Northwestern Polytechnical University

Theoretical Course Syllabus

Course Title: Object Oriented Programming

1. Course Information

1) Course code: U10M12004
2) Course title: 面向对象程序设计（英）/Object Oriented Programming
3) Hours and credits: 40 hours/2.5 credits
4) Prerequisite courses: C Programming Language
5) Course offered by: School of Computer Science
6) Starting semester: Spring
7) Course category: Discipline elementary course
8) Course Description: (in Chinese and English)

面向对象程序设计（OOP）是现代软件开发中一种主流的编程架构。由于其要求开发人员着重关注在要操纵的对象，而不是操纵它们所需要的逻辑，OOP 往往更加适用于大型、复杂的程序开发。得益于其封装、抽象、继承、多态等特性，OOP 以及相关的编程语言取得了广泛的关注与应用。本课程旨在帮助本科生通过学习 JAVA 编程语言掌握解决现实世界中复杂问题所需的面向对象编程技能。

Object-oriented programming (OOP) is one of the popular programming paradigms in modern software development. OOP is well-suited for large and complex programs, as it focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. OOP has been an incredible

success because of features like encapsulation, abstraction, inheritance, polymorphism, etc. This course is designed to help students learn the JAVA programming language and obtain OOP skills that are both required to solve sophisticated problems in the real world.

9) Textbooks and Reference Books

Textbooks:

Cay Horstmann, *Core Java Volume I-Fundamentals 11th Edition*, Pearson, 2018.

Reference Books:

Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide 2nd Edition*, Addison-Wesley Professional, 2017.

Joshua Bloch, *Effective Java 3rd Edition*, Addison-Wesley Professional, 2017.

2. Teaching Objectives

1) O1: Learn Java programming language, including types, operators, program control, and several useful classes.
2) O2: Develop problem-solving skills through practice and understanding of the divide-and-conquer and top-down approaches.
3) O3: Learn the principles of OOP in Java with the usage of classes, inheritance, polymorphism, interfaces, containers, and with the goal of understanding code reuse and building scalable programs.
4) O4: Use UML tools to visualize a system design.

3. Teaching Contents and Requirements

Chapter 1. Introduction (4 hours)

- 1.1 Introduction to the Course
- 1.2 Introduction to Java

Chapter 2. The Java Programming Environment (2 hours)

- 2.1 Installing the JDK
- 2.2 Using the Command-Line Tools
- 2.3 Using the Eclipse IDE

Chapter 3. Fundamental Programming Structures in Java (8 hours)

- 3.1 Data Types
- 3.2 Variables and Constants
- 3.3 Operators
- 3.4 Strings
- 3.5 Basic Input and Output
- 3.6 Control Flow
- 3.7 Big Numbers
- 3.8 Arrays

Chapter 4. Object and Classes (6 hours)

- 4.1 Introduction to OOP
- 4.2 Using Predefined Classes
- 4.3 Defining Your Own Classes
- 4.4 Static Fields and Methods
- 4.5 Method Parameters
- 4.6 Object Construction
- 4.7 Packages

Chapter 5. Inheritance (6 hours)

- 5.1 Classes, Superclasses, and Subclasses
- 5.2 Object: The Cosmic Superclass
- 5.3 Generic Array Lists
- 5.4 Object Wrappers and Autoboxing

Chapter 6. Interfaces, Lambda Expressions, and Inner Classes (5 hours)

- 6.1 Interfaces
- 6.2 Lambda Expressions
- 6.3 Inner Classes

Chapter 7. Exceptions (3 hours)

- 7.1 Dealing with Errors
- 7.2 Catching Exceptions
- 7.3 Tips for Using Exceptions

Chapter 8. Collections (4 hours)

- 8.1 Java Collections Framework
- 8.2 Concrete Collections
- 8.3 Maps

Chapter 9. I/O (2 hours)

- 9.1 I/O Streams
- 9.2 Reading and Writing Binary Data
- 9.3 Object I/O Streams and Serialization
- 9.4 Working with Files

4. Ideological and Political Education

1) Objectives:
    a) Understand that science and technology is the first productivity and the mission of software engineers.
    b) Courage to face technical challenges for the future and complex systems.
    c) Learn to analyze real-world software problems with OOP.
2) Teaching Contents:
    a) Introduce the history of OOP and Java.
    b) Learn how to use Java programming language.

c) Know how to design real-world software through divide-and-conquer and top-down approaches.

3) Teaching Methods:

a) Example illustration and code analysis.

b) Group project and examinations.

5. Course Sections and Hours

Chapters	**Chapter title**	**In-Class Teaching hours**	**Off-Class Self-study hours**	**Notes**
Chapter 1	Introduction	4	4	
Chapter 2	The Java Programming Environment	2	2	
Chapter 3	Fundamental Programming Structures in Java	8	8	
Chapter 4	Object and Classes	6	6	
Chapter 5	Inheritance	6	6	
Chapter 6	Interfaces, Lambda Expressions, and Inner Classes	5	5	
Chapter 7	Exceptions	3	3	
Chapter 8	Collections	4	4	
Chapter 9	I/O	2	2	
Total		40	40	

6. Assessment

Score composition	Components of assessment	Score (or percentage)	Assessment rules	Teaching objectives
10% Attendance +	Attendance	10%	Student attendance	Encourage students to join in the course on time.

20% Midterm Quiz + 20% Assignments + 50% Final Exam	Midterm Quiz	20%	Questions that cover the content of the first five chapters	Learn Java programming language and some principles of OOP.
	Assignments	20%	Two tasks (programming and UML designs)	Develop problem-solving skills in OOP.
	Final Exam	50%	Questions that cover the whole chapters	Understand Java programming language and the principles of OOP.

7. Information of practical teaching contents containing in this course

This course mainly introduces the fundamental theories about object-oriented programming with Java programming language. It does not have in-class labs. Thus, the students should also select the course "Experiment of Object Oriented Programming", which covers the practical teaching contents of this course.

8. Correlation between course contents and talent training objectives

No.	Objectives	Course Contents	Expected Training Outcomes
1	O1	Chapter 1, 2, 3	1. Learn Java programming language, including types, operators, program control, and several useful classes. 2. Develop problem-solving skills through practice and understanding of the divide-and-conquer and top-down approaches. 3. Learn the principles of OOP in Java with the usage of classes, inheritance, polymorphism, interfaces, containers, and with the goal of understanding code reuse and building scalable programs. 4. Use UML tools to visualize a system design.
2	O2	Chapter 4, 5, 6, 7, 8, 9	
3	O3	Chapter 4, 5, 6	
4	O4	Chapter 4, 5, 6	

Signature of Course Head Teacher: Helei Cui（崔禾磊）

Review Opinions of the Expert Group: Xiaochun Tang（汤小春）

Signature of Expert Group Leader:

Date of Review:

Review Opinion of the Teaching Committee of the College/Department:

Signature of Teaching Committee Leader（Unit Seal）:

Date of Review: