

OBJECT ORIENTED PROGRAMMING

1. Thông tin về học phần (General Information)

Tên học phần (Course name): Object Oriented Programming

Mã học phần (Course code): INT1332

Số tín chỉ (Number of credits): 3

Loại học phần (Course type): Compulsory

Học phần tiên quyết (Prerequisites):

Introduction to computing and programming

Học phần trước (Previous courses):

Học phần song hành (Parallel courses):

Các yêu cầu đối với học phần (Course requirements):

- Lecture room: Projector, microphone, speaker, and air conditioner.
- Laboratory: Computers with NetBeans or Eclipse, JDK

Giờ tín chỉ đối với các hoạt động (Teaching and Learning hours):

- Lectures (lí thuyết): 30h
- Exercises (bài tập): 0h
- Projects (bài tập lớn): 08h
- Lab (thực hành): 07h
- Individual reading (tự đọc): 0h

Địa chỉ Khoa/Bộ môn phụ trách học phần (Address of the Faculty/Department in charge of the course):

- Address: Faculty of Information Technology 1 - Posts and Telecommunications Institute of Technology, Km10, Nguyen Trai Street, Ha Dong District, Hanoi.
- Phone number: (024) 33510432

2. Mục tiêu học phần (Objectives)

Về kiến thức (Knowledge):

On completion of this course, students will be able to develop a program with object-oriented specifications such as classes, objects, interface and so on. The advanced knowledge is about programming with graphical user interface and I/O. Students are also able to solve a real problem with the Java programming language.

Kỹ năng (Skills):

On successful completion of this course, students will:

- understand of the basic characteristics of object-oriented methodology;
- have basic object-oriented programming skills with Java language;
- be able to apply object-oriented programming method with Java to solve the practical problems.

Thái độ, Chuyên cần (Attitude):

Learners are required to attend the classes and complete assignments/projects.

3. Tóm tắt nội dung học phần (Description)

This course provides fundamental knowledge of object-oriented programming and advanced programming skills with the Java programming language. Students will be equipped with object-oriented methodology such as concept formulation, class modelling and fundamentals of object modelling technique. This course also provides students with basic to advanced skills of Java programming language.

4. Nội dung chi tiết học phần (Outlines)

Chapter 1 Introduction to Java programming

- 1.1. Introduction to Java
- 1.2. Java programming steps
- 1.3. Compile and run
- 1.4. Java terminology and syntax
- 1.5. Java program template
- 1.6. Output via System.out.println() and System.out.print()
- 1.7. A simple program
- 1.8. Java program
- 1.9. Variables and operations
 - 1.9.1. Variables
 - 1.9.2. Data types
 - 1.9.3. Constants
 - 1.9.4. Type conversion
 - 1.9.5. Assignment
 - 1.9.6. Basic arithmetic operations
- 1.10. Questions and exercises

Chapter 2 Java basics

- 2.1. Basic syntaxes
 - 2.1.1. Steps in writing a Java program
 - 2.1.2. Java program template
 - 2.1.3. A sample program illustrating sequential, decision and loop constructs
 - 2.1.4. Comments
 - 2.1.5. Statements and blocks
 - 2.1.6. White spaces and formatting source code
- 2.2. Variables and types
 - 2.2.1. Variables - name, type and value
 - 2.2.2. Identifiers (or names)
 - 2.2.3. Variable declaration
 - 2.2.4. Constants (final variables)
 - 2.2.5. Expressions
 - 2.2.6. Assignment (=)
- 2.3. Primitive types and string
 - 2.3.1. Built-in primitive types
 - 2.3.2. Integers vs. floating-point numbers
 - 2.3.3. Data representation
 - 2.3.4. Maximum/minimum values of primitive number types
 - 2.3.5. One more important type - string
 - 2.3.6. Choice of data types for variables
 - 2.3.7. Literals for primitive types and string

