哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Software Construction: Developing High-Quality Software Systems

软件构造：开发高质量的软件系统

Wang Zhongjie
rainy@hit.edu.cn

February 21, 2020

任课教师

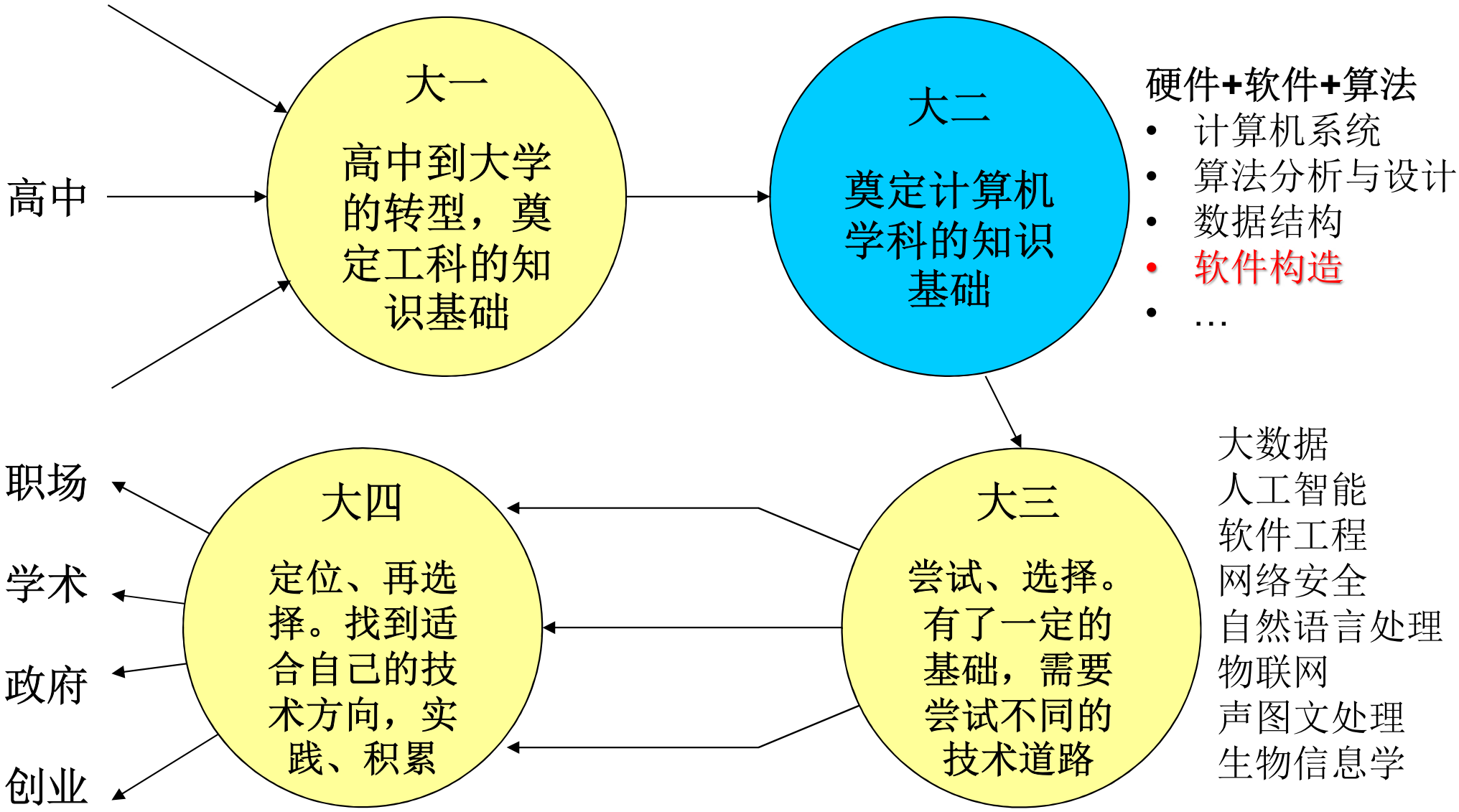
- **王忠杰 教授，博士生导师**
  - 企业与服务智能计算研究中心 (ICES)
  - 电子邮件： rainy@hit.edu.cn
    rainy.wang@gmail.com
  - 联系电话： 18604507162
  - 地点： 新技术楼510房间
  - 答疑时间： 每周一10:00-11:30 (1-15周)
  - 研究方向：
    - 互联网服务计算
    - 软件工程、软件体系结构

