Nguyen Ngoc Minh

Saigon Institute of Technology
Readings

- Readings for these lecture notes
  - Textbook:
    - Tony Gaddis & Godfrey, *Starting Out with Java From Control Structures through Data Structures*
  - Reference:
    - Cay S. Horstmann and Gary Cornell, *Core Java™*

- These lecture notes contain material © Tony Gaddis, and Godfrey Muganda, 2007
Chapter Objectives

To understand:

• The necessity of programming,
• The need for programming languages
• The advantages of Java language
• Elements common to all programming languages
• Role of the Java compiler and Java Virtual Machine
• Steps involved in creating programs
• How to create, compile, and run a simple Java program
Outline

• Introduction to Computer
• Introduction to Java
• Programming Process
• Methods of Programming
Introduction

Why program?

• Computer is a versatile tool used by many professions.

• It can perform many different jobs because they can be programmed.

• Computer programmers implement programs, or software to instruct the computer how to perform specific jobs.
Introduction

What’s a program?

• A program is a set of instructions that a computer follows in order to perform a task.

• These instructions are called an algorithm.

• An algorithm is a set of well defined steps to completing a task.

• The steps in an algorithm are performed sequentially.
Introduction
Programming language

- A computer only understands the algorithm written in machine language.
- Machine language is written using binary numbers.
- The binary numbering system (base 2) only has two digits (0 and 1).
- Example of a machine language instruction: 1011010000000101
- It’s very difficult to implement an algorithm by using machine language.
Introduction
Programming language

• Programming languages were invented to ease the task of programming.

• What is programming language?
  • A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.

• Two kinds of programming language:
  • Low level language
  • High level language
Introduction
Programming language

- Low level programming languages are very close to machine languages
  ➡️ hard to read and understand.
- Ex: Assembly language
  
  ```
  mov edx, [esp+8+1]
  cmp edx, 0
  ja @f
  mov eax, 0
  ret
  ```

- High level languages are close to human language
  ➡️ easy to understand and implement algorithm.
- Ex: Java
  
  ```java
  if (a > b) print("a is greater than b")
  ```
**Introduction**

Common Language Elements

- Key words
- Operators
- Punctuation
- Programmer-defined identifiers
- Strict syntactic rules.

```java
public class HelloWorld {
    public static void main(String[] args) {
        String message = "Hello World";
        System.out.println(message);
    }
}
```
Outline

• Introduction to Computer
• **Introduction to Java**
• Programming Process
• Methods of Programming
Java Language

What’s java?

- Java is a high level programming language developed by Sun Microsystems

- Java derives much of its syntax from C and C++ but has a simpler object model

- Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM)
Java Language
Why java?

• Five primary goals in the creation of the Java language:
  • It should use the object-oriented programming methodology.
  • It should allow the same program to be executed on multiple operating systems.
  • It should contain built-in support for using computer networks.
  • It should be designed to execute code from remote sources securely.
  • It should be easy to use by selecting what were considered the good parts of other object-oriented languages.

(source:http://java.sun.com/docs/white-langenv/Intro.doc2.html)
Java Language
Java application and applet

• Two types of Java programs:
  • Applications
    • Stand-alone programs that run without the aid of a web browser.
    • Relaxed security model since the user runs the program locally.
  • Applets
    • Small applications that require the use of a Java enabled web browser to run.
    • Enhanced security model since the user merely goes to a web page and the applet runs itself.
Java Language
The Compiler

• Programming statements written by the programmer are called **source code**.

• A **text editor** is used to edit and save a source code file.

• A **compiler** translates source code file into another file holding the translated instructions.

• **Syntax errors** may be detected during translation process.
The Compiler

- Most compilers translate source code into executable files containing machine code.
- The Java compiler translates a Java source file (.java file) into a file that contains byte code instructions (.class file).
- Byte code instructions are the machine language of the Java Virtual Machine (JVM) and cannot be directly executed directly by the CPU.
Java Language
Java Virtual Machine

• The JVM is a program that emulates a micro-processor.
• The JVM executes instructions as they are read.
• JVM is often called an interpreter.
• Java is often referred to as an interpreted language.
• Each type of computer has a Java Virtual Machine developed for it.
  Ex: JVM for Windows, Linux, Solaris, etc.
Java Language
Program Development Process

Text editor

Saves Java statements

Java compiler

Is read by

Produces

Source code (.java)

Is interpreted by

Byte code (.class)

JVM

Text editor saves Java statements, which are then read by the Java compiler to produce byte code (.class). Byte code is interpreted by the JVM.
Java Language

Portability

- **Portable** means that a program may be written on one type of computer and then run on a wide variety of computers, with little or no modification.

- With most programming languages, portability is achieved by the creation of a compiler for each platform. Source code is compiled into machine code for a specific platform.

- With Java, there are many implementations of the JVM on different hardware and operating system platforms. Source code is compiled into byte code **independent of platform**.