- 2.3.8. var - local variable type inference (JDK 10)
- 2.4. Basic operations
 - 2.4.1. Arithmetic operators
 - 2.4.2. Arithmetic expressions
 - 2.4.3. Type conversion in arithmetic operations
 - 2.4.4. More on arithmetic operators
 - 2.4.5. Overflow/underflow
 - 2.4.6. More on integer vs. floating-point numbers
 - 2.4.7. Type casting
 - 2.4.8. Compound assignment operators
 - 2.4.9. Increment/decrement
 - 2.4.10. Relational and logical operators
 - 2.4.11. String and '+' concatenation operator
- 2.5. Flow control
 - 2.5.1. Sequential flow control
 - 2.5.2. Conditional flow control
 - 2.5.3. Exercises on getting started and conditional
 - 2.5.4. Loop flow control
 - 2.5.5. Terminating program
 - 2.5.6. Exercises on decision and loop
- 2.6. Input/output
 - 2.6.1. Formatted output via “printf()” (JDK 5)
 - 2.6.2. Input from keyboard via “Scanner” (JDK 5)
 - 2.6.3. Code examples
 - 2.6.4. Exercises on decision/loop with input
 - 2.6.5. Input from text file via “Scanner” (JDK 5)
 - 2.6.6. Formatted output to text file
 - 2.6.7. Input via a dialog box
 - 2.6.8. java.io.console (JDK 1.6)
- 2.7. Writing correct and good programs
 - 2.7.1. Programming errors: compilation, runtime and logical errors
 - 2.7.2. Debugging programs
 - 2.7.3. Exercises on decision/loop with input
 - 2.7.4. Testing your program for correctness
- 2.8. More on loops - nested-loops, break & continue
 - 2.8.1. Nested loops
 - 2.8.2. break and continue - interrupting loop flow
- 2.9. Questions and exercises

Chapter 3 Strings, arrays and methods

- 3.1. String and char operations
 - 3.1.1. char arithmetic operations
 - 3.1.2. Converting char to int
 - 3.1.3. String operations
 - 3.1.4. Converting string to primitive
 - 3.1.5. Converting primitive to string
 - 3.1.6. Formatting strings - string.format()
 - 3.1.7. The StringBuilder, StringBuffer and StringTokenizer classes
 - 3.1.8. Java Regex
- 3.2. Arrays
 - 3.2.1. Array index
 - 3.2.2. Array's length
 - 3.2.3. Array and loop
 - 3.2.4. Enhanced for-loop (or “for-each” Loop) (JDK 5)

- 3.2.5. Code examples: Read and print array
- 3.2.6. Multi-dimensional array
- 3.3. Methods (Functions)
 - 3.3.1. Why methods?
 - 3.3.2. Using methods
 - 3.3.3. The "return" statement
 - 3.3.4. The "void" return-type
 - 3.3.5. Actual parameters vs. formal parameters
 - 3.3.6. Pass-by-value for primitive-type parameters
 - 3.3.7. Pass-by-reference for arrays and objects
 - 3.3.8. Method overloading
 - 3.3.9. "boolean" methods
 - 3.3.10. Mathematical methods
- 3.4. Command-line arguments
 - 3.4.1. Code example: arithmetic
 - 3.4.2. Exercises on command-line arguments
- 3.5. Bitwise operations
 - 3.5.1. Bitwise logical operations
 - 3.5.2. Bit-shift operations
 - 3.5.3. Types and bitwise operations
- 3.6. Questions and exercises

Chapter 4 Introduction to object-oriented programming

- 4.1. Why OOP?
- 4.2. OOP in Java
 - 2.1. Class & instances
 - 2.2. A Class is a 3-compartment box encapsulating data and operations
 - 2.3. Class definition in java
 - 2.4. Creating instances of a class
 - 2.5. Dot (.) operator
 - 2.6. Member variables
 - 2.7. Member methods
 - 2.8. Putting them together: An OOP example
 - 2.9. Constructors
 - 2.10. Revisit method overloading
 - 2.11. The access control modifiers: public/private
 - 2.12. Information hiding and encapsulation
 - 2.13. The public getters/setters for private variables
 - 2.14. Keyword "this"
 - 2.15. Method toString()
 - 2.16. Constants (final)
 - 2.17. Putting them together in the finalized circle class
- 4.3. Examples on classes
- 4.4. Abstraction - thinking in object
- 4.5. Encapsulation
- 4.6. Class relationship
- 4.7. Questions and exercises

Chapter 5 Composition, inheritance and polymorphism

- 5.1. Composition
- 5.2. Inheritance
 - 5.2.1. Inheritance EG. 1: The circle and cylinder classes
 - 5.2.2. Method overriding & variable hiding
 - 5.2.3. Annotation @Override (JDK 1.5)

