别在于，机器人玩家会自动买卡，只需要告诉外界：他购买了哪张。而人类玩家需要在控制台交互，选择想要购买的卡。所以，我们重写了购买卡的函数（buy_card），使其在玩家的类型不同时有不一样的交互。

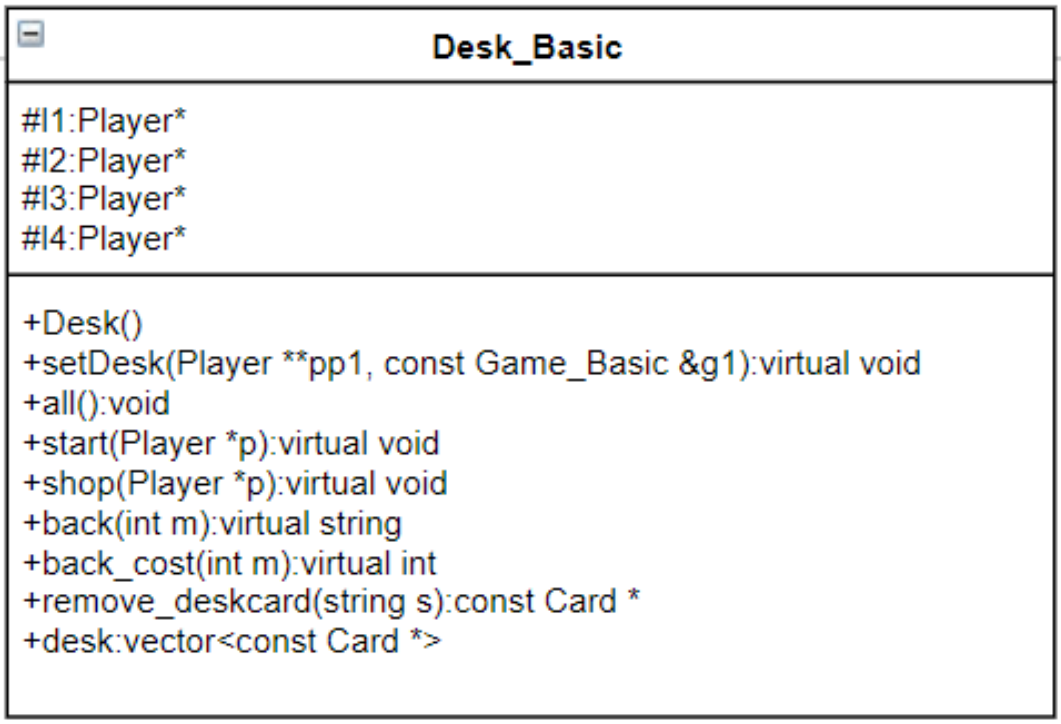
机器人玩家的买卡逻辑相对欠缺很好的智能性，但仍有一定的策略。它会着重选择购买与农业相关的卡组，以及在能够解锁地标时及时解锁地标。不过这个策略也是经过我们实际测试后，胜率较高的一种玩法（我们小组的郑益同学在游玩时往往采用这种策略）。

根据不同版本，版本选择功能会生成不同的子类 Player_Basic 或 Player_Port，在其中添加了获取新地标 bool 状态的 get 和 set 函数，以及重写了判定胜利者函数（virtual checkWinner），购买地标函数（virtual buyLandmark），这两个函数都紧密联系于地标的种类（地标的种类的增

加是一个新版本的重要特征)。

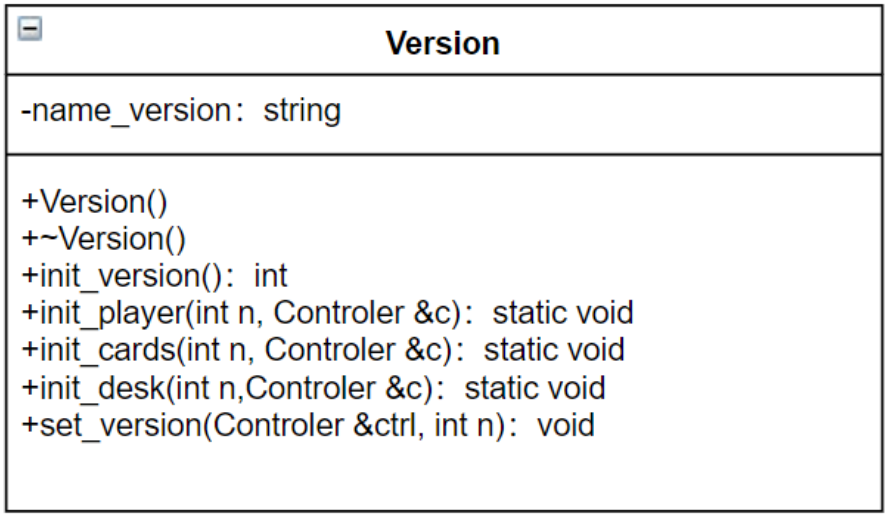
基于对版本扩展以及新增地标的触发范围的理解，我们在不同版本重写了结算函数（virtual settleAccount)，以及地标触发的位置函数（landmark_before_roll，landmark_after_roll，landmark_before_buy，landmark_after_buy)并在 controler 中调用，这也是我们小组在增加新的版本时，不需要生成 controler 子类的原因。

5.Desk



Desk_Basic 类用于与用户交互，将玩家和这把游戏用到的牌传入到 Desk_Basic 中。Desk_Basic 将每回合的各个玩家的所拥有的卡和商店里有的卡进行打印，back 将数字与卡片名字对应，方便顾客买卡。desk 容器用来存储这局游戏里用到的牌。当玩家买卡时，从 desk 容器中将卡片删除即可。

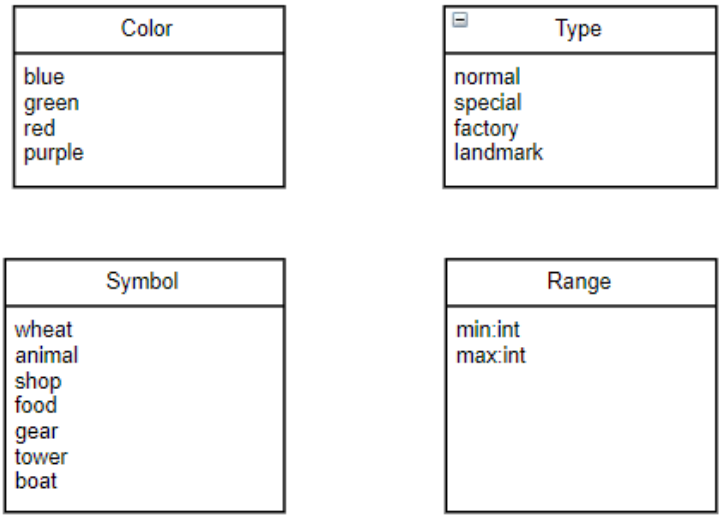
6.Version



Version 是 Controler 的友元类，其三个静态函数（便于在 Controler 类中 init_game()函数中用 Version::使用，不需要生成 Version 类的实例）可以访问 Controler 的内部成员指针变量 PlayerList 数组(Player*),Game*和 Desk*，从而实现版本的更改。

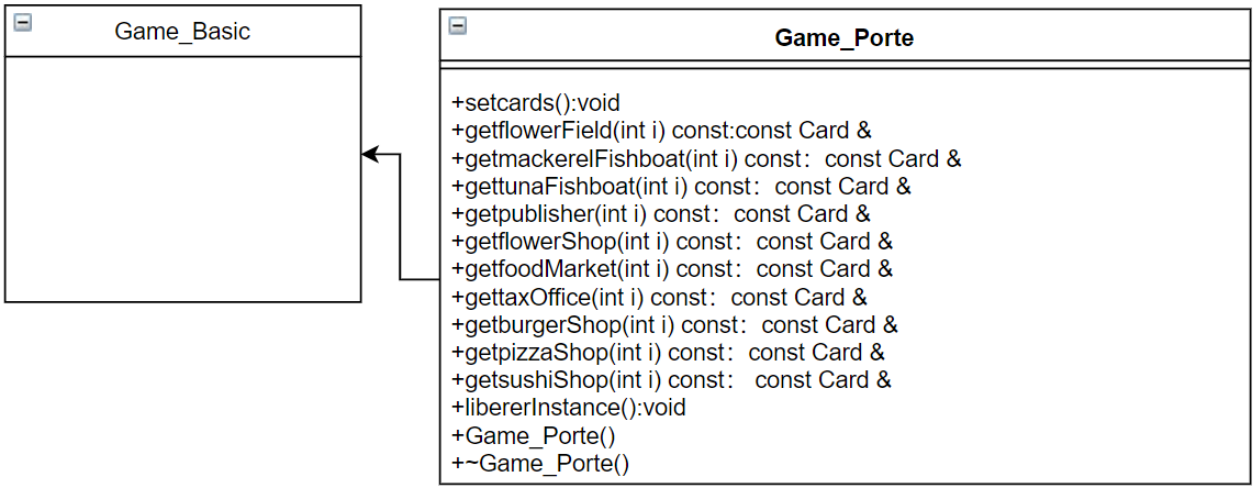
更改版本的关键在于利用 dynamic_cast 将 Controler 中各成员指针变量的父类指针转换成子类指针，然后再分别调用 Desk, Game 的 setter 函数，具体实现在代码中有详细解释。