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

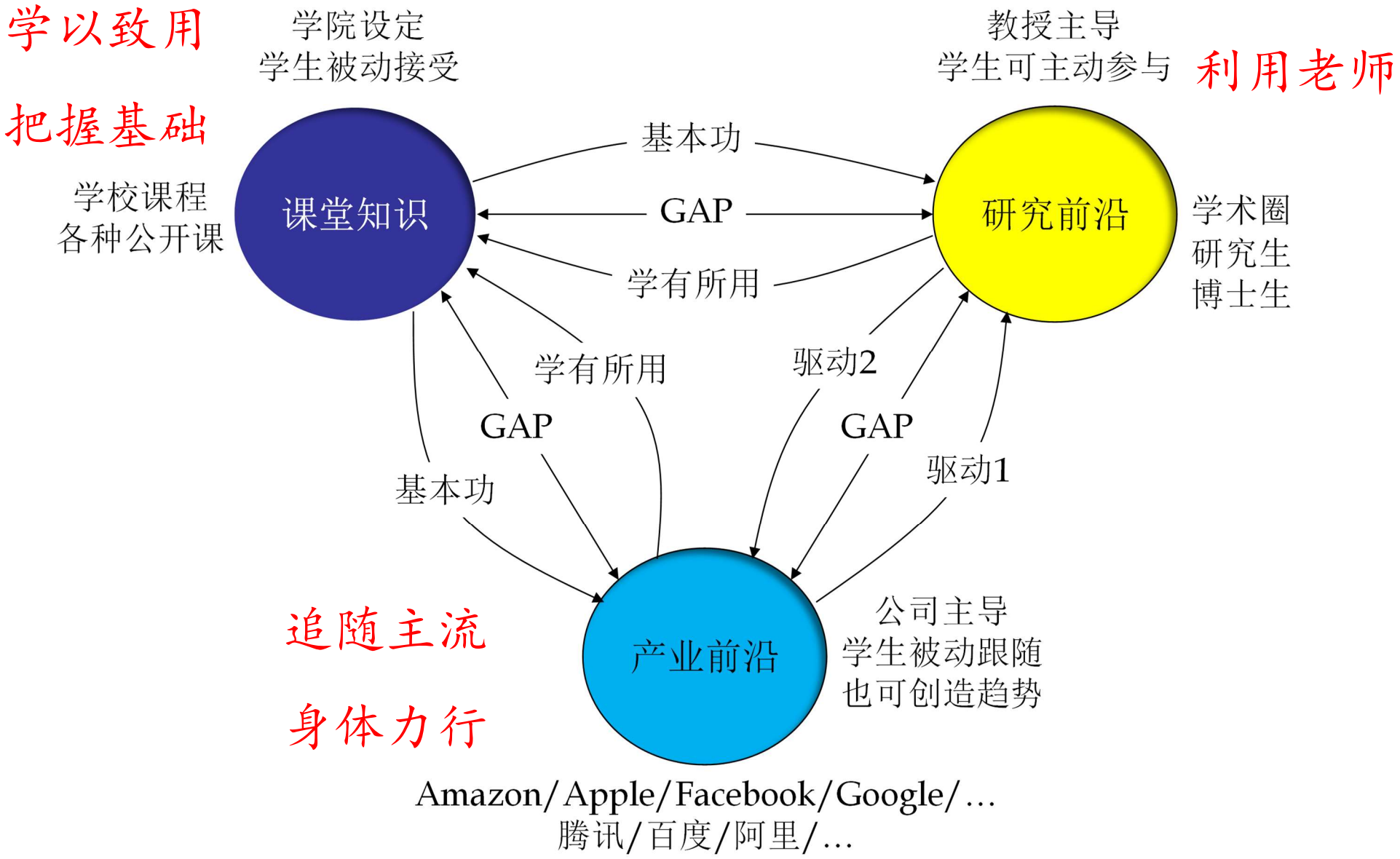
Wechat group

- **2020春软件构造 (软工+英才)**

建议 1：打好基础，为“选择”做准备



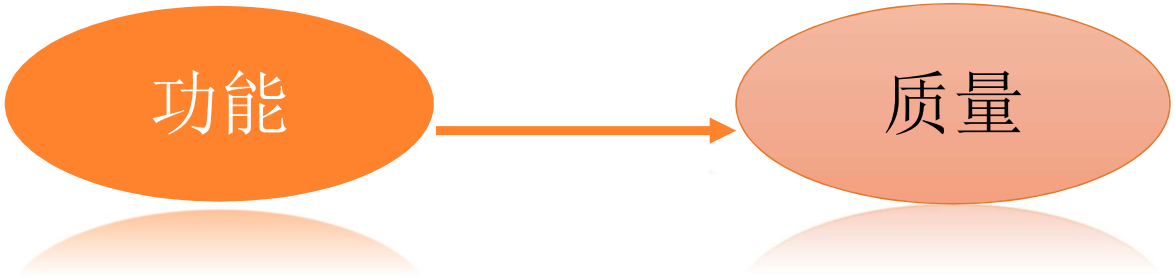
建议2：弥补课堂和现实的gap



Goals of this Course

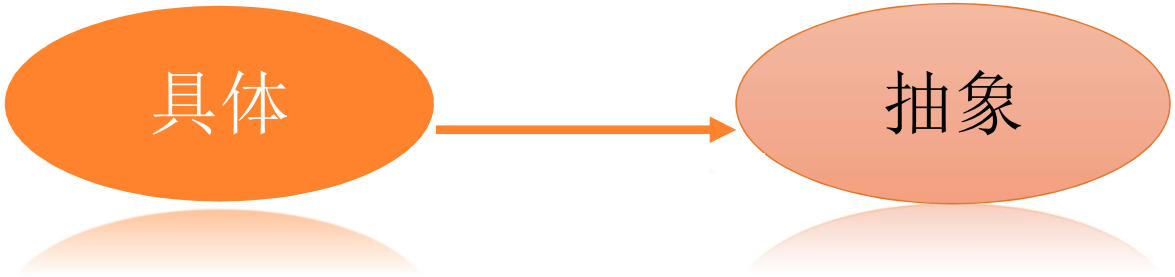
- **Goal: understanding both the building blocks and the design principles for construction of software systems 构造原理？**
  - 在高级语言程序设计的基础上，认识软件构造的质量标准与目标，学习软件构造的基本过程，从而具备面向质量目标的复杂软件构造方法与能力
  - 深入学习抽象数据类型 ADT 与面向对象编程 OOP
  - 初步掌握面向关键质量目标（可理解性、可维护性、可复用性、健壮性、时空性能）的软件构造基本技术
  - 了解软件代码重构和面向更复杂软件系统的高级构造技术
- **For each desired program behavior there are infinitely many programs 多种不同的软件构造方案，有什么差异？如何选择？**
  - What are the differences between the variants?
  - Which variant should we choose?
  - How can we synthesize a variant with desired properties?

Goals of this Course



程序设计与实现能力

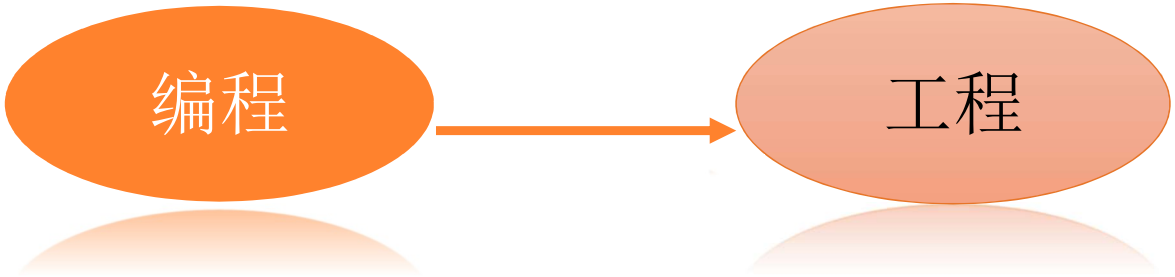
- 了解软件开发过程中应考虑哪些质量目标
- 掌握面向关键质量目标的软件基本构造技术
- 形成面向质量目标的软件开发思维模式



系统设计与实现能力

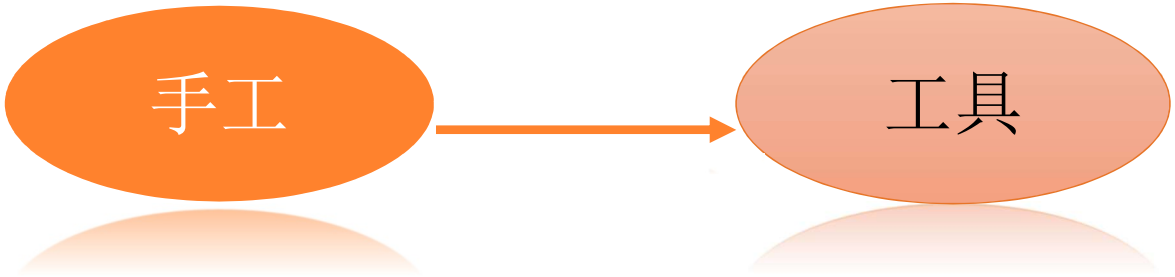
- 掌握“面向抽象编程”的核心思想和面向对象软件开发的基本过程
- 能够对实际应用问题进行抽象和建模
- 利用模型与开发者和用户进行有效表达和沟通

Goals of this Course



系统分析与评价能力

- 从关注单一开发环节到关注全开发过程的转换
- 根据用户期望质量特性进行全生命周期**系统分析与评价**
- 发现系统设计的缺陷并做出优化和改进



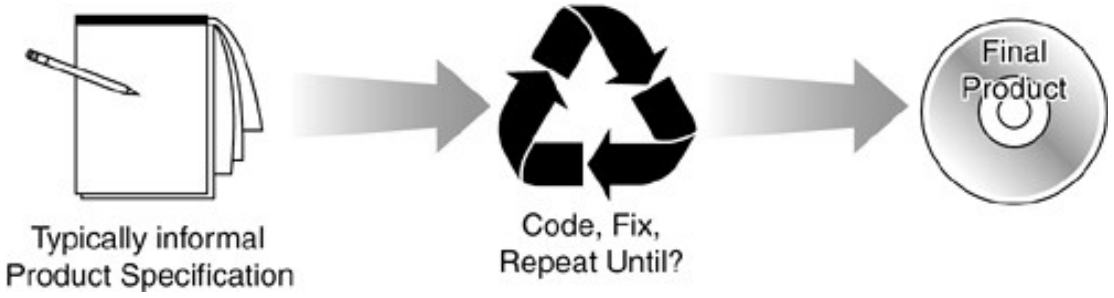
利用现代软件构造工具的能力

- 了解复杂软件系统相对于简单程序的本质差异
- 初步掌握利用各类软件开发工具进行编码、测试和质量保障
- **利用现代软件构造工具**进行高质量和高效率软件开发

A typical software design process

1. **Discuss software that needs to be written**
2. **Write some code**
3. **Test the code to identify the defects**
4. **Debug to find causes of defects**
5. **Fix the defects**
6. **If not done, return to step 1**