- 5.2.4. Keyword "super"
- 5.2.5. More on constructors
- 5.2.6. Default no-arg constructor
- 5.2.7. Single inheritance
- 5.2.8. Common root class - java.lang.Object
- 5.2.9. Examples
- 5.3. Composition vs. inheritance
- 5.4. Polymorphism
 - 5.4.1. Substitutability
 - 5.4.2. Polymorphism examples
 - 5.4.3. Upcasting & downcasting
 - 5.4.4. The "instanceOf" operator
 - 5.4.5. Summary of polymorphism
- 5.5. Abstract classes & interfaces
 - 5.5.1. The abstract method and abstract class
 - 5.5.2. Abstract class examples
 - 5.5.3. The Java's interface
 - 5.5.4. Interface examples
 - 5.5.5. Implementing multiple interfaces
 - 5.5.6. Interface formal syntax
 - 5.5.7. Why interfaces?
 - 5.5.8. Interface vs. abstract superclass
 - 5.5.9. Dynamic binding or late binding
- 5.6. Additional object-oriented design issues
 - 5.6.1. Encapsulation, coupling & cohesion
 - 5.6.2. "is-a" vs. "has-a" relationships
 - 5.6.3. Program at the interface specification, not the implementation
- 5.7. Questions and exercises

Chapter 6 Exception handling and I/O

- 6.1. Exception handling
 - 6.1.1. Introduction to exception handling
 - 6.1.2. Method call stack
 - 6.1.3. Exception & call stack
 - 6.1.4. Exception classes - Throwable, Error, Exception & RuntimeException
 - 6.1.5. Checked vs. unchecked Exceptions
 - 6.1.6. Exception handling operations
 - 6.1.7. try-catch-finally
 - 6.1.8. Common exception classes
 - 6.1.9. Creating your own exception classes
- 6.2. Assertion (JDK 1.4)
- 6.3. Java I/O
 - 6.3.1. The file class
 - 6.3.2. File input and output
 - 6.3.3. Character stream
 - 6.3.4. Binary stream
- 6.4. Questions and exercises

Chapter 7. Programming Graphical User Interface (GUI)

- 7.1. Introduction
- 7.2. Programming GUI with AWT
 - 7.2.1. AWT packages
 - 7.2.2. Containers and components
 - 7.2.3. AWT container classes

- 7.2.4. AWT component classes
- 7.2.5. AWT examples
- 7.3. AWT event-handling
- 7.4. Nested (Inner) classes
- 7.5. Event listener's adapter classes
- 7.6. Layout managers and panel
 - 7.6.1. FlowLayout
 - 7.6.2. GridLayout
 - 7.6.3. BorderLayout
 - 7.6.4. Using Panels as sub-container to organize components
 - 7.6.5. BoxLayout
- 7.7. Swing
 - 7.7.1. Introduction
 - 7.7.2. Swing's features
 - 7.7.3. Using swing API
 - 7.7.4. Swing program template
 - 7.7.5. Swing examples
- 7.8. Using visual GUI builder - NetBeans/Eclipse
 - 7.8.1. NetBeans
 - 7.8.2. Eclipse
- 7.9. Questions and exercises

Chapter 8 Generics and collections

- 8.1. Generic types and methods
- 8.2. Bounded type Parameters
- 8.3. Generics, inheritance, and subtypes
- 8.4. Wildcards
 - 8.5.1. Upper bounded wildcards
 - 8.5.2. Unbounded wildcards
 - 8.5.3. Lower bounded wildcards
 - 8.5.4. Wildcards and subtyping
 - 8.5.5. Wildcard capture and helper methods
- 8.5. Java collections framework overview
- 8.6. List, stack, queue
- 8.7. Sets and maps
- 8.8. Algorithms
- 8.9. Questions and exercises

5. Học liệu (Textbooks)

5.1. Học liệu bắt buộc (Required Textbooks)

[1]. 1. Y. Daniel Liang. Introduction to Java Programming. 10th edition, Prentice Hall, 2015.

5.2. Học liệu tham khảo (Optional Textbooks)

[2]. H. Schildt. Java: The Complete Reference. 11th edition, McGraw-Hill, 2018. Education.

[3]. The Java Tutorials, Oracle Java Documentation (<https://docs.oracle.com/javase/tutorial/>).

6. Phương pháp, hình thức kiểm tra – đánh giá kết quả học tập học phần (Grading Policy)

Grading method	Percentage	Group/Individual
- Attendance	10%	Individual

- Exercises	20%	Individual
- Mid-term projects/exams	20%	Group or individual
- Final examination (lab)	50%	Individual

Trưởng Bộ môn
(Head of Department)

Giảng viên biên soạn
(Lecturer)

Nguyễn Mạnh Hùng

Hoàng Hữu Hạnh