

Lab Manuals for  
**Software Construction**

**Lab-2  
Abstract Data Type (ADT) and Object-Oriented Programming (OOP)**



**School of Computer Science and Technology**

**Harbin Institute of Technology**

**Spring 2020**

**目录**

[1 实验目标 1](#_Toc34505210)

[2 实验环境 1](#_Toc34505211)

[3 实验要求 2](#_Toc34505212)

[3.1 Poetic Walks (MIT) 2](#_Toc34505213)

[3.2 Re-implement the Social Network in Lab1 3](#_Toc34505214)

[3.3 Playing Chess 3](#_Toc34505215)

[4 实验报告 7](#_Toc34505216)

[5 提交方式 8](#_Toc34505217)

[6 评分方式 8](#_Toc34505218)

# 实验目标

本次实验训练抽象数据类型（ADT）的设计、规约、测试，并使用面向对象编程（OOP）技术实现ADT。具体来说：

* 针对给定的应用问题，从问题描述中识别所需的ADT；
* 设计ADT规约（pre-condition、post-condition）并评估规约的质量；
* 根据ADT的规约设计测试用例；
* ADT的泛型化；
* 根据规约设计ADT的多种不同的实现；针对每种实现，设计其表示（representation）、表示不变性（rep invariant）、抽象过程（abstraction function）
* 使用OOP实现ADT，并判定表示不变性是否违反、各实现是否存在表示泄露（rep exposure）；
* 测试ADT的实现并评估测试的覆盖度；
* 使用ADT及其实现，为应用问题开发程序；
* 在测试代码中，能够写出testing strategy并据此设计测试用例。

# 实验环境

实验环境设置请参见Lab-0 实验指南。

除此之外，本次实验需要你在Eclipse IDE中安装配置EclEmma（一个用于统计JUnit测试用例的代码覆盖度的plugin）。请访问<http://www.eclemma.org>，了解EclEmma并学习其安装、配置和使用。

本次实验在GitHub Classroom中的URL地址为：

<https://classroom.github.com/a/z9utaaos>

请访问该URL，按照提示建立自己的Lab2仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向GitHub仓库提交。实验包含的3个任务分别在不同的目录内开发，具体目录组织方式参见各任务最后一部分的说明。请务必遵循目录结构，以便于教师/TA进行测试。

# 实验要求

针对以下所有四个任务，请为每个你设计和实现的ADT撰写mutability/ immutability说明、AF、RI、safety from rep exposure。给出各ADT中每个方法的spec。为每个ADT编写测试用例，并写明testing strategy。

## Poetic Walks (MIT)

请阅读<http://web.mit.edu/6.031/www/sp17/psets/ps2/>，遵循该页面内的要求完成编程任务。

* 在Get the code步骤中，你无法连接MIT的Athena服务器，请从以下地址获取初始代码：  
  <https://github.com/rainywang/Spring2020_HITCS_SC_Lab2/tree/master/P1>
* 在作业描述中若遇到“commit and push”的要求，请将你的代码push到你的GitHub Lab2仓库中。
* MIT作业页面提及的文件路径，请按照下表的目录结构进行调整。例如“test/poet”应为“test/P1/poet”，“src/poet”应为“src/P1/poet”。
* 其他步骤请遵循MIT作业页面的要求。

**项目的目录结构：**

项目名称： Lab2-学号

src

P1

graph

….java

poet

….java

….txt

test

P1

graph

…Test.java

poet

…Test.java

….txt

请使用git指令将符合上述结构的代码push到你的GitHub Lab2仓库中。

## Re-implement the Social Network in Lab1

回顾Lab1实验手册中的3.2节Social Network，你针对所提供的客户端代码实现了FriendshipGraph类和Person类。

在本次实验中，请基于你在3.1节Poetic Walks中定义的Graph<L>及其两种实现，重新实现Lab1中3.3节的FriendshipGraph类。

**注1：**可以忽略你在Lab1中实现的代码，无需其基础上实现本次作业；

**注2：**在本节FriendshipGraph中，图中的节点仍需为Person类型。故你的新FriendshipGraph类要利用3.1节已经实现的ConcreteEdgesGraph<L>或ConcreteVerticesGraph<L>，L替换为Person。根据Lab1的要求，FriendshipGraph中应提供addVertex()、addEdge()和getDistance()三个方法：针对addVertex()和addEdge()，你需要尽可能复用ConcreteEdgesGraph<L>或ConcreteVerticesGraph<L>中已经实现的add()和set()方法，而不是从0开始写代码实现或者把你的Lab1相关代码直接复制过来；针对getDistance()方法，请基于你所选定的ConcreteEdgesGraph<L>或ConcreteVerticesGraph<L>的rep来实现，而不能修改其rep。

**注3：**不变动Lab1的3.3节给出的客户端代码（例如main()中的代码），即同样的客户端代码仍可运行。重新执行你在Lab1里所写的JUnit测试用例，测试你在本实验里新实现的FriendshipGraph类仍然表现正常。

**项目的目录结构：**

项目名称： Lab2-学号

src

P2

FriendshipGraph.java

Person.java

...

test

P2

FriendshipGraphTest.java

...

(无需将3.1节实现的Graph<L>的程序源文件复制到P2目录下)

请使用git指令将符合上述结构的代码push到你的GitHub Lab2仓库中。

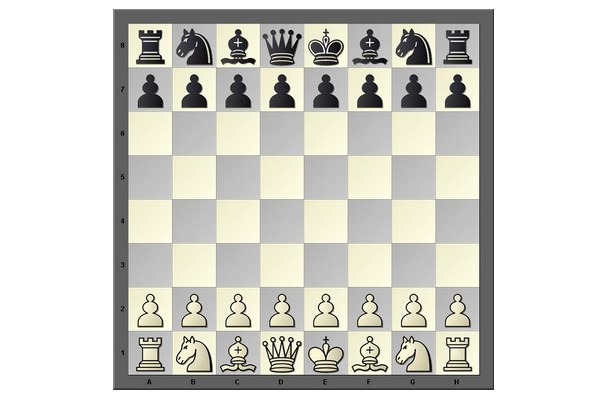
## Playing Chess

前面3.1和3.2节是在已经给定ADT设计的基础上进行具体实现和测试。本节要求你从0开始设计一套ADT，支持实现特定的功能需求。

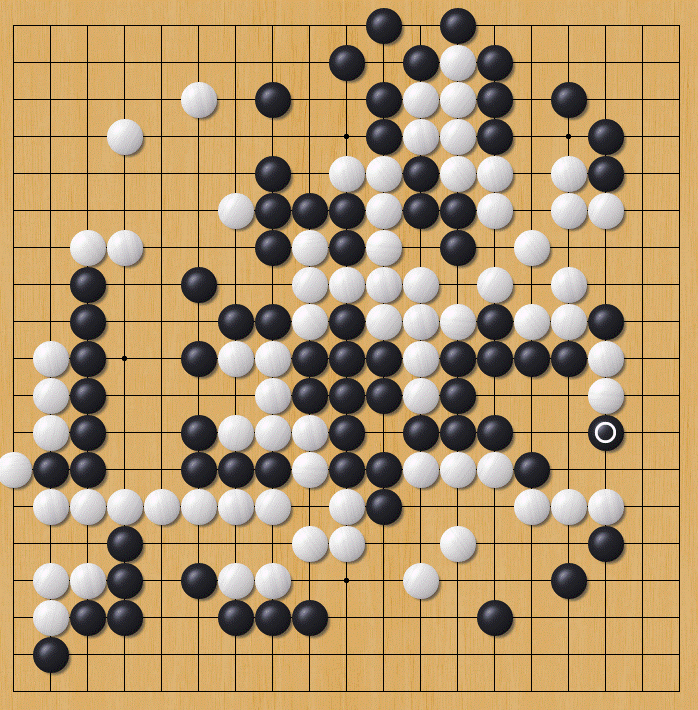
棋类游戏由一个棋盘、一组棋子组成，双方交替在棋盘上走棋。本实验仅考虑在正方形棋盘上进行的棋类游戏，包括国际象棋和围棋。

一个尺寸为n\*n的棋盘：其中n表示每一行/每一列的格子数。例如：围棋盘由18\*18个格子构成，国际象棋盘由8\*8个格子构成。棋盘上共有n\*n个格子和(n+1)\*(n+1)个交叉点。

一组棋子：每个棋子属于一个特定的种类，不同的棋类游戏中包含的棋子种类不同，例如：围棋/五子棋中仅包含黑子和白子；国际象棋中包含king、queen、rock、bishop等类型，且双方棋手拥有同样的棋子（用颜色区分）。在一盘棋局中，不同种类的棋子的数量不同，例如国际象棋中每方有2个rock、2个bishop；围棋中的黑子和白子数量无限多。在游戏进行过程中，需要区分棋子是属于游戏中的哪一方玩家。



国际象棋棋盘和棋子示例



围棋棋盘和棋子示例

下棋过程中，棋手的动作包括：

* 将一个棋子从棋盘外放到棋盘上的某个合法位置（围棋、五子棋等）
* 将一个棋子从棋盘上的一个合法位置移动到另一个合法位置（中国象棋、国际象棋等）
* 将对方的一个或多个棋子从棋盘上移走

这里的合法位置是指：格子内（例如国际象棋、五子棋）或交叉点（例如围棋、中国象棋），均使用坐标(x, y)表示。如果是格子内的位置，x和y分别表示该格子的横纵坐标，x和y的取值范围是[0,n-1]；如果是交叉点的位置，那么将棋盘的第(0,0)个格子的左下角作为坐标系的原点，那么x和y表示棋盘上第(x,y)个格子的左下角的点，且x和y的取值范围是[0,n]。如下图所示，红色标签表示其左下方的交叉点的坐标，蓝色坐标表示其所处的格子的坐标。两个玩家分别位于棋盘的下方和上方。



使用Java OOP实现一个简单的棋类模拟软件。为“一盘棋类游戏”、“玩家”、“棋盘”、“棋子”、“棋盘上的位置”、“下棋动作”设计ADT（类或接口），命名分别为Game、Player、Board、Piece、Position、Action。如果针对不同的棋类游戏需要从这些类派生子类或者实现接口，请自行进行设计。

你可自由为所有的类/接口设计rep和方法，能够支持完成以下功能：

1. 输入是游戏的类型（围棋或国际象棋），创建一个符合该棋类游戏的一个Board对象、一组Piece对象。棋盘大小、棋子种类、棋子数量无需外部参数输入，你的方法可读取外部配置文件或以静态常量写入代码，需要符合围棋和国际象棋的真实规则；
2. 给定两个名字，初始化两个Player对象；将各Piece对象的所有权分配给两个Player对象（在围棋中，所有白子都属于player1，所有黑子都属于player2；在国际象棋中，双方的棋子种类和数量都是完全一样的，只有颜色不同）；如果是国际象棋，需要将所有Piece对象放置到棋盘上的初始位置（见上页图，各棋子的初始位置**必须要符合**国际象棋规则）；如果是围棋，则所有棋子不需放到棋盘上（上页图表示的是下棋过程中的某个特定时刻的状态，围棋棋盘的初始状态是棋盘上无任何棋子）。
3. 给定“棋手、一颗棋子、指定位置的横坐标、指定位置的纵坐标”作为输入参数，将该棋手的该颗棋子放置在棋盘上（考虑游戏的类型，不同的棋类游戏中的位置含义不同）。需考虑异常情况，例如：该棋子并非属于该棋手、指定的位置超出棋盘的范围、指定位置已有棋子、所指定的棋子已经在棋盘上、等。——**注：无需考虑**实际的落子规则，但你可以扩展该spec让你的该操作能具备遵循国际象棋/围棋的真实落子规则的能力。
4. 移动棋子（针对国际象棋）：给定“棋手、初始位置和目的位置的横纵坐标”，将处于初始位置的棋子移动到目的位置。需要考虑处理各种异常情况，例如：指定的位置超出棋盘的范围、目的地已有其他棋子、初始位置尚无可移动的棋子、两个位置相同、初始位置的棋子并非该棋手所有、等。——**注：无需考虑**实际的走棋规则，但你可以扩展该spec让你的该操作能具备遵循国际象棋的真实走棋规则的能力。
5. 提子（针对围棋）：给定“棋手、一个位置的横纵坐标”，将该位置上的对手棋子移除。需要考虑处理异常情况，例如：该位置超出棋盘的范围、该位置无棋子可提、所提棋子不是对方棋子、等。——**注：无需考虑**实际的提子规则，但你可以扩展该spec让你的该操作能具备遵循围棋的真实提子规则的能力。
6. 吃子（针对国际象棋）：给定“棋手、两个位置横纵坐标”，将第一个位置上的棋子移动至第二个位置，第二个位置上原有的对手棋子从棋盘上移除。需要处理异常情况，例如：指定的位置超出棋盘的范围、第一个位置上无棋子、第二个位置上无棋子、两个位置相同、第一个位置上的棋子不是自己的棋子、第二个位置上的棋子不是对方棋子、等。——**注：无需考虑**实际的吃子规则，但你可以扩展该spec让你的该操作能具备遵循实际国际象棋吃子规则的能力。