写代码----试错----改错，如此循环



Better software design

- **Think before coding** 未雨绸缪
- **Consider non-functional quality attributes** 考虑非功能质量属性
  - **Maintainability, extensibility, performance, …**
- **Propose, consider design alternatives** 考虑多种设计选择
- **Make explicit design decisions** 把设计决策明确写下来

- **Using a design process…**
  - A design process organizes your work
  - A design process structures your understanding
  - A design process facilitates communication

Design goals, principles, and patterns

- **Design goals** enable evaluation of designs
  - e.g. maintainability, reusability, scalability
- **Design principles** are heuristics that describe best practices
  - e.g. high correspondence to real-world concepts
- **Design patterns** codify repeated experiences, common solutions
  - e.g. template method pattern

设计目标：编程的“视野”

设计原则：编程的“标尺”

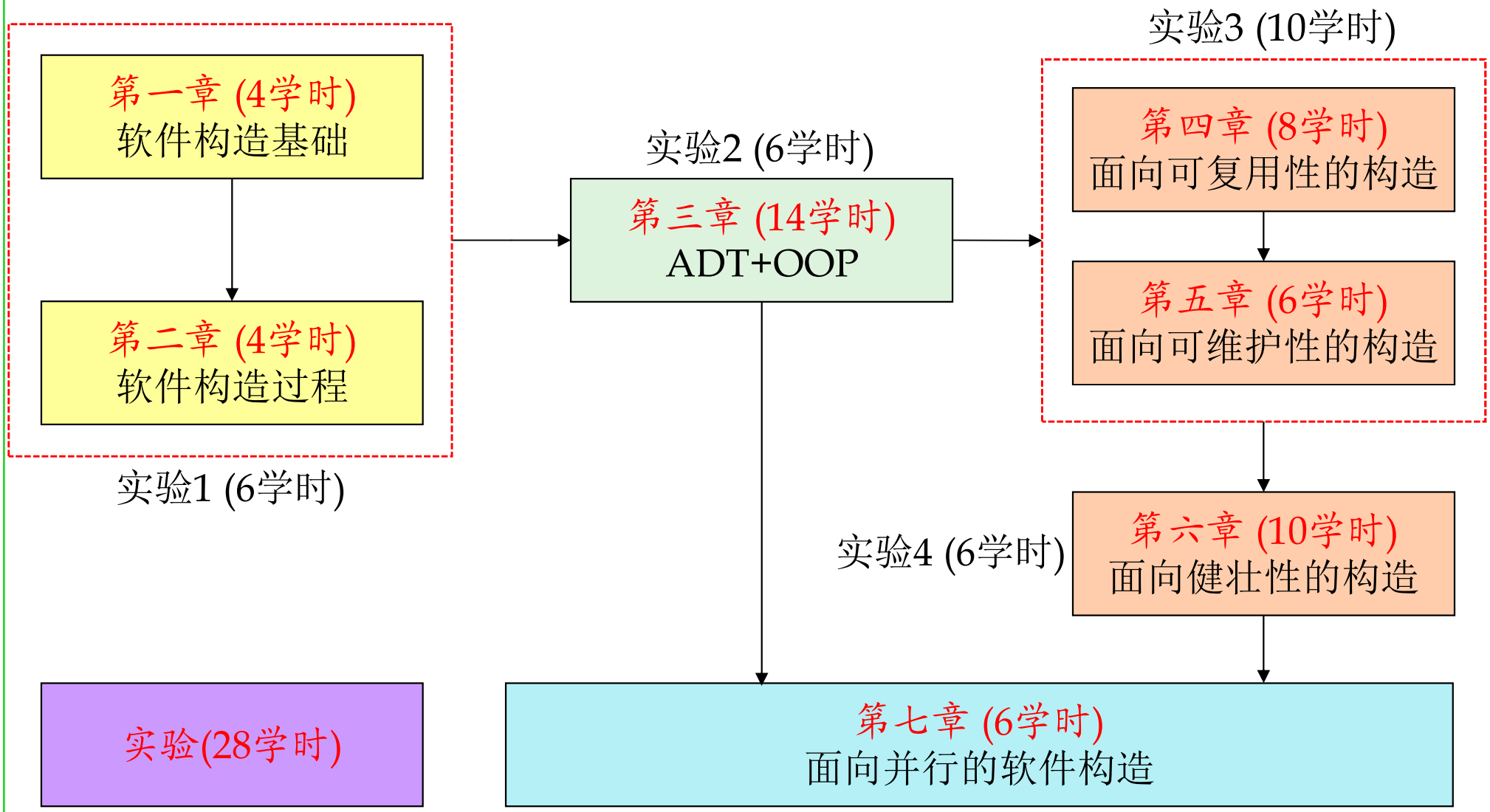
设计模式：编程的“经验”

Learning goals

- Ability to **design** medium-scale programs
- Understanding **OO programming** concepts & design decisions
- Proficiency with basic **quality assurance** techniques for functional correctness
- Fundamentals of **concurrency and distributed systems**
- Practical skills

We who cut mere stones
must always be envisioning cathedrals.
我们切割石头的人心里也必须要想着大教堂

Chapters and hours of the course



Basic information

- 授课对象: 计算机学院**2018**级本科生 **(**软件工程、英才**)**
- 课程分类: 核心基础课
- 学时: **80 (52+28)**
- 先修课程: **C/C++/Java**高级语言程序设计；
  计算机系统；数据结构与算法；
- 上课时间/地点:
  - 1-14周 周一1-2节/周三3-4节 正心楼44
  - 1-15周 周二9-10节 格物楼机房207/213
- 考试时间:
  - 17周 待定

Reading materials (mandatory)

- **MIT Course 6.031: Software Construction**

  http://web.mit.edu/6.031/www/fa19/

6.031: Software Construction

Fall 2019 · Course Staff · MWF11-12:30 (34-101)

General

General Information
Collaboration and Public Sharing
Code Reviewing
Nanoquiz Grading and Makeups
Technical Tips & Troubleshooting
I have a question, who do I ask?

Calendar: classes, assignments, OH/lab

Problem Sets

0: Turtle Graphics
1: Flashcards
2: Multi-Startup Set
3: Memely
4: Memory Scramble