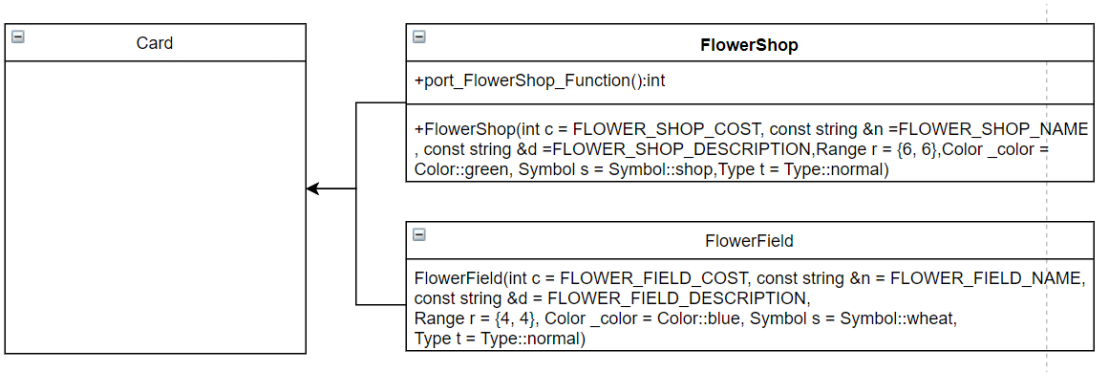
7. Constant



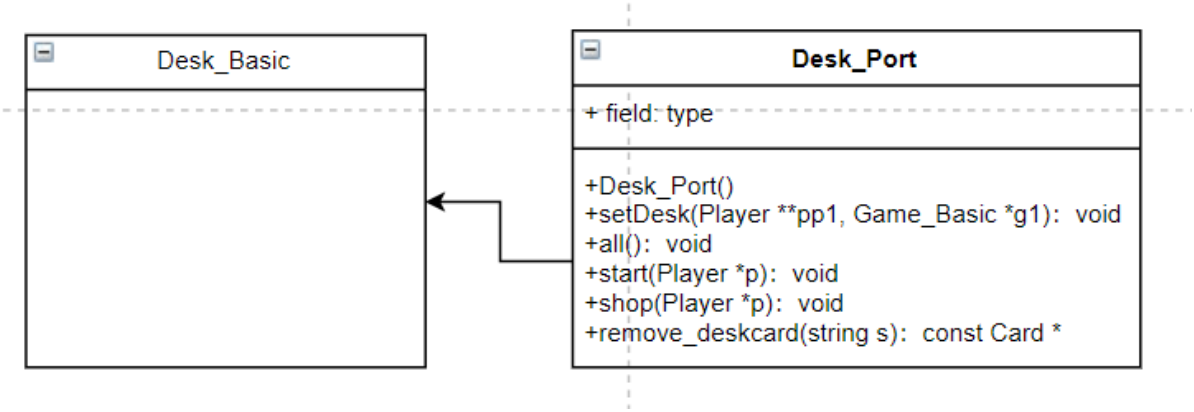
在 constant 中自定义了 enum 结构的 Color，Type，Symbol，Range 用于储存卡片的颜色，类型，功能符号和作用范围等性质，方便后续程序中对卡片性质的定义和使用

