AP Computer Science A: Java Programming

Zheng-Liang Lu

Department of Computer Science & Information Engineering
National Taiwan University

APcomSciA 285
Summer 2017
Class Information

• The class website,  
  http://www.csie.ntu.edu.tw/~d00922011/APcomSciA.html,  
  is for the schedule, homework assignments, announcements, etc.
  • Lecture notes will be uploaded right before class.
  • Note that the lecture notes are organized in English.

• You can contact me, if necessary, by
  • d00922011@csie.ntu.edu.tw, or
  • Facebook.
Prerequisites

• This class is organized for students who are not EE/CS majors.
• No programming experience required; it would be helpful if you have some programming experiences.
• May involve with high school math in examples.
• I promise to keep everything simple in this class.\(^1\)

\(^1\)“Simple is not easy. ... Easy is a minimum amount of effort to produce a result. ... Simple is very hard. Simple is the removal of everything except what matters. ...” See here.
Teaching Philosophy

- First, I try to lower the barriers to entry.
- Second, I provide resources as many as possible.
- Third, I answer your questions.
Learning Tips

• Start with just one language and master it.
• Ask lots of questions; Google first.
• Practice makes permanent (and hopefully, perfect).
• It may take 10000 hours, more or less; it is never too late.
• Grasp the fundamentals for long-term benefits; code from the bottom.
• Code by hand.²

²It sharpens proficiency and you’ll need it to get a job. For example, technical interview of Google.
“Knowledge is of no value unless you put it into practice.”

– Anton Chekhov (1860-1904)

“Many roads lead to the path, but basically there are only two: reason and practice.”

– Bodhidharma
Grading Policy

To acquire the certificate, you need at least **70 pts** at the end of class:

- **Programming assignments (30%)**
  - Would be 5 to 10 questions
  - Practice makes perfect.

- **Final exam (70%)**
  - On-site programming
  - 4 problems
  - 2 hours
  - Open everything
class Lecture1 {
    "Introduction"
}

// Keywords:
public, class, static, void
What Is Programming?

Programming is the activity of writing a sequence of instructions to tell a machine to perform a specific task.

- A sequence of instructions → program
- A set of well-defined notations is used to write a program → programming language
- The person who writes a program → programmer designer

Writing codes is not what the CS people work for. We are writing codes to make a better world.
WHAT MY MOM THINKS I DO

WHAT MY FRIENDS THINK I DO

WHAT SOCIETY THINKS I DO

WHAT ARTISTS THINK I DO

WHAT I THINK I DO

WHAT I ACTUALLY DO

[JavaScript Application]

Hello, world!

OK

WHY WON'T YOU COMPILE?
Deep Learning

What society thinks I do

What my friends think I do

What other computer scientists think I do

What mathematicians think I do

What I think I do

What I actually do

photo credit: here.
In Practice

Programming is to provide a solution to a real-world problem using computational models supported by programming languages.

- The computational solution is a program.
A program is a sequence of instructions, written in an artificial language, to perform a specified task with a machine.

- They are almost everywhere.
- For example,
  - Computer virus
  - Video games
  - Operating systems
  - ATM, traffic light, Google search engine, recommendation system...

- Differences: goal, functionality, algorithms, complexity...
How and Where The Programs Run

- The programs are activated from the disk into the main memory.
- Now we call them the processes.
- CPUs contain the arithmetic and logic unit (ALU) and the registers.
  - ALU is responsible for the computational power.
  - Registers store the data to be used temporarily.
- The outputs are written back to the main memory and further stored into the disk if necessary.

---

3 The “process” is a formal terminology used in operating systems.
4 CPUs have only limited number of registers.
5 You may refer to any class for an introduction to computer system. For example, Introduction to Computer Science & Programming in C.
Figure 1.9  An example of a memory hierarchy.

See Figure 1-9 in Bryant, p. 14.
A **programming language** is an artificial language to communicate with machines.

Recall how you learned the 2nd nature language when you were a kid.

Programming languages → **syntax and semantics**
- Used to express algorithms
- Used to control the behavior of machines

How many programming languages in the world?
- More than 1000.
- Top 20 programming languages can be found in [TIOBE](https://www.tiobe.com/tiobe-index/).
- Java: top 3

Note that every language originates from reasons.
History

- 1st generation: machine code
- 2nd generation: assembly code
- 3rd generation: high-level programming languages
- Post 3rd generations
- Java is one of the 3rd-generation programming languages.

---

High-level language program (in C)

```java
swap(int v[], int k)
{
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compiler

Assembly language program (for MIPS)

```assembly
swap:
multi $2, $5, 4
add $2, $4, $2
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
jr $31
```

Assembler

Binary machine language program (for MIPS)

```
0000000001010001000000000100011000
000000001000000010001000010000010001
100110111100010000000000000000000000
100111100001001000000000000000000100
1010110000010010000000000000000000
10101101111001000000000000000001000
00000011111000000000000000000001000
```
1st-Generation Programming Languages

- Computers understand instructions only in binary, which is a sequence of 0’s and 1’s. (Why?)
- Each computer has its own set of instructions.\(^8\)
- So the programs at the very early stage were machine-dependent.
- These are so-called the machine language, aka machine code.
- Pros:
  - Most efficient for machines
- Cons:
  - Hard to program for human
  - Not portable
- Still widely used in programming lower level functions of the system, such as drivers, interfaces with firmware and hardware.

\(^8\)For example, X86 and ARM.
2nd-Generation Programming Languages

• An assembly language uses mnemonics to represent instructions as opposed to the machine codes.
• Hence, the code can be read and written by human programmers.
• Yet, it is still machine-dependent.

• To run on a computer, it must be converted into a machine readable form, a process called assembly.
More often used in extremely intensive processing such as games, video editing, graphic manipulation/rendering.

Note that machine languages and assembly languages are also known as low-level languages.
3rd-Generation Programming Languages

- High-level programming languages use English-like words, mathematical notation, and punctuation to write programs.
- They are closer to human languages.
- Pros:
  - Portable, machine-independent
  - Human-friendly
- For example, C\textsuperscript{10}, C++\textsuperscript{11}, and Java\textsuperscript{12}.

\begin{flushright}
\textsuperscript{10}Dennis Ritchie (1973)\\
\textsuperscript{11}Bjarne Stroustrup (1983)\\
\textsuperscript{12}James Gosling (1995)
\end{flushright}
• Note that the machines understand and execute only the machine codes as before.

• The translation is accomplished by a compiler, an interpreter, or a combination of both.\(^{13}\)

\(^{13}\)If you’ve learned C, you should take a look at the design of compiler.
What Can A Program Do?

- A **program** is an implementation of an **algorithm** expressed in a specific **programming language**.

![Diagram of the process from Real World Problem to Program](image-url)
Algorithms In A Nutshell

- An algorithm is a well-defined computational procedure that takes a set of values as input and produces a set of values as output.
- Simply put, an algorithm is a procedure that solves a particular class of problems, such as a cookbook.
Properties of Algorithms\textsuperscript{15}

• An algorithm must possess the following properties\textsuperscript{14}:
  • **Input** and **output**
  • **Correctness**
  • **Definiteness**: basic instructions provided by a machine, e.g. $+\, -\, \times\, \div$.
  • **Effectiveness**: action which can be completed by combination of basic instructions.
  • **Finiteness**: resource requirement, especially **time** and **space**.

• Note that an algorithm is not necessarily expressed in a specific programming language.
  • Could use human languages, graphs, and **pseudo codes**.

\textsuperscript{14}Alan Turing (1912–1954)
\textsuperscript{15}Donald E. Knuth (1938–)
Example

• Organize an algorithm that finds the greatest element in the input list, say A.

**Input:** A (a list of $n$ numbers)

**Output:** max (the greatest element in A)

• Can you provide a procedure to determine the greatest element? For all situations?
My Solution

- The first element of A can be fetched by calling A(1).
- Let ← be the assignment operator in the following pseudo code.

```plaintext
max ← A(1)
for i ← 2 ~ n
    if A(i) > max
        max ← A(i)
    end
end
return max
```

- How to find the minimal element?
- How to find the location of the greatest element?
- Why not max ← 0?
“Computers are good at following instructions, but not at reading your mind.”

– Donald Knuth (1938-)

“Computer science is no more about computers than astronomy is about telescopes.”

– Edsger Wybe Dijkstra (1930-2002)

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.”

– Tony Hoare (1934-)
Alan Turing

• Provided a formalization of the concepts of *algorithm* and *universal computation model* for general-purpose computers.

• The central subject of study in computing theory is *Turing machine*:
  • Toy example from Google
  • Also proved that there exist problems which are *undecidable* by Turing machines.\(^{16}\)

• Father of *computing theory and artificial intelligence*\(^{17}\)

• **Turing Award** of ACM\(^{18}\)

• **The Imitation Game** (2014)

---

\(^{16}\)See *Halting problem*.

\(^{17}\)See *Turing test*.

\(^{18}\)Association for Computing Machinery
Alan Turing
What Is Java?

• Java is a general purpose programming language.
• It has features to support programming based on the object-oriented paradigms.
• The initial version of the Java platform was released by Sun Microsystems in 1995.\textsuperscript{19}
• At the very early stage, this language was called Oak and it was meant to be used in set-top boxes for televisions.
• Slogan: “Write once, run anywhere.”
• That is, Write a Java program once and run it on any platform. (How?)

\textsuperscript{19}Now owned by Oracle Corporation, since January 2010.
Java Virtual Machine (JVM)\textsuperscript{22}

Java Virtual Machine (JVM) is used to run the bytecodes on each platform.

\begin{itemize}
\item JVM is a program, not a physical machine.
\item The job of JVM is to translate the bytecodes into machine codes according to the platform it is running on.\textsuperscript{20}
\item To enhance the security, the JVM verifies all bytecodes before the program is executed.\textsuperscript{21}
\item “No user program can crash the host machine.”
\end{itemize}

\textsuperscript{20}Herein we mean operating systems, say Linux.
\textsuperscript{21}However, there are a number of possible sources of security vulnerabilities in Java applications. See \url{here}.
\textsuperscript{22}See \url{JVM}.
Compiling and Running A Java Program

See Figure 2-19 in Sharan, p. 59.
An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.

- An IDE normally consists of a source code editor, build automation tools and a debugger.
- Most modern IDEs offer the intelligent code completion.

In this class, we need **Java Development Kit (JDK)** and **Eclipse IDE for Java Developers**.
Example: A Simple Template

Write a program which says hello to Java.

```java
public class HelloJava {
    public static void main(String[] args) {
        // Print "Hello, Java." on the screen.
        System.out.println("Hello, Java.");
    }
}
```

Keywords are marked in violet.

- **class**: declare a new class followed a distinct class name
- **public**: can be accessed by any other class
- **static**: can be called without having to instantiate a particular instance of the class
- **void**: do not return a value
• Every statement ends with a semicolon (;).
• A special method `main` is used as the entry point of the program.
• `System.out` refers to the standard output device, normally the screen.
• `println()` is a method within `System.out`, which is automatically imported by default.
Public Classes

The public keyword is one of access modifiers\(^{24}\), which allows the programmer to control the visibility of classes and also members.

- One public class in the java file whose filename is identical to that of the public class.
- There must be at most one public class in one java file.

\(^{24}\)We will visit the access control later when it comes to encapsulation.
How To Run A Java Program

Source code (developed by the programmer)

```java
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

```
... Method Welcome()
  0 aload_0 ...

Method void main(java.lang.String[])
  0 getstatic #2 ...
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 ...
  8 return
```

“Welcome to Java” is displayed on the console

Welcome to Java!

---

25 See Figure 1.14 in YDL, p.20.
Table of Special Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>Opening and closing braces</td>
<td>Denote a block to enclose statements.</td>
</tr>
<tr>
<td>()</td>
<td>Opening and closing parentheses</td>
<td>Used with methods.</td>
</tr>
<tr>
<td>[]</td>
<td>Opening and closing brackets</td>
<td>Denote an array.</td>
</tr>
<tr>
<td>//</td>
<td>Double slashes</td>
<td>Precede a comment line.</td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>Opening and closing quotation marks</td>
<td>Enclose a string (i.e., sequence of characters).</td>
</tr>
<tr>
<td>;</td>
<td>Semicolon</td>
<td>Mark the end of a statement.</td>
</tr>
</tbody>
</table>

See Table 1.2 in YDL, p.18.
Bugs

A bug is an error, flaw, failure, or fault in a computer program or system, producing an incorrect or unexpected result, or misbehaving in unintended ways.