Project

Norn Mailing List System

Quizzes

Quiz 1 and Quiz 1 solutions
Quiz 2 and Quiz 2 solutions

Quiz Archive

Course Archive

Previous semesters

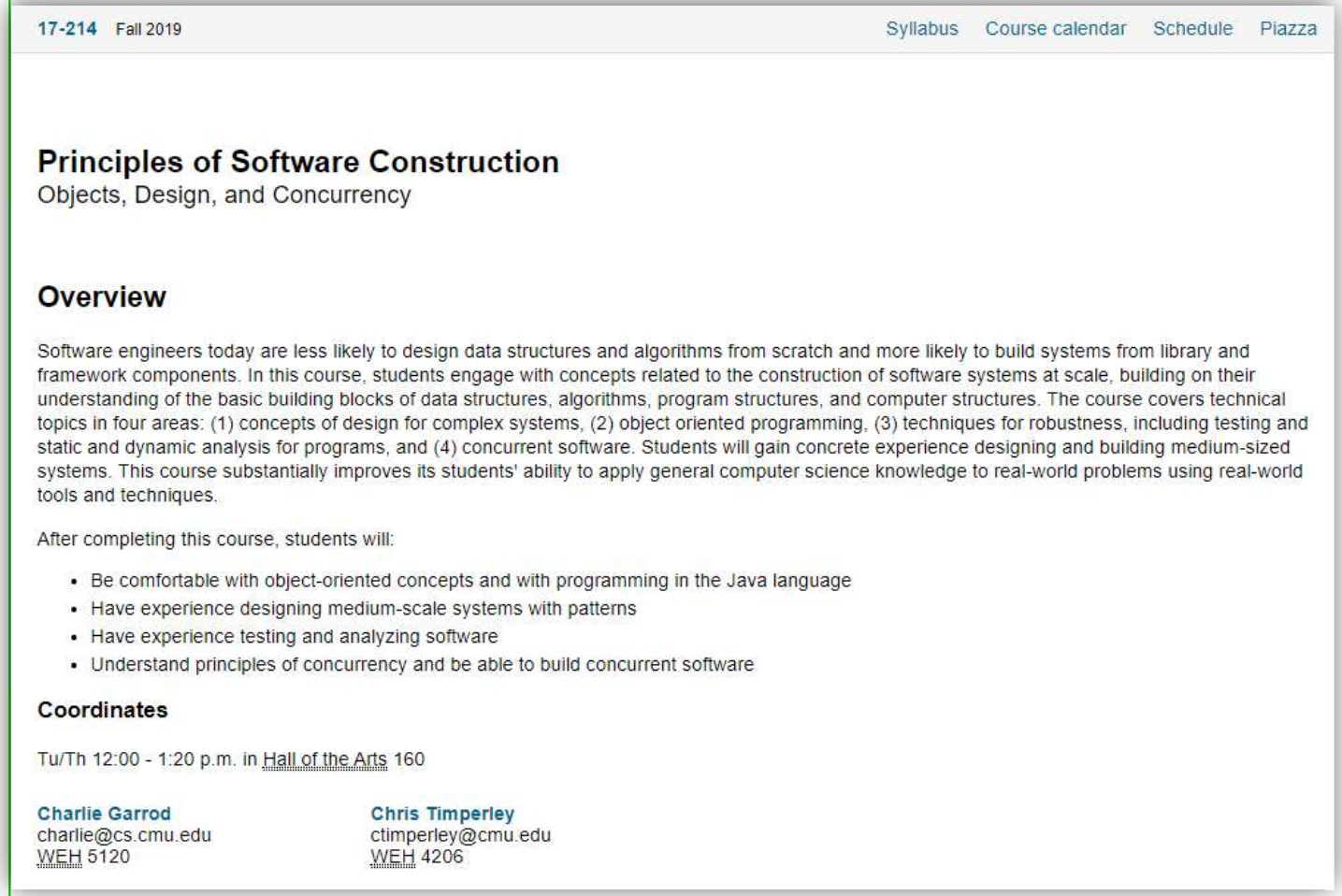
Readings

01: Static Checking
02: Basic Java
03: Testing
04: Code Review
05: Version Control
06: Specifications
07: Designing Specifications
08: Mutability & Immutability
09: Avoiding Debugging
10: Abstract Data Types
11: Abstraction Functions & Rep Invariants
12: Interfaces, Generics, & Enums
13: Debugging
14: Recursion
15: Equality
16: Recursive Data Types
17: Regular Expressions & Grammars
18: Parsers
19: Concurrency
20: Thread Safety
21: Locks & Synchronization
22: Queues & Message-Passing
23: Sockets & Networking
24: Callbacks
25: Map, Filter, Reduce
26: Little Languages I
27: Little Languages II
28: Team Version Control
29: Ethical Software Engineering

Reading materials (highly recommended)

https://www.cs.cmu.edu/~charlie/courses/17-214/2019-fall/

- **CMU 17-214 Principles of Software Construction: Objects, Design, and Concurrency**



17-214 Fall 2019 Syllabus Course calendar Schedule Piazza

Principles of Software Construction
Objects, Design, and Concurrency

Overview

Software engineers today are less likely to design data structures and algorithms from scratch and more likely to build systems from library and framework components. In this course, students engage with concepts related to the construction of software systems at scale, building on their understanding of the basic building blocks of data structures, algorithms, program structures, and computer structures. The course covers technical topics in four areas: (1) concepts of design for complex systems, (2) object oriented programming, (3) techniques for robustness, including testing and static and dynamic analysis for programs, and (4) concurrent software. Students will gain concrete experience designing and building medium-sized systems. This course substantially improves its students' ability to apply general computer science knowledge to real-world problems using real-world tools and techniques.

After completing this course, students will:

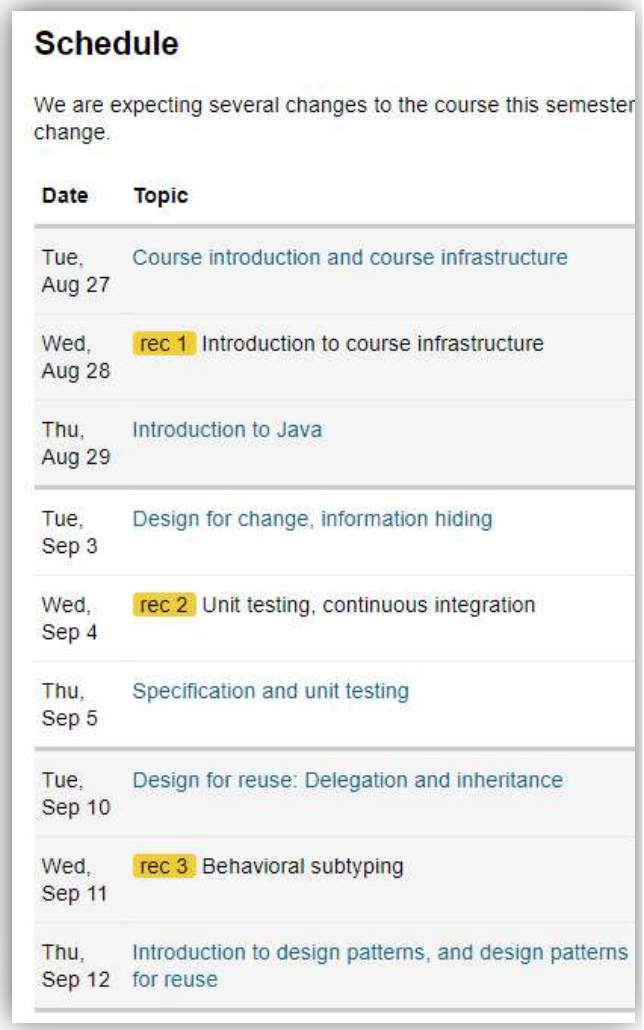
- Be comfortable with object-oriented concepts and with programming in the Java language
- Have experience designing medium-scale systems with patterns
- Have experience testing and analyzing software
- Understand principles of concurrency and be able to build concurrent software