8. 港口拓展版





Game_Porte 类继承于 Game_Basic,拥有 Game_Basic 里所有的卡，并额外拥有港口拓展包的卡，同时 setcard()函数也继承于 Game_Basic，额外生成港口卡片。如果后续还需加入其它版本，新的 Game 类就可以再继承 Game_Basic 类，然后往新的 Game 类里添加新的卡片，并重载 setcard()函数，再加入新的 getcard()函数。而对于新的 card，继承 Card 类即可。

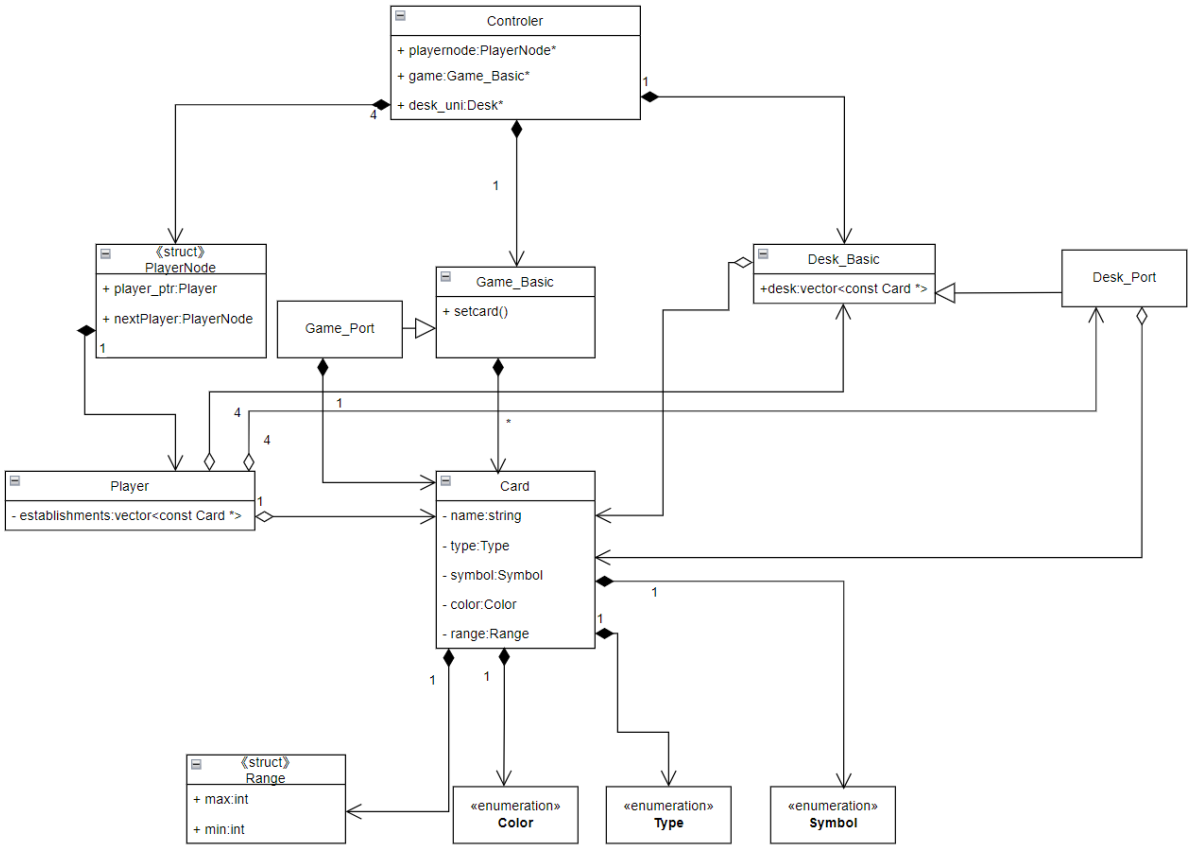


Desk_Port 继承 Desk_Basic 类，用于进行拓展包的桌面交互，为了能够适配拓展包内容，back 函数重写将新增的卡片与数字对应打印玩家信息的 start 函数需进行重写打印新的地标，shop 函数因为新的卡片的新增也要进行重写，以可以打印新的卡。牌桌将拓展包的卡牌存到其中的容器，其余功能保持不变