- Compilers are much more difficult to develop than interpreters.
Java Language
Portability

Byte code (.class)

- Java Virtual Machine for Windows
- Java Virtual Machine for Linux
- Java Virtual Machine for Unix
- Java Virtual Machine for Macintosh
Java Language
Java Versions

• Java began at version 1.0 and is now at version 5.0 (Sun skipped from 1.4 to 5.0).
• With the advent of version 1.2, Java became Java2 because it provided much more functionality.
• Java2 version 5.0 can still compile Java 1.0 programs as long as no features of any other version of Java are present.
• This is called backwards compatibility.
Java Language

Java Versions

• Java began as the Java Development Kit (JDK).
• With the advent of Java2, through version 1.4 it changed to the Java Software Development Kit (SDK)
• In Java 5, JDK is back
• There are different editions of Java:
  • J2SE - Java2 Standard Edition.
  • J2EE - Java2 Enterprise Edition.
Java Language
Compiling and running a Java Program

• The Java compiler is a command line utility.

• The command to compile a java file is:
  
  javac filename.java

  javac is the Java compiler.
  
  Ex: To compile a java source code file named HelloWorld.java you would use the command:

  javac HelloWorld.java

• The command to run a java program is:
  
  java filename

  Ex: java HelloWorld
Outline

• Introduction to Computer
• Introduction to Java
• Programming Process
• Methods of Programming
Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.
4. Check the model for logical errors.
5. Enter the code and compile it.
6. Correct any errors found during compilation.
7. Run the program with test data for input.
8. Correct any runtime errors found while running the program.
9. Validate the results of the program.
Programming Process

Example

• We will go through steps to create a program solving the problem below:

• Problem:

  Develop a program to calculate grade point average of a student. Assuming that the student study only 2 courses: Math, and English.
Programming Process

Example

• Step1: Clearly define what the program is to do.

Example:

_PURPOSE_: Calculate the student’s GPA

_INPUT_: Grade of each course

_PROCESS_: Add grade of English & Math. Divide the total for 2. The result is the student’s GPA

_OUTPUT_: Display a message indicating the GPA.
Programming Process

Example

• Step 2: Visualize the program running on the computer.
  • Imagine what the computer screen will look like
  • Draw pictures of the screen with sample input and output

Example:

```
Please enter your Math’s grade: 9
Please enter your English’s grade: 8
Your GPA is: 8.5
```
Programming Process

Example

• Step 3: Use design tools to create a model of the program.
  • Use pseudocode to design an algorithm.

Example:

```
Display “Please enter your Math's grade:”
Input grade
Display “Please enter your English's grade:”
Input grade
Calculate GPA = (Math + English) / 3
Display “Your GPA is:”
```
Programming Process

Example

- Step4: Check the model for logical errors.

  Display “Please enter your Math's grade:”
  
  Input grade
  
  Display “Please enter your English's grade:”
  
  Input grade
  
  Calculate GPA = (Math + English) / 3
  
  Display “Your GPA is:”
Programming Process

Example

• Step5: Enter the code and compile it.

```java
import java.util.Scanner;
public class GPACalculation {

    public static void main(String[] args) {
        double mathGrade;
        double englishGrade;
        double totalGrade;
        double gPA;

        Scanner keyboard = new Scanner(System.in);
        System.out.print("Please enter your Math's grade: ");
        mathGrade = keyboard.nextDouble();
        System.out.print("Please enter your English's grade: ");
        englishGrade = keyboard.nextDouble();
        totalGrade = mathGrade + englishGrade;
        gPA = totalGrade / 2;
        System.out.println("Your GPA is: "+ gPA);
    }
}
```
Programming Process
Example

• Step 6: Correct any compile time errors found during compilation.
  • A compile-time error is an error found at compile time.
  • They are usually syntax errors.
Programming Process

Example:

• Step 7: Run the program with test data for input.

<table>
<thead>
<tr>
<th>Math</th>
<th>English</th>
<th>GPA (expected result)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>5.5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>9.8</td>
</tr>
</tbody>
</table>
Programming Process

Example

• Step 8: Correct any runtime errors found while running the program.
  • A run-time error is an error that occurs while the program is running
  • They are usually logical errors or mathematical mistakes
Programming Process

Example

- Step 9: Validate the results of the program.
  - Check with test data and determine if the program solves the original problem.
Outline

• Introduction to Computer
• Introduction to Java
• Programming Process
• Methods of Programming
Methods of Programming

• Two primary methods of programming: procedural and object-oriented programming.

• Older programming languages were procedural.

• Java is an object-oriented language.
Procedural Programming

• A procedure is a set of programming language statements that, together, perform a specific task.

• Procedures typically operate on data items that are separate from the procedures.

• In a procedural program, the data items are commonly passed from one procedure to another.
Procedural Programming

Data Element

Procedure A

Procedure B
Object-Oriented Programming

• Object-oriented programming is centered on creating objects rather than procedures.
• Objects are a melding of data and procedures that manipulate that data.
• A program is simply a collection of objects that interact with each other to accomplish a goal.
• Data in an object are known as attributes.
• Procedures in an object are known as methods.
Object-Oriented Programming

Object

Attributes (data)

Methods (behaviors / procedures)
Object-Oriented Programming

Classes and Objects

• A class is a collection of programming statements that define the required object.
• An object is the realization (instantiation) of a class in memory.
• Classes can be used to instantiate as many objects as are needed.
• Each object that is created from a class is called an instance of the class.
The *Insect* class defines the attributes and methods that will exist in all objects that are instances of the Insect class.

The housefly object is an instance of the Insect class.

The mosquito object is an instance of the Insect class.