Coordinates

Tu/Th 12:00 - 1:20 p.m. in Hall of the Arts 160

Charlie Garrod
charlie@cs.cmu.edu
WEH 5120

Chris Timperley
ctimperley@cmu.edu
WEH 4206



Schedule

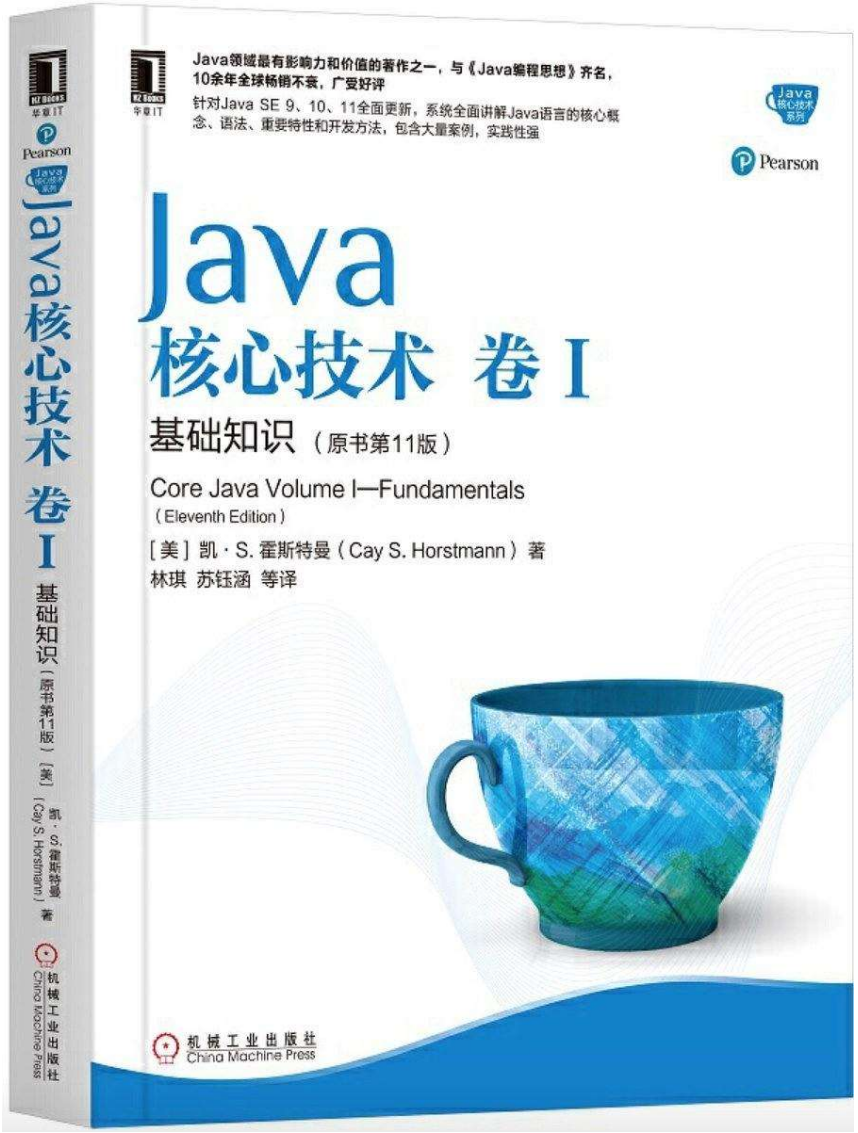
We are expecting several changes to the course this semester change.

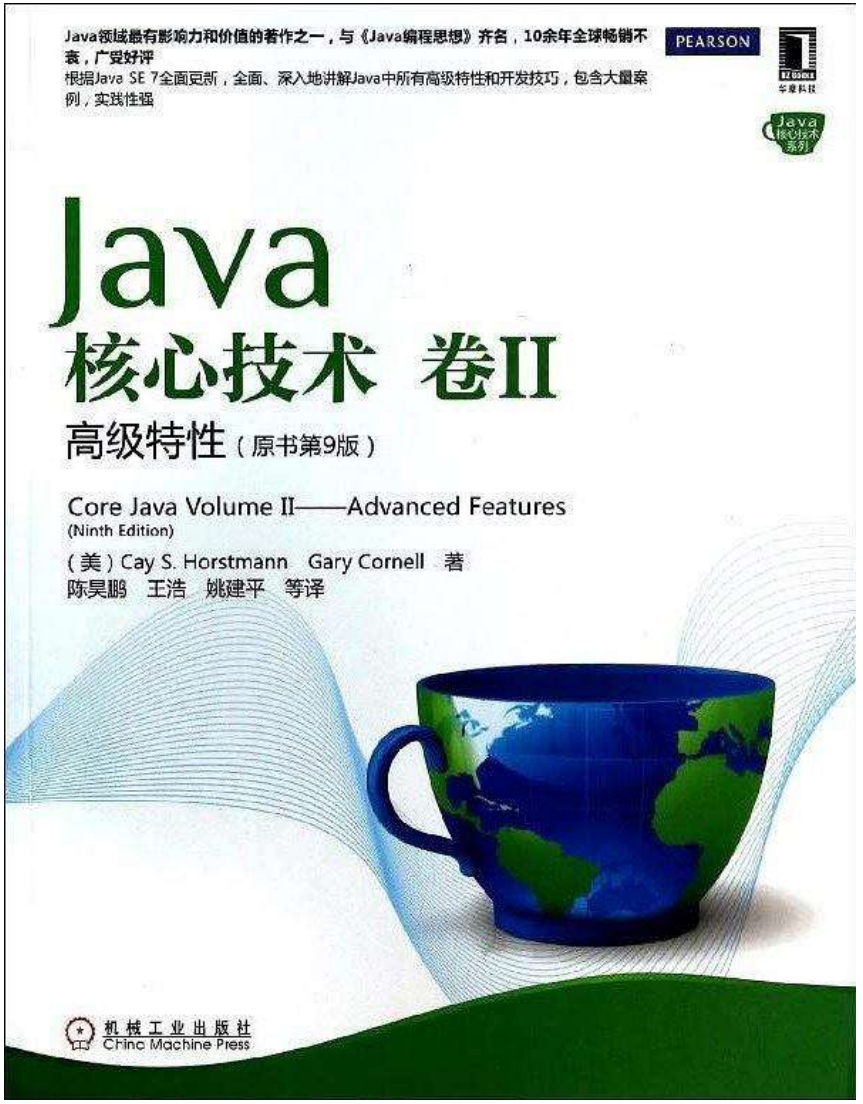
Date	Topic
Tue, Aug 27	Course introduction and course infrastructure
Wed, Aug 28	rec 1 Introduction to course infrastructure
Thu, Aug 29	Introduction to Java
Tue, Sep 3	Design for change, information hiding
Wed, Sep 4	rec 2 Unit testing, continuous integration
Thu, Sep 5	Specification and unit testing
Tue, Sep 10	Design for reuse: Delegation and inheritance
Wed, Sep 11	rec 3 Behavioral subtyping
Thu, Sep 12	Introduction to design patterns, and design patterns for reuse

Reading materials, a lot more (optional)

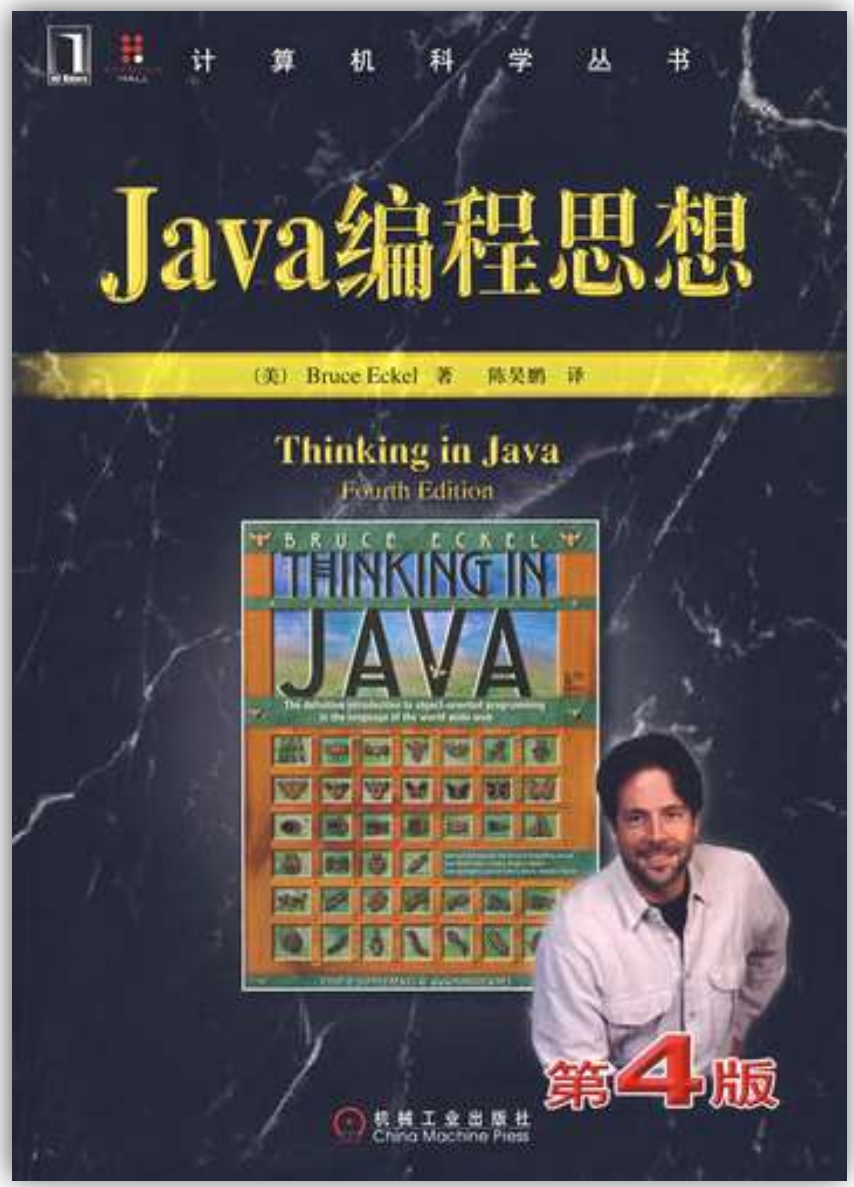
- B. Eckel. Java编程思想 (Thinking in Java), 机械工业出版社, 2016.
- J. Bloch. Effective Java 中文版, 机械工业出版社, 2009.
- GoF. 设计模式：可复用面向对象软件的基础 (Elements of Reusable Object-Oriented Software), 机械工业出版社, 2017.
- S. McConnell. 代码大全 (Code Complete), 电子工业出版社, 2006.
- R. Martin. 代码整洁之道 (Clean Code: A Handbook of Agile Software Craftsmanship), 人民邮电出版社, 2010.
- R. Pressman. 软件工程--实践者的研究方法 (Software Engineering: A Practitioner's Approach, 7th edition), 机械工业出版社, 2011.
- B. Goetz. Java Concurrency in Practice, Addison-Wesley, 2006.
- A. Oram, G. Wilson. 代码之美 (Beautiful Code), 机械工业出版社, 2009.