在完成上述功能时，除了类名已确定无需修改，属性和方法的名字、数据类型可自由设计，方法的输入参数也可自行设计。若需要其他的辅助类，也请自行设计。

写一个基于命令行的主程序MyChessAndGoGame.java，在其main()实现以下功能：

1. 让用户选择创建一盘国际象棋或一盘围棋，用户输入“chess”或“go”分别代表国际象棋和围棋；让用户输入两个玩家的名字；
2. 启动比赛，程序提示玩家双方交替采取行动，直到一方输入“end”而结束。双方分别采取行动的时候，可以选择以下行为之一，也可以选择“跳过”（即放弃本次采取行动的权利）：

* 将尚未在棋盘上的一颗棋子放在棋盘上的指定位置；
* 移动棋盘上某个位置的棋子至新位置；
* 提子或吃子；
* 查询某个位置的占用情况（空闲，或者被哪一方的什么棋子所占用）；
* 计算两个玩家分别在棋盘上的棋子总数。

请自行为用户设计针对上述行为所需输入的数据，用户输入越简洁越好。

1. 当某一方输入end结束游戏之后，双方可以查看本次比赛的走棋历史，即能够查询自己所走的所有步骤。

注：本次任务并非完整实现围棋和国际象棋的所有规则，甚至有些要求与实际的棋类规则有冲突，请严格按照上述说明进行设计，无需扩展。

**项目的目录结构**

项目名称： Lab2-学号

src

P3

...java

test

P3

...java

请使用git指令将符合上述结构的代码push到你的GitHub Lab2仓库中。

# 实验报告

针对上述3个编程题目，请遵循雨课堂中Lab2页面给出的**报告模板**，撰写简明扼要的实验报告。

**实验报告的目的是记录你的实验过程，尤其是遇到的困难与解决的途径。**不需要长篇累牍，记录关键要点即可，但需确保报告覆盖了本次实验的所有开发任务（3个问题，每个问题下有一系列任务）。

注意：

* 实验报告不需要包含所有源代码，请根据上述目的有选择的加入关键源代码，作为辅助说明。
* 请确保报告格式清晰、一致，故请遵循目前模板里设置的字体、字号、行间距、缩进；
* 实验报告提交前，请“目录”上右击，然后选择“更新域”，以确保你的目录标题/页码与正文相对应。
* 实验报告文件可采用Word或PDF格式，命名规则：Lab2-学号-Report。

# 提交方式

**截止日期：**第7周周日夜间23:55。

**源代码：**从本地Git仓库推送至个人GitHub的Lab2仓库内。

**实验报告：**随代码仓库（doc）目录提交至GitHub。

# 评分方式

TA在第4-6周实验课上现场验收：学生做完实验之后，向TA提出验收申请，TA根据实验要求考核学生的程序运行结果并打分。现场验收并非必需，由学生主动向TA提出申请。

Deadline之后，教师和TA对学生在GitHub上的代码进行测试、阅读实验报告，做出相应评分。