如果还有新的版本只需要创建新的类 Desk_another 继承 Desk 类，只需要将 start，shop，back，back_cout 进行重写，将新的卡牌新的地

标的打印出来，其他功能全都保持不变即可正常使用

四、 UML 图



五、 功能特点

1. 较好地运用了面向对象思想，将现实玩桌游按照各个对象进行分解，分为各个功能模块（例如玩家，卡牌，牌桌，假想的游戏“管理员“，假象的版本选择者），从而全部完成了法方老师提出的所有基本要求。

2. 将所有卡牌封装在固定的 Card 类型中，并且使用虚函数定义卡牌功能，使用继承的方法一方面确保卡牌基本结构一致，一方面可以在每个子类当中重新定义功能函数

3. 版本选择的优化层面，合理利用了父类和子类指针的上下行转换关系拓展了港口功能和新的功能卡包。由玩家自行选择游戏版本，增加游戏

的可玩性

4. 合理利用了链表数据结构进行玩家循环游玩，加入了电脑玩家，增加游戏趣味性和实用度

5.拓展了港口功能和新的功能卡包，由玩家自行选择游戏版本，增加游戏的可玩性

六、 改进方向

1.人机交互页面还需优化，使用 Visual Studio 等页面构造工具使得用户体验更佳，甚至可以制作精美的 UI 界面。

2.扩展版本中可以购买的卡有所改动，因时间原因还未实现

3.卡片可以在页面上进行显示，可以制作渲染牌桌和买卡动画

4.电脑玩家的游戏逻辑可以进行优化，使其更接近人类玩家的玩法

5.改进代码效率，优化游戏运行内存

6.每个模块文件有清晰的功能特点，便于程序员理解和修改

七、 会议记录

10.15——确定各个文件及其功能，完成文件分工，确定要定义的类，画出类之间的 UML 图

10.17——完成 Card.h 定义，完成单例设计模式，生成牌堆，在 constant.h 中确定各卡牌的属性，开启 desk，player，controler 的定义

10.23——完成除紫色卡外的各卡牌功能函数，定义 player 的获取卡牌函数，完成赢家结算函数，完成 desk 的定义，继续编写 controler

10.27——完成 Card.cpp 紫色卡牌功能，完成 desk 和 controler 之间的接

口，完成 desk 和 player 之间的接口，继续编写 player

11.5——完成 desk 文件，完成游戏流程框架，定义 robot 玩家，完成 player.cpp 中的玩家操作函数，完成骰子函数 dice

11.12——完成地标卡的功能函数，完成 robot 玩家的游戏逻辑，完成 main 函数，完成析构函数，检查完善各文件之间的接口

11.17——进行游戏完整操作试验，检查修改 bug，优化代码，并且确定港口拓展版的卡包和游戏实现方式

11.20——完成拓展卡包的定义，功能函数和各接口

11.23——完成拓展版的基本功能，对代码查漏补缺，进行整体优化，编写项目报告，完成项目演示视频

八、 个人贡献

姓名	学号	负责内容	时长及贡献度
王泓杰	20124770	主编文件: Controler.h Controler.cpp class Controler class Version 功能实现: 版本选择，游戏管理，商业中心紫卡功能 其他工作: 代码调试及 bug 修复 部分报告撰写 基础版演示视频剪辑制作	40h 20%

娄宇鑫	20124772	主编文件: Player_1.0.h Player_1.0.cpp class Player 父类 Player 的各个子类 功能实现: 港口扩展卡牌的实现 其他工作: Gitee 仓库搭建 港口演示视频制作 代码调试及 bug 修复 部分报告撰写	40h 20%
王楚涵	20124771	主编文件: Card.h Card.cpp Card_2.0.h Card_2.0.cpp 功能实现: Card 的生成，储存和调用 其他工作: 部分紫卡和地表卡的功能实现 UML 图 交互流程图	32h 20%
李鑫宇	20124769	主编文件: Controler.h&Controler.cpp class Controler class Version	

		desk_1.0.h desk_2.0.h desk_1.0.cpp desk_2.0.cpp 功能实现: Desk 的生成和调用 显示界面的设计	30h 20%
郑益	20124768	主编文件: Card_2.0.h Card_2.0.cpp 功能实现: 拓展建筑卡功能 玩家获取建筑 其他工作: 项目报告编写	32h 20%