2021-11-7

C++ 报告

课程报告

璀璨宝石

SUN HUDIE 、JIA ZHENGWEI、WANG LEER、CHEN NINGNING

目　录

一、项目概况

我们成功设计出了 Splendor 游戏的简略版本，并完成了项目中的大部分要求，能够实行 2-4 名玩家通过对操作台的控制，自由进行对抗，项目引用了单例模式,保证 class ToutesLesCartesNobiliaires 只能产生一个实例，并面向整个系统提供；同时，我们引入了牌库，以此保证每张发展卡及其相应声望值、宝石数、宝石红利的数据流的稳定性；最后，我们为程序进行了拓展，加入了 AI 玩家，并解锁了城市面板模式的新玩法。整个程序可以实现三种玩法：人机对抗、玩家对抗以及 AI 对抗，两种模式：普通模式和城市面板模式，都可以由玩家自行选择。最后，可以对游戏实现存档。

稍有瑕疵的是，目前游戏的交互界面还较为粗糙，卡牌的显示不够人性化。

二、项目流程

1.小组成员组织进行桌游初体验

2.熟悉桌游之后构想程序的大致框架（所要实现的功能对应的类，小组分块进行编程）

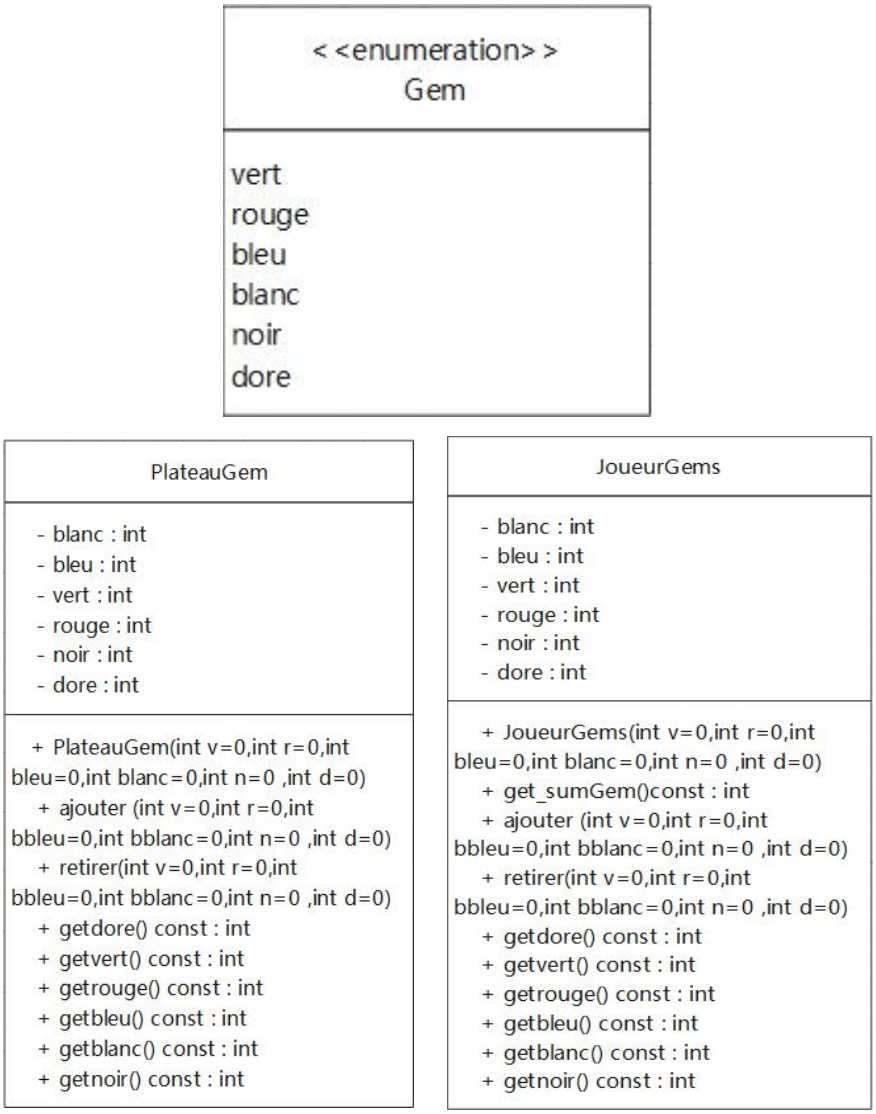
3.画出每个类的类图，实现类中性质和方法的声明

4.整合小组编程内容，并修补程序的漏洞