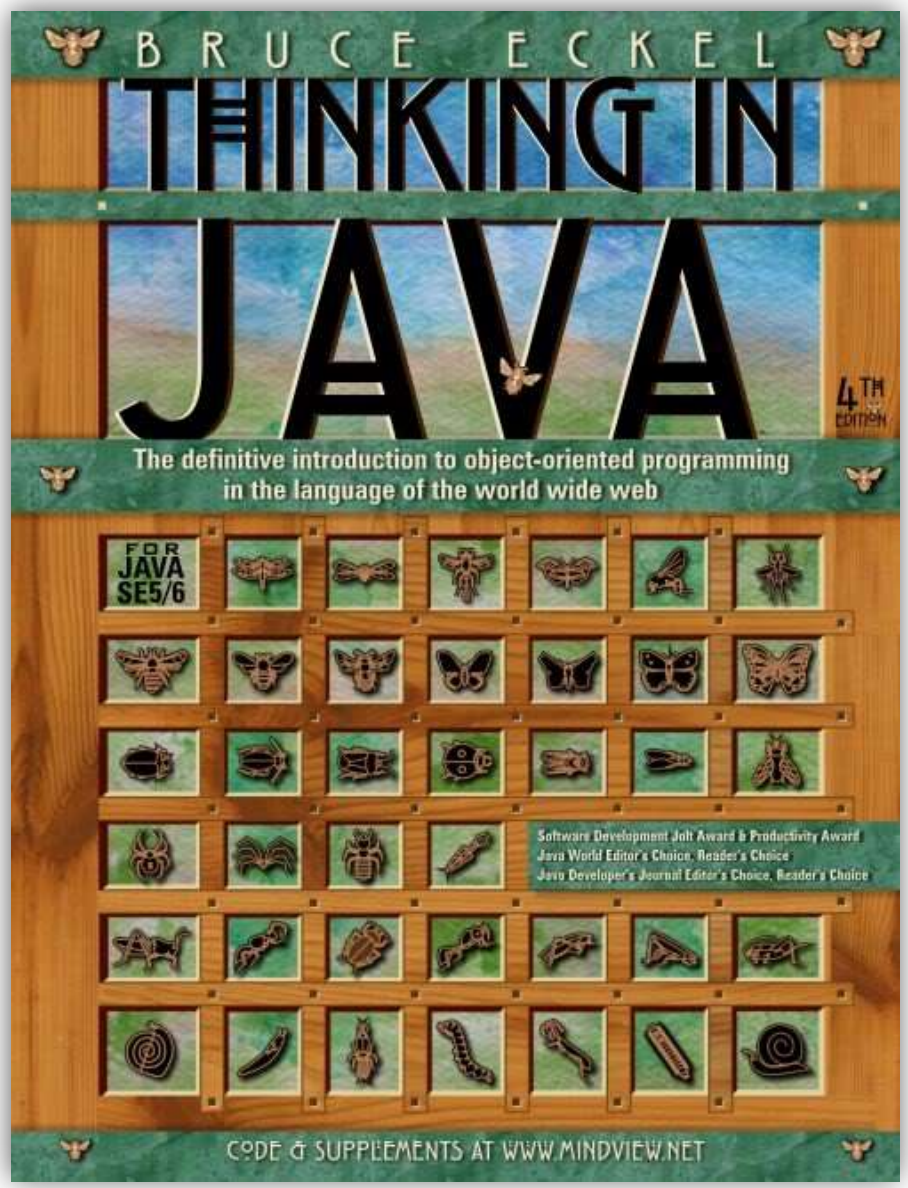
If your prefer to have a textbook





Or


计 算 机 科 学 丛 书
Java编程思想
(美) Bruce Eckel 著 陈昊鹏 译
Thinking in Java
Fourth Edition
BRUCE ECKEL
THINKING IN
JAVA
第4版
机械工业出版社
China Machine Press


BRUCE ECKEL
THINKING IN
JAVA
4TH EDITION
The definitive introduction to object-oriented programming
in the language of the world wide web
FOR
JAVA
SE5/6
Software Development Jolt Award & Productivity Award
Java World Editor's Choice, Reader's Choice
Java Developer's Journal Editor's Choice, Reader's Choice
CODE & SUPPLEMENTS AT WWW.MINDVIEW.NET

Grading policy

- **雨课堂测试：5%**
  - 课堂上使用手机雨课堂工具参与答题；
- **个人博客：5%**
  - 针对教师提出的讨论问题进行课后阅读，或对实验进展过程遇到的问题和经验教训进行总结思考，以网上公开博客的形式发表见解；
  - 2月28日（第一周周五），所有人通过以下链接提交个人博客信息，超期关闭文档链接，教师不再查看。
- **实验：30%**
  - 共4个，均为个人完成；
  - 提交实验报告/实验代码至GitHub；
- **期末考试：60%**
  - 闭卷

提交个人博客信息

About Personal Blog

- 根据课堂上学习的理论方法、教师提出的讨论问题，查阅相关资料，进行系统化的思考；
- 对实验进展过程遇到的问题和经验教训进行总结思考；
- 撰写个人博客，公开发表；
- 评判标准：
  - 博客数量
  - 与课程内容/实验内容的相关性
  - 博客内容的深度与独特性
  - 个人思考与借鉴网上公开资料的比例
  - 上课期间博客发布的时间均衡程度
- 截止日期：期末考试前一天23:55

Using "Rain Classroom" for in-class tests

- 学生带手机到教室，微信里关注**“雨课堂”公众号**或打开**“雨课堂”小程序**；
- 扫描二维码加入**“软件构造”课程**；
- **所有讲义**通过雨课堂发布。疫情期间，通过微信群发布；
- 上课过程中，教师在讲完某些知识点后会通过雨课堂发布**小测验**，以了解学生的学习效果。
- 进入雨课堂公众号，点击“我的”-“课程”，找到右上角的“我”，点击用户头像编辑个人信息，**填写真实姓名、学号、邮箱地址，否则影响成绩。**

