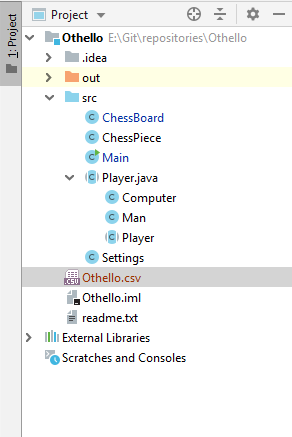
Othello项目文档

17302010009 陆逸凡

# 项目架构：

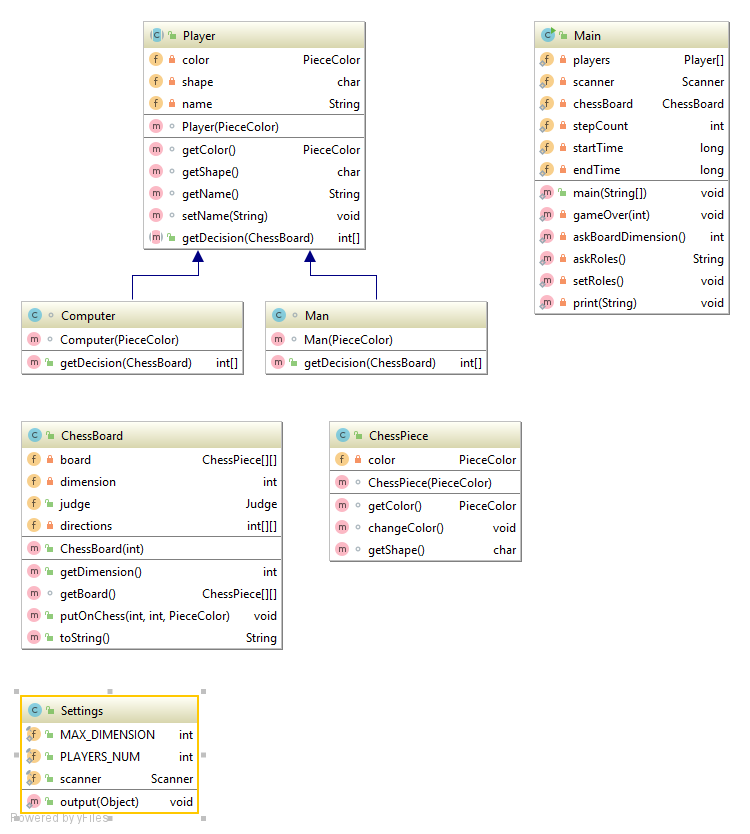


项目的目录结构如上图所示。

其中Main类为程序的执行入口，在main函数里面实例化了其他已定义的类，并实现了黑白棋的游戏流程，另外还负责从控制台获取用户的输入。

从面向对象的角度考虑黑白棋的整个游戏流程，我试图把整个游戏**按照功能、而不是过程来划分**，有了以下的初步思路：每一颗棋子都是棋子类的一个对象，整个棋盘也应该是属于一个类。另外，在人机博弈中，人和计算机都应该是游戏的博弈者，但他们具体的功能有差异，应该属于不同的类。

按功能划分，黑白棋的具体设计如下面的UML图所示（该图由Intellij IDEA自动生成）：



ChessPiece类是黑白棋棋子的类，它包含了一个可枚举的内部类PieceColor类（黑、白），另外还拥有一个颜色的属性，以及被翻转过来、改变颜色的方法等。

ChessBoard类是棋盘所属的类，它包含了一个由ChessPiece类的棋子组成的二维数组，还包含了一个名为Judge的内部类，用来实现涉及到规则、判决方面的功能。由于Judge是ChessBoard的内部类，它访问棋盘的状态、从而进行判决的过程也便简单了很多。

Player类是一个抽象类，无法被实例化，它只是用来抽象整个游戏的博弈者们，定义了作为一个博弈者所应该拥有的属性（如执棋的颜色、棋子的形状等），以及基本的方法（如抽象方法getDecision，通过某种方式获得试图落子的位置）。