5.实现游戏代码运行

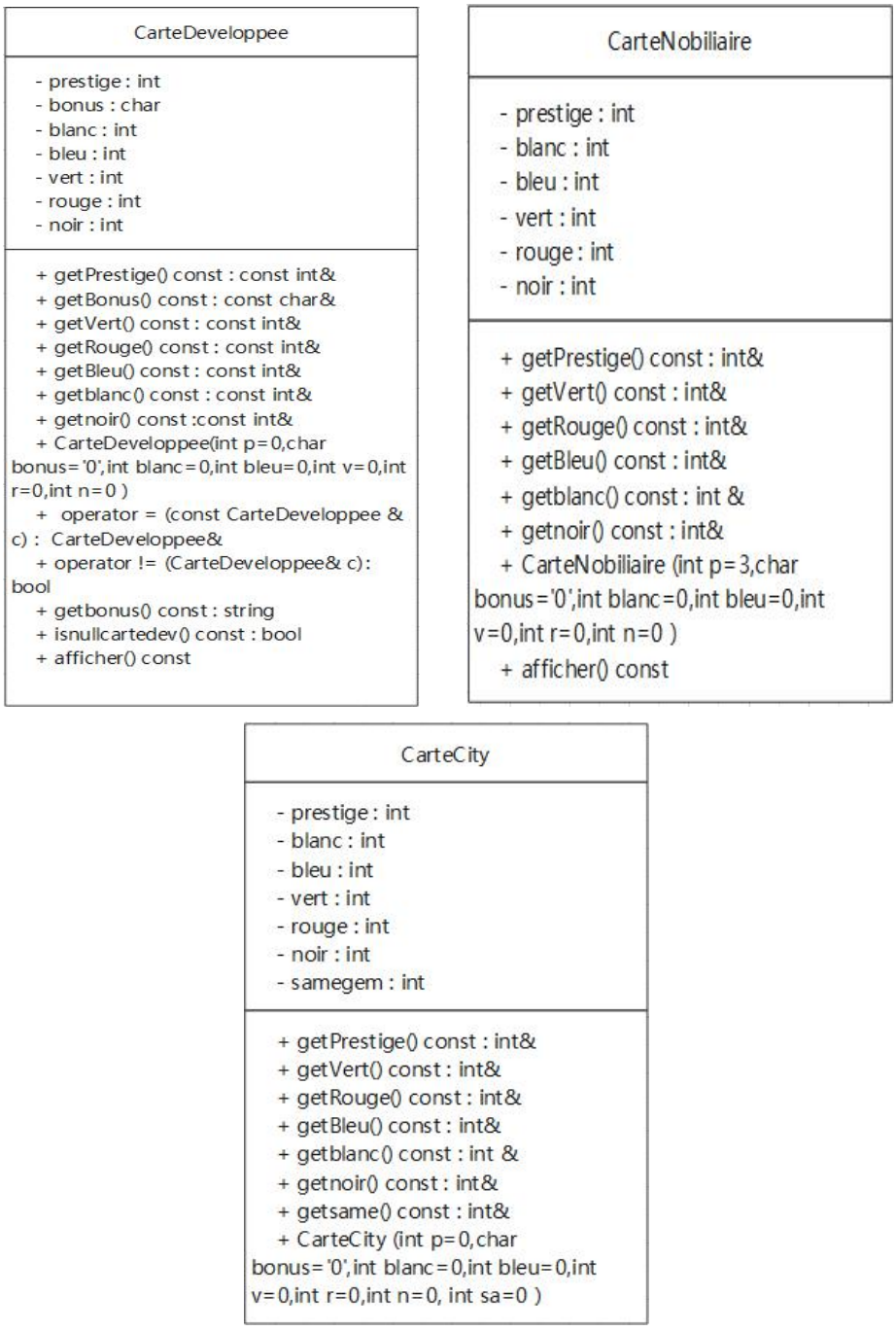
三、代码实现

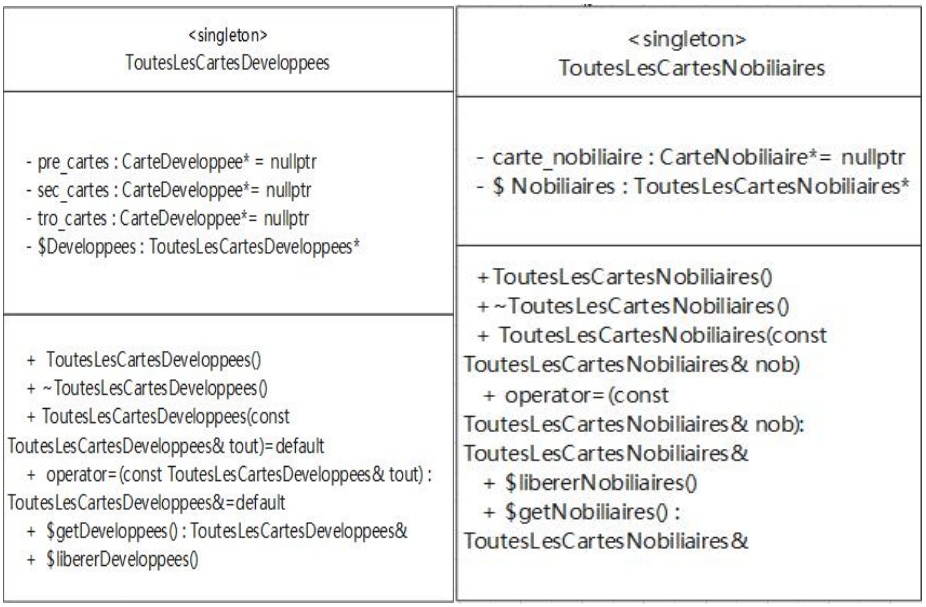
1.Gem

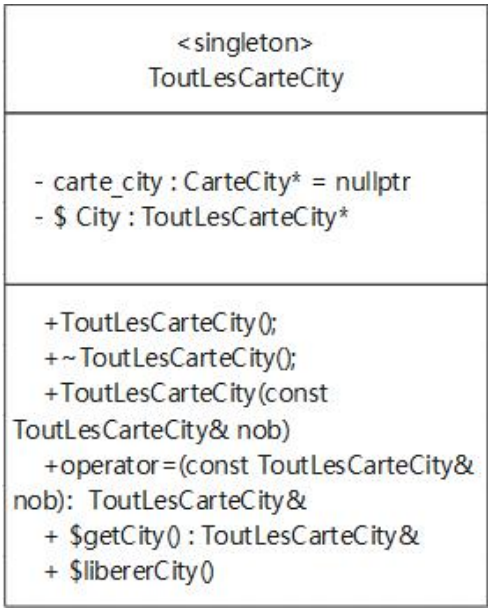


命名为 Gem 的文件中定义了宝石类型（包括黄金的定义），对【宝石】的生命周期进行负责，PlateauGem 类中记录了桌面宝石数量的动态变化，JoueurGems 类对玩家拥有宝石数量和类型进行了封装，记录了玩家拥有宝石数量的动态变化。