- **评判标准：雨课堂自动打分。**

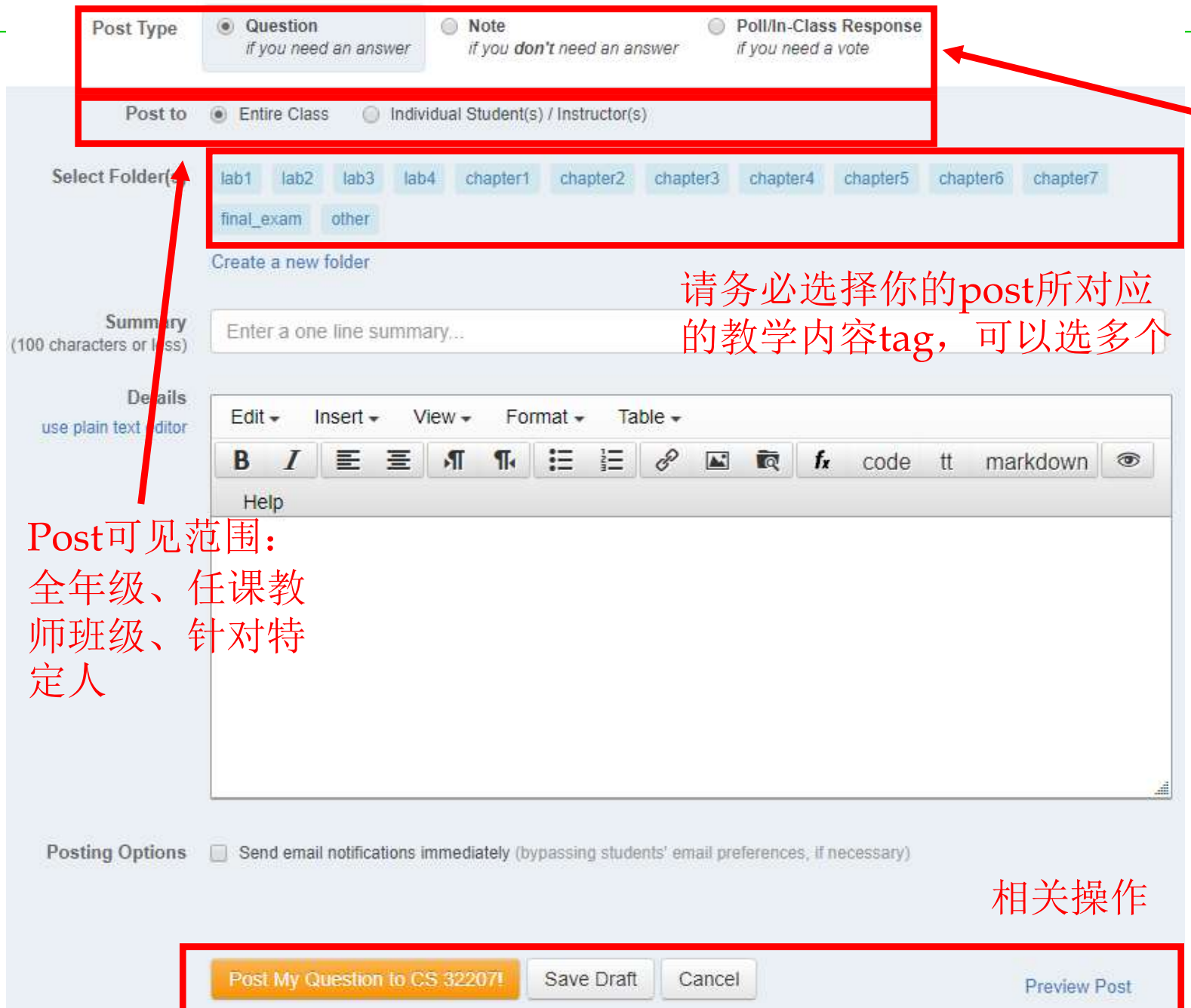
Using Piazza for after-class Q&A and discussion

- 到piazza.com注册账号，注册后请实名，TA不回答非实名的问题；
- 访问以下地址加入课程：

  http://piazza.com/harbin_institute_of_technology/spring2020/cs32207
- **Class access code: cs32207**

- 关于各章节内容、各实验内容、个人博客、期末考试等的任何问题，请在**Piazza**提出。
- 欢迎其他学生回答技术类问题。

Using Piazza for after-class Q&A and discussion



三种类型Post：

- 提问
- 笔记（分享经验体会）
- 投票（获取其他人的看法）

请务必选择你的post所对应的教学内容tag，可以选多个

Post可见范围：全年级、任课教师班级、针对特定人

相关操作

About labs

- 共4个实验，均为单人完成；
- 28学时实验课，课上+课后完成；
- 按照提交时间、代码/模型的质量、实验报告的质量进行打分；
- 4次成绩加权平均，得到总成绩。
- **Deadline: 各截止周的周日夜间23:55 (GitHub)**

序号	实验内容	覆盖章节	实验课时间	提交截止日期	分数
1	Java编程基础/测试/构建基础	第1/2/6章	1-3周	第3周	15%
2	ADT/OOP	第3章	4-6周	第7周	30%
3	代码可维护性/扩展/复用	第4/5章	8-12周	第12周	40%
4	健壮性/调试	第6章	13-15周	第15周	15%

About labs

- **在Java+Eclipse+Git环境下进行，通过GitHub Classroom提交：**
  - https://classroom.github.com，具体参见实验指导手册Lab Guidelines
  - 实验提前1周开放，学生可提前准备好相应的开发环境，熟悉开发任务，实验课上以开发+Q&A为主；
  - 代码和实验报告 (Word或PDF格式) 只需要提交至GitHub仓库，请确保完整的开发历史都在仓库里，而不仅仅只是最终结果。
  - 不接收通过Email、微信等其他方式的提交，TA无权私下接收实验结果；
  - 进行抄袭检测，若有抄袭出现，双方均无成绩。
- **TA抽查学生代码，若发现雷同代码，视为抄袭他人；**
- **TA课后阅读学生提交的实验报告，结合自动测试、人工评判代码，进行打分。**