两个类继承了Player抽象类——Man类和Computer类，他们分别表示游戏过程中的玩家和电脑，他们都是游戏的博弈者。其中，Man类的getDecision方法从控制台获取用户的输入来作为“试图落子的位置”；而Computer类中实现的getDecision方法则是运用局部最优的思维，穷举棋盘的所有位置以获得当下收益最多的点，来作为“试图落子的位置”。

Settings类则是汇总了所有当前黑白棋版本的配置参数，包括当前版本下棋盘所允许的最大的格数，参与游戏的博弈者数目，获取输入的方式，以及信息输出的位置等。

# 可维护和可拓展性：

如上文所说，面向对象是按功能来划分问题的，那么功能上的统一，以及高内聚、低耦合的设计方式，将会使得程序易于维护和拓展功能。

我认为，该项目的亮点在于利用的面向对象的**多态特性**，把Man和Computer“下棋”的这个功能用getDecision方法统一起来了。那么，在实例化的过程中，就无需对Man和Computer作区别处理，而仅需要把他们都看做是游戏的博弈者，按照他们的父类抽象类里面的属性和方法进行调用。实际执行中的不同之处则利用给面向对象的多态特性来处理。



main函数里面设计游戏流程的主要部分如上图所示。由于**不同的博弈者统一于Player父类，这里是用一个全局的Player[]数组来保存博弈者在实例化的过程中，把人和计算机都当做Player博弈者来处理**。

方便拓展之处在于，这里是用的长度为Settings.PLAYERS\_NUM的Player[]数组来保存所有博弈者的信息，**博弈者回合之间的切换具体实现方式为对这个Player[]数组的顺序重复遍历**（用全局变量stepCount来记录回合数，每回合末让回合数加一，然后对博弈者数目取模，得到下一回合的博弈者）。这个程序的主框架可以十分轻松地变更为Man和Man的二人对决，或Computer对决Computer的傻瓜式表演，抑或拓展到更多的博弈者（>=3）参与的情况。

不难注意到，游戏结束的操作调用的是gameOver函数，通过传入不同的status状态值，执行不同的游戏结束程序。这样在之后规则拓展、博弈人数增加的情况下，仍然具有很高的可维护性。

另外，涉及到规则的判决方面的复杂逻辑操作都被封装在Judge类的函数里面，暴露给外界的方法只有canMove（检查当前棋盘状态下是否某一方可以继续落子）、isLegal（检查当前将要落子的位置是否合法）、package-private的computePoint（计算某一方在当前位置落子的时间上的局部收益）、getResult（返回当前状态下棋盘的黑白棋数目统计）。并且，所有规则判断都包装在Judge里面，方便后期对规则的修改和拓展；Judge作为ChessBoard类的一个内部类，方便访问到ChessBoard的全部属性。

考虑到后期的需求变更，**所有游戏本身的参数都被设置在Setting.java文件里面**。由于猜想游戏后期可能会拓展到UI界面，输入输出可能都不在控制台上，于是此文件里**还包括了获取输入的来源，以及信息输出的位置**。在项目的其他位置，统一调用Settings里面的输入输出。

但是，在涉及到判断规则的具体实现过程方面，并没有设计得那么的可拓展，原因是我无法猜想和估计三人以上的棋类博弈，规则究竟是什么样。所以，我能优化设计的地方，只是游戏的主框架。

# 细节补充说明

日志文件Othello.csv在打开时，请不要在纯文本界面和表格界面来回切换。**Intellij的csv插件貌似有一个恼人的bug**，即：在程序已经完全退出的静态的情况下，如果在纯文本界面光标定为在新的一行的开头，那么切换到表格界面后再切回来，有概率新的一行就凭空多出来几个逗号，从而影响下一次写的位置，导致错位。

当然，如果很不幸助教一开始就遇到了这个错位的问题，请将文本界面里面csv插件自动生成的逗号全部删去，且光标停留在新的一行的起点，然后在表格界面和文本界面来回切换，在确认此次csv插件不会自动写入逗号之后，停留在表格界面，再开始以后的测试（且不要再切到文本界面了，说不定这个csv插件又出bug）。