2. Cartes


CarteDeveloppee
- prestige : int
- bonus : char
- blanc : int
- bleu : int
- vert : int
- rouge : int
- noir : int
+ getPrestige() const : const int&
+ getBonus() const : const char&
+ getVert() const : const int&
+ getRouge() const : const int&
+ getBleu() const : const int&
+ getblanc() const : const int&
+ getnoir() const :const int&
+ CarteDeveloppee(int p=0,char bonus='0',int blanc=0,int bleu=0,int v=0,int r=0,int n=0)
+ operator = (const CarteDeveloppee & c) : CarteDeveloppee&
+ operator != (CarteDeveloppee& c): bool
+ getbonus() const : string
+ isnullcartedev() const : bool
+ afficher() const
CarteNobiliaire
- prestige : int
- blanc : int
- bleu : int
- vert : int
- rouge : int
- noir : int
+ getPrestige() const : int&
+ getVert() const : int&
+ getRouge() const : int&
+ getBleu() const : int&
+ getblanc() const : int &
+ getnoir() const : int&
+ CarteNobiliaire (int p=3,char bonus='0',int blanc=0,int bleu=0,int v=0,int r=0,int n=0)
+ afficher() const
CarteCity
- prestige : int
- blanc : int
- bleu : int
- vert : int
- rouge : int
- noir : int
- samegem : int
+ getPrestige() const : int&
+ getVert() const : int&
+ getRouge() const : int&
+ getBleu() const : int&
+ getblanc() const : int &
+ getnoir() const : int&
+ getsame() const : int&
+ CarteCity (int p=0,char bonus='0',int blanc=0,int bleu=0,int v=0,int r=0,int n=0, int sa=0)