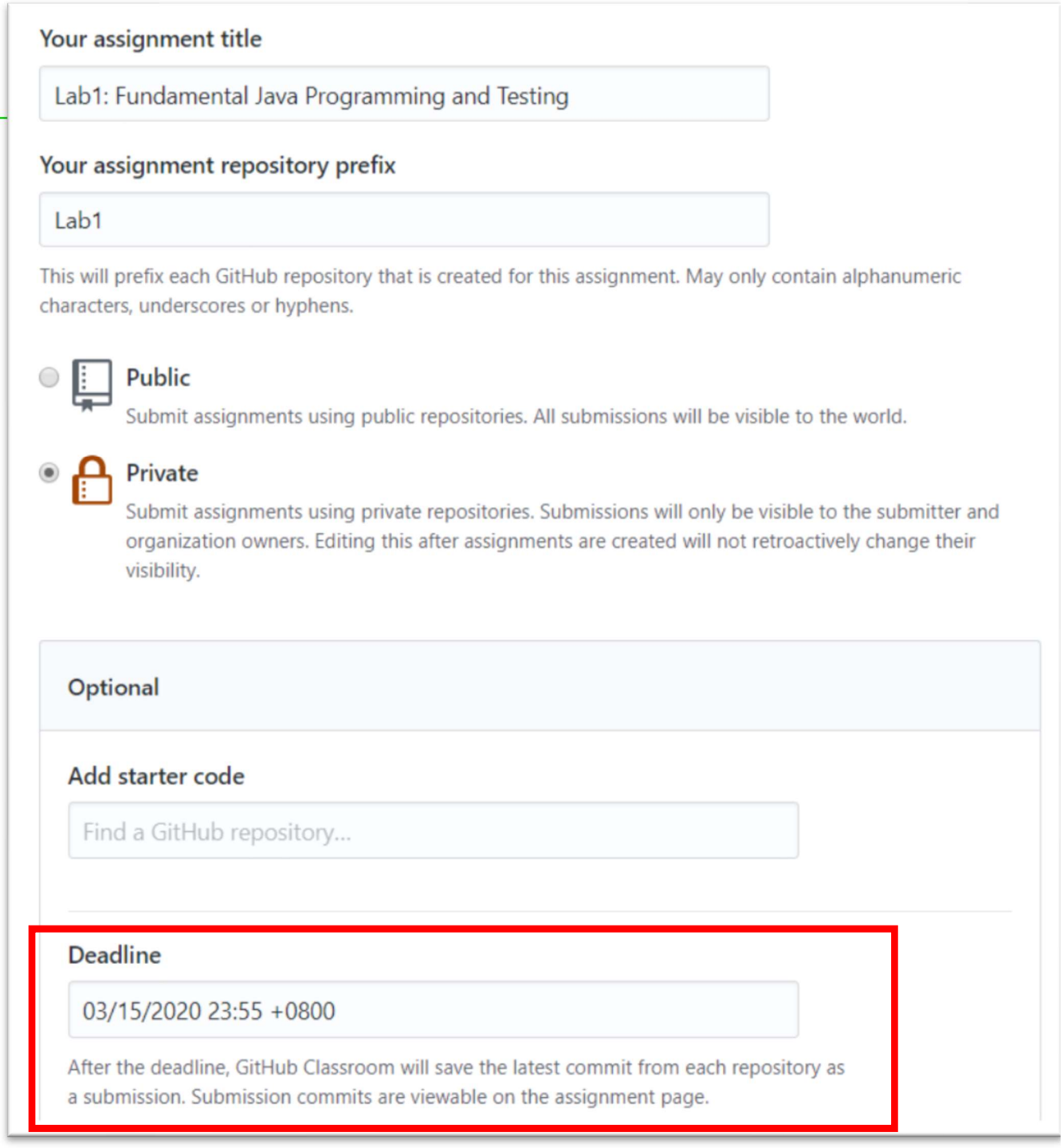
Late day policy for labs

- 请尽可能在**deadline**之前提交。但是——
  - 每人共有4天的free late days，以“天”为单位计算（迟交1秒和24小时均耗费1个free late day）；
  - 如果free late days已用完，后续实验每延迟1天，分数扣除40%；
  - 不管有没有剩余free late days，每次实验最多延迟2天，意即：超过deadline 48小时，分数=0。
- 以下是一个例子：

实验	Deadline	实际提交日期	剩余free late days	分数计算
Lab1	3月15日 23:55	3月16日 23:56	2	100%
Lab2	4月12日 23:55	4月14日 23:54	0	100%
Lab3	5月17日 23:55	5月19日 23:56	0	0 （迟交3天）
Lab4	6月 7日 23:55	6月 8日 23:54 6月 9日 23:54	0 0	60%（迟交1天） 20%（迟交2天）

GitHub Classroom

- 在**GitHub Classroom**里，作业**deadline**都设定为周日晚上**23:55**。
- 超时仍可继续向**GitHub**仓库**commit**，**Classroom**自动将最后一次**commit**作为作业结果交付给教师/**TA**，根据最后一次**commit**的时间计算**late days**。
- 所以，在确定提交作业后请不要再次**push**！



TAs for labs

- **TA:**
  - 李楠 977272991@qq.com 英才班
  - 李伟枫 970983393@qq.com 软件工程班

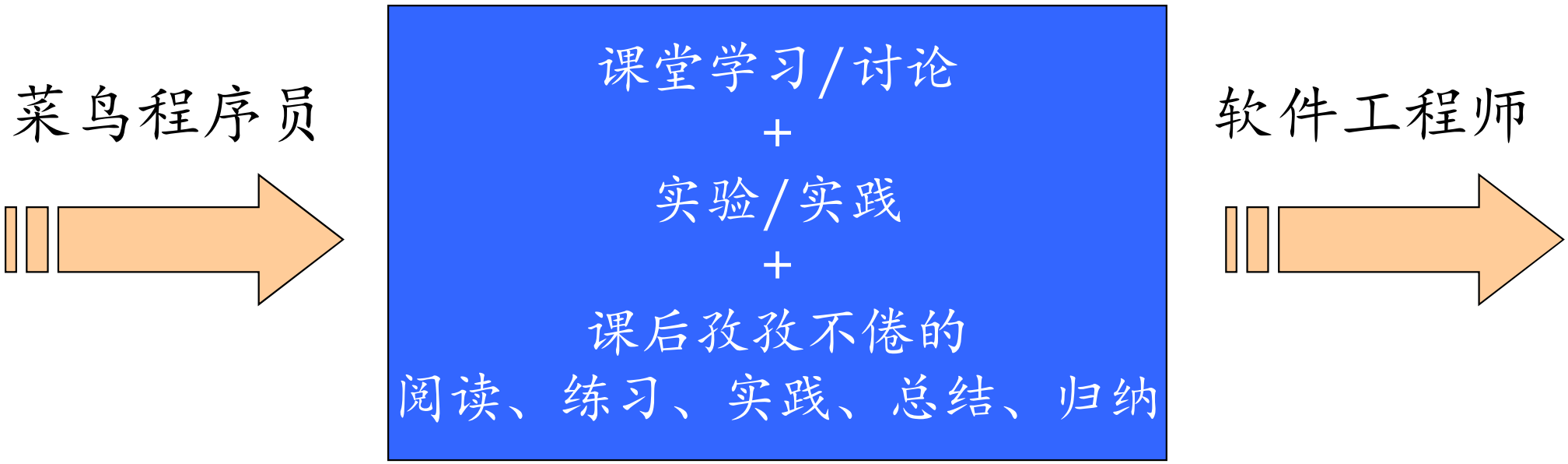
- **实验课期间，与TA面对面交流 (疫情期间，教师和TA在线直播答疑)**
- **其他时间，通过Piazza平台进行Q&A**
- **考虑到TA自己具有的学习和研究任务，时间紧张，非实验课时间恕不接待通过微信或Email等手段的提问**

A summary of tools used in this course

- **GitHub Classroom：** 获取实验链接，创建私人Git仓库，提交代码和实验报告至仓库；TA从各人的Git仓库获取最后一次commit进行评价打分，计算late days
- **Piazza：** 与本年级其他班级学生构成Q&A论坛，教师、TA、学生均可在其中提问和回答
- **雨课堂：** 微信公众号/小程序；获取讲义、实验要求；教师讲课期间发布随堂测试题，学生现场做答，便于教师了解学习情况
- **微信群：** 日常交流

To summarize your own best practices

- 多动手、多实践，方可成为合格的“程序员”；
- 实践越多、写的代码越多、参与的项目越大，积累经验越多；
- 首先遵循他人提出的“最佳实践”，进而创造自己的“最佳实践”；
- 从“菜鸟程序员”成长为“软件工程师”。



How to learn and get a high score

课程日历

- 时刻关注课程日历，了解课程的整体进度安排，尤其是各实验的上课时间、现场检查时间、提交时间；
  - 提前搭建好实验环境，学习实验所用的工具，提前开始实验，实验课上用于与TA的交流，答疑解惑，并接受验收。
  - 单纯使用实验课学时无法完成实验。
- 提前阅读下一次课程的待讲授内容，阅读教材相关章节，进行预习；
  - “需要我学习的知识，老师未必会在课堂上去讲”
  - 课堂上讲思想和难点，仅靠听课无法获得全部的考试点
  - 需要阅读大量的辅助教材
- 对下一次课要讲的内容，提前阅读资料做好准备。

To be a pragmatic programmer

持续深入某个技术领域，坚持学习和总结

不断积累自己的代码

github
SOCIAL CODING

stackoverflow

与其他程序员交流经验和教训

记录你的成绩和经历

LinkedIn

The end

February 21, 2020