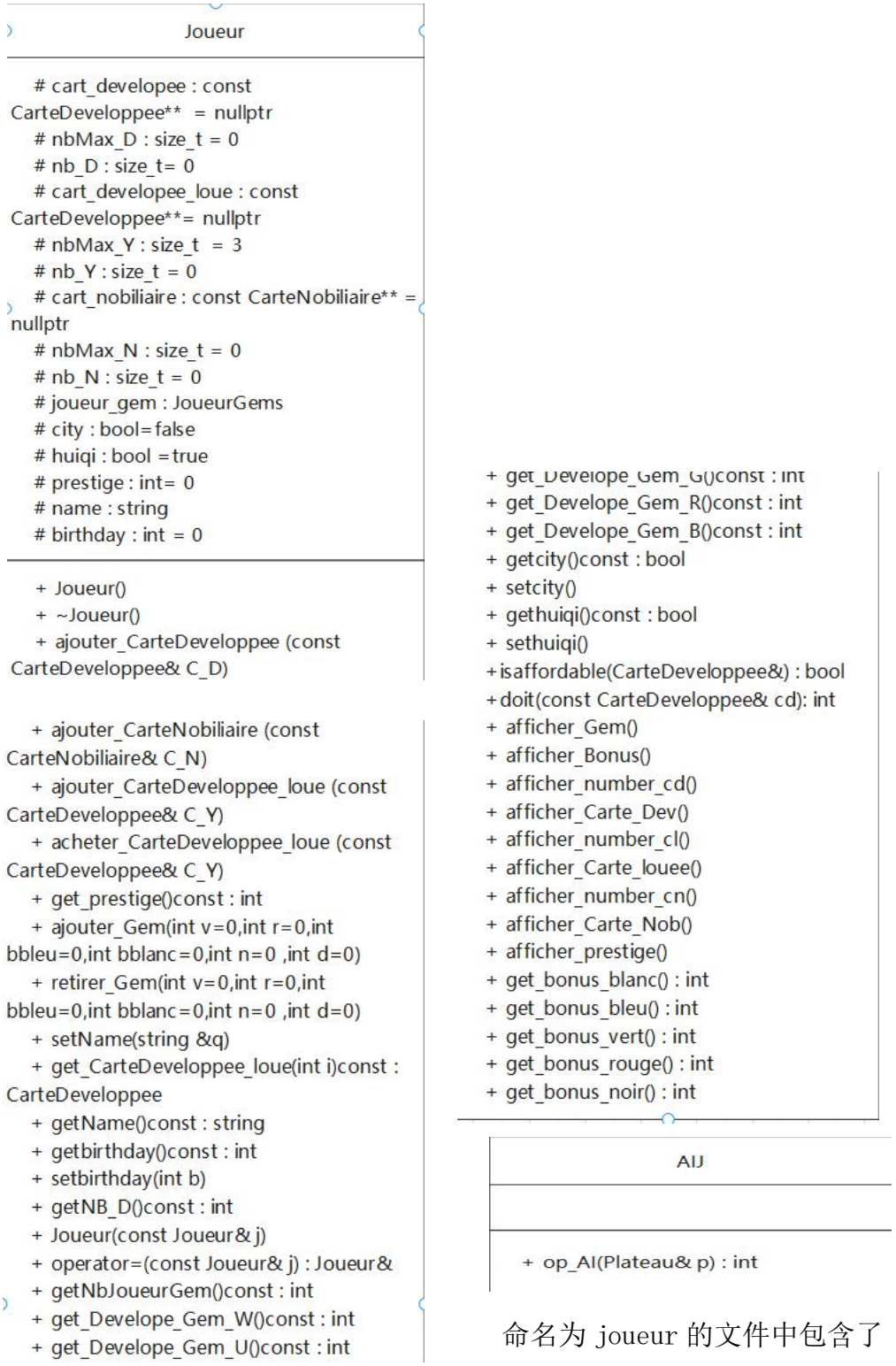


命名为 cartes 的文件中定义了【发展卡】、【贵族卡】，并生成整套【发展卡】、整套【贵族卡】，ToutesLesCartesDeveloppees 类对【发展牌】的生命周期负责，并包含 3 个级别的【发展卡】，分别对应 3 个私有成员变量；ToutesLesCartesNobiliares 对【贵族卡】的生命周期负责。

还定义了一个 ToutLesCartesCity 类对【扩展卡】的生命周期负责，当满足城市要求之后，可以拿到一个【城市面板】同时直接获得胜利。

使用 vector 容器存储各类卡牌，运用 rand（）函数和 swap（）函数对各类卡牌进行洗牌操作；getbonus_string()获得【发展卡】的宝石红利；afficher()来实现对各类卡牌的打印。

3. Joueur



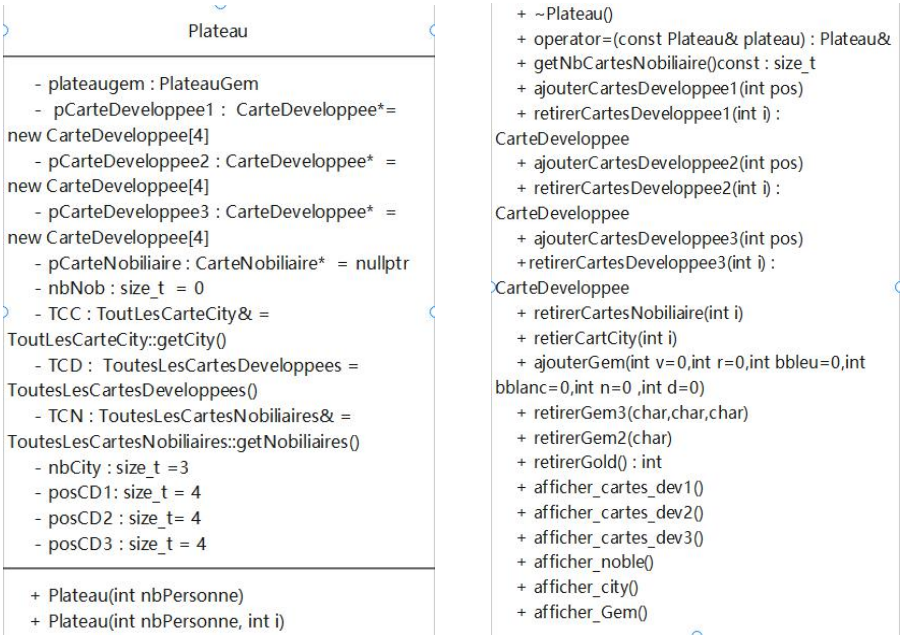
命名为 joueur 的文件中包含了 Joueur 这一个类，主要用来存储玩家信息以及实现游戏中玩家的一系列操作。Joueur 类不对【发展牌】、【贵族牌】、【宝石】的生命周期负责，但对【声望】的生命周期负责。

在游戏中，玩家将持有【发展牌】、【贵族牌】、【宝石】、【声望】、【预留发展牌】，玩家可执行的操作有{获取}宝石、{购买}发展卡、{抵押}发展卡、{拥有}贵族卡(此项操作达到条件后自动为玩家分配贵族卡)。在程序中，Joueur 类实现了对玩家手中的发展卡的{添加}、贵族卡的{添加}、宝石的{增加}和{减少}的操作。

游戏开始时，玩家需要输入自己的名字以及生日，按照年龄的大小来决定游戏中玩家的回合顺序，并对玩家的声望进行初始化（初始化为 0）。每当玩家在己回合选择获取宝石时，调用{ajouter_Gem}函数添加玩家宝石；玩家若选择购买发展卡，则调用{ ajouter_CarteDeveloppee}函数增加玩家发展卡，并调用{retirer_Gem}函数按所购买发展卡所需宝石数对玩家宝石进行删减；若玩家选择抵押发展卡，则调用{ ajouter_CarteDeveloppee_loue} 函数增加玩家发展卡；若玩家欲购买在之前回合抵押的发展卡，则调用{ acheter_CarteDeveloppee_loue}购买抵押发展卡；若玩家的发展卡数量达到贵族卡标准，则自动调用{ ajouter_CarteNobiliaire}函数为符合标准的玩家增加贵族卡。

同时，以 Joueur 为基类，创建了一个 AIJ 作为派生类，可以根据桌子上的卡牌和宝石做出具有一定规律性的操作,实现 Joueur 的基本操作，解锁了人机玩法。

4.Plateau



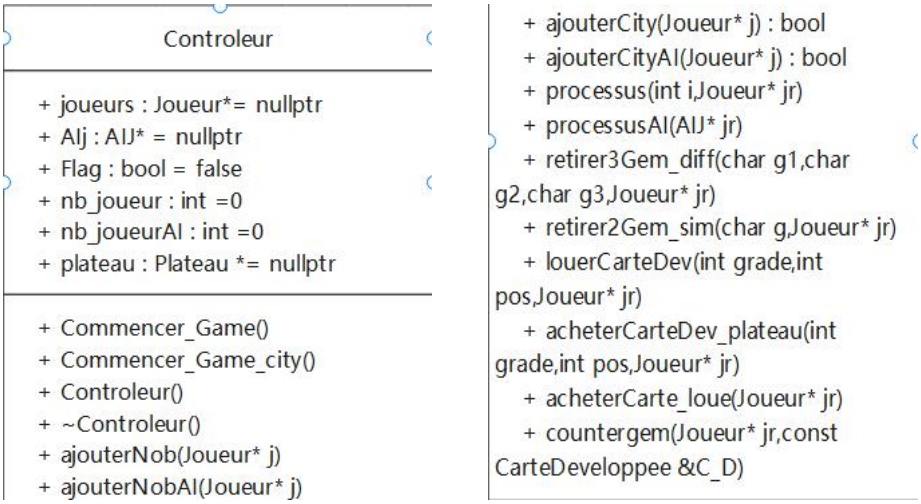
命名为 plateau 的文件是对游戏中桌面上的卡牌以及宝石的一系列操作的实现，Plateau 类对【宝石】的生命周期负责而不对【卡牌】生命周期负责。

在游戏中，桌面上应有【可见发展卡】、【可见贵族卡】、【宝石】。在游戏中，桌面上的发展卡每个等级应各有四张，被玩家购买或抵押后应及时添加，直至不可见卡牌数量为 0。在同一局游戏中，贵族卡应有五张且不再添加。在程序中，plateau 类中实现了对发展卡和贵族卡的{添加}、{移除}，以及宝石数量的{增加}和{减少}操作。

在 plateau 的构造器中，实现了不同玩家人数的桌面牌堆以及宝石的初始化；调用{ajouterCartesDeveloppee1}、{ajouterCartesDeveloppee2}、{ajouterCartesDeveloppee3}、{retirerCartesDeveloppee1}、{retirerCartesDeveloppee2}、{retirerCartesDeveloppee3}、实现对卡牌的增添；{ajouterGem}、{retirerGem2}、{retirerGem3}实现对桌面宝石变化控制；并使用{retirerCartesNobiliaire}函数实现对桌面贵族卡牌的移除，使用{retirerCartCity}函数实现对桌面城市卡牌的移除。

将 class AIJ，class Controleur 作为友元类放在 class plateau 类内，可以实现两者对 plateau 类内所有成员的访问。

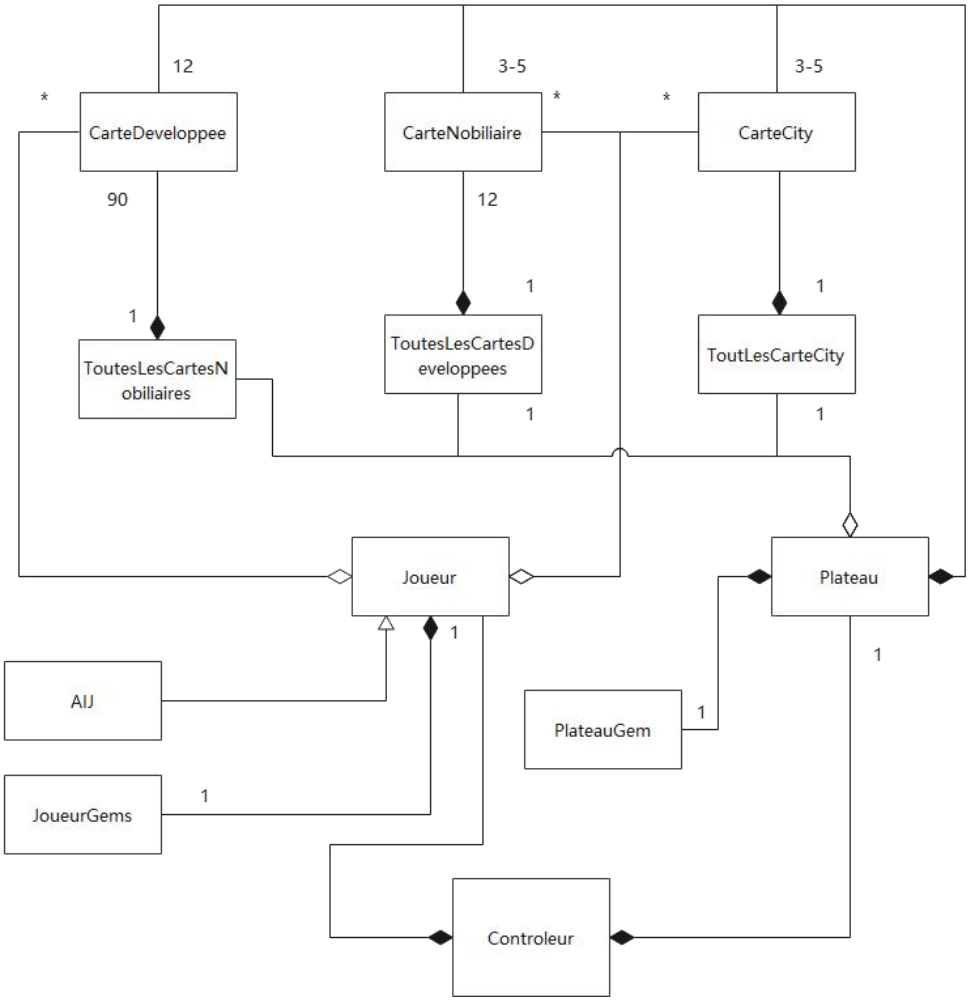
5. Slpendor



命名为 splendor 的文件是对整个游戏流程的实现，包含一个 Controleur 类，对【Joueur】的生命周期负责。在游戏中，游戏管家类是整个游戏的中控系统,能够维持游戏的正常运行以及对胜利条件的判断。在程序中，游戏管家类拥有的方法：（1）【贵族拜访】：判断，当玩家（或 AI）手中的发展卡对应的宝石红利总量与种类达到召唤贵族的要求时，将能够自动为玩家（或 AI）添加一张【贵族牌】（2）【流程控制】：能够依次让玩家（或 AI）从｛回合操作｝的四种选项中选择其一执行命令（3）【结果判断】：能够判断是否有玩家（或 AI）胜出（声望值判断或者是否触发城市面板）

函数{Commencer_Game}负责控制游戏的开始和玩家信息收集,在开始游戏前输入玩家人数及 AI 人数，相应生成对应个数的 Joueur 或者 JoueurAI 实例；再输入玩家的姓名及生日，以玩家年龄大小决定出牌顺序。之后开启回合数，玩家在每个回合拿取符合要求的宝石数或者换取发展卡、租赁发展卡完成回合数。AI 依据场上牌数货币数的形势进行判定，同意在每个回合拿取符合要求的宝石数或者换取发展卡、租赁发展卡完成回合数。调用函数｛ajouterNob｝、｛retirer3Gem_diff｝、｛retirer2Gem_sim｝、｛LouerCarteDev｝、｛acheterCarteDev_plateau｝、｛acheterCarte_loue｝以及{processus}总控函数,完成对桌面宝石变化控制。

四、UML



五、优点

1.引用单例模式,保证 class ToutesLesCartesNobiliaires 只能产生一个实例，并面向整个系统提供

2.引入牌库，以此保证每张发展卡及其相应声望值、宝石数、宝石红利的数据流的稳定性

3.模块化编程,将整个程序分成 Gem,Cartes,Joueur,Plateau,Splendor 多个模块进行编写，方便修改。

4.添加扩展包内容，引入了城市面板，给玩家们提供了新的宝石工厂，加入扩展卡之后，游戏的策略度明显变高，增添了游戏的趣味性

5.实现 AI 和玩家共同进行游戏，可以完成纯玩家玩法，AI 和玩家互战玩法以及 AI 玩法

6.通过将输入的文字转化成 string，存入 TXT，来实现对游戏的存档

六、改进之处

1，发展卡和贵族卡可以改为基类和派生类

2，可以考虑使用工厂方法模式，来集中实现对各个卡牌内存空间的开辟

3，考虑使用策略模式来调用各个函数实现对桌面上卡牌和货币数量的控制

4，图形用户界面需要进行优化和美化

5，占用内存过大

七、个人贡献

姓名	学号	负责内容	时长	贡献度
王乐儿	19124655	主编文件： carte.h carte.cpp （class CarteDeveloppee class CarteNobiliaire class ToutesLesCartesDeveloppees class ToutesLesCartesNobiliaires 发展卡与贵族卡） 其他工作： 撰写报告 代码测试与反馈	30h	15%

陈宁宁	19124741	主编文件： carte.h carte.cpp （class CarteDeveloppee class CarteNobiliaire class ToutesLesCartesDeveloppees class ToutesLesCartesNobiliaires 发展卡与贵族卡） 其他工作： 撰写报告 代码测试与反馈	30h	15%
郑振崴	19124708	主编文件： joueur.h joueur.cpp (class Joueur and class AIJ 人类玩家与 AI 玩家) splendor.h splenfor.cpp （class Controleur 游戏控制器） gem.h gem.cpp （class JoueurGems 玩家宝石） carte.h carte.cpp （class CarteCity class ToutLesCarteCity 城市卡） 重要功能实现： 悔棋； 存档； AI 玩家 其他工作： 部分报告编写	45h	38%
孙胡蝶	19124701	主编文件： plateau.h plateau.cpp （class Plateau 桌面类） gem.h gem.cpp （class PlateauGem 桌面宝石） splendor.h	35h	32%

		splenfor.cpp （class Controleur 中相关游戏进程方法；相关打印） 辅助编写文件： carte.h carte.cpp （基于已有卡牌的洗牌；单张卡牌的打印；） joueur.h joueur.cpp (相关打印) 重要功能实现： 所有面向玩家的打印 其他工作： 框架搭建 主要功能的伪代码描述（不包括 AI 玩家、城市拓展模块与存档） 部分报告编写　　演示视频制作		