- **Compile-time error**: most of them are syntax errors
- **Runtime error**: occurs when Java program runs, e.g. 1/0
- **Logic error**: introduced by the programmer by implementing the functional requirement incorrectly

Note that logic errors are the obscurest in programs since they are hard to be found.
“If debugging is the process of *removing* software bugs, then programming must be the process of *putting* them in.”

Good programming style and proper documentation make a program easy to read and help programmers prevent errors.

- **Indentation** is used to illustrate the *structural* relationships between a program’s components or statements.
- In a long program, you should also include *comments* that introduce each major step and explain anything that is difficult to read.
- A block is a group of statements surrounded by curly braces:
  - next-line style
  - end-of-line style
- For example, [Google Java Style](https://google.github.io/styleguide/jspuide.html).
class Lecture2 {
    "Data types, Variables, and Operators"
}

// Keywords:
byte, short, int, long, char, float, double, boolean, true,
false, import, new
Example

Given the radius of a circle, say 10, determine the area.

Recall that a program comprises data and algorithms.

- How to store the data?
  → variables, data types

- How to compute the area?
  → arithmetic operators

- How to show the result?
  → System.out.println()
public class ComputeArea {
    public static void main(String[] args) {
        // input
        int r = 10;
        // algorithm
        double area = r * r * 3.14;
        // output
        System.out.println(area);
    }
}

- The type int and double are two of primitive data types.
- We use two variables r and area.
Variable $\approx$ Box
Variable Declaration

- You give a name for the variable, say x.
- Additionally, you need to assign a type for the variable.
- For example,

```
...  
1 int x; // x is a variable declared an integer type. 
2 ... 
```

- Variable declaration tells the compiler to allocate appropriate memory space for the variable based on its data type.\(^{27}\)
- It is worth to mention that, the data type determines the size, which is measured in bytes\(^{28}\).

---

\(^{27}\)Actually, all declared variables are created at the compile time.

\(^{28}\)1 byte = 8 bits; bit = binary digit
Naming Rules

- Identifiers are the names that identify the elements such as variables, methods, and classes in the program.
- The naming rule excludes the following situations:
  - cannot start with a digit
  - cannot be any reserved word\(^{29}\)
  - cannot include any blank between letters
  - cannot contain +, −, *, / and %
- Note that Java is case sensitive\(^{30}\).

\(^{29}\)See the next page.

\(^{30}\)The letter A and a are different.
Reserved Words\textsuperscript{31}

<table>
<thead>
<tr>
<th>abstract</th>
<th>double</th>
<th>int</th>
<th>super</th>
</tr>
</thead>
<tbody>
<tr>
<td>assert</td>
<td>else</td>
<td>interface</td>
<td>switch</td>
</tr>
<tr>
<td>boolean</td>
<td>enum</td>
<td>long</td>
<td>synchronized</td>
</tr>
<tr>
<td>break</td>
<td>extends</td>
<td>native</td>
<td>this</td>
</tr>
<tr>
<td>byte</td>
<td>final</td>
<td>new</td>
<td>throw</td>
</tr>
<tr>
<td>case</td>
<td>finally</td>
<td>package</td>
<td>throws</td>
</tr>
<tr>
<td>catch</td>
<td>float</td>
<td>private</td>
<td>transient</td>
</tr>
<tr>
<td>char</td>
<td>for</td>
<td>protected</td>
<td>try</td>
</tr>
<tr>
<td>class</td>
<td>goto</td>
<td>public</td>
<td>void</td>
</tr>
<tr>
<td>const</td>
<td>if</td>
<td>return</td>
<td>volatile</td>
</tr>
<tr>
<td>continue</td>
<td>implements</td>
<td>short</td>
<td>while</td>
</tr>
<tr>
<td>default</td>
<td>import</td>
<td>static</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td>instanceof</td>
<td>strictfp*</td>
<td></td>
</tr>
</tbody>
</table>

\textsuperscript{31}See Appendix A in YDL, p. 1253.
Variable as Alias of Memory Address

- The number 0x000abc26 stands for one memory address in hexadecimal (0-9, and a-f).\(^{32}\)
- The variable \(x\) itself refers to 0x000abc26 in the program after compilation.

Data Types

• Java is a strongly typed programming language.
• Every variable has a type.
• Also, every (mathematical) expression has a type.
• There are two categories of data types: primitive data types, and reference data types.

\[33\text{You cannot change the type of the variable after declaration.}\]
Primitive Data Types

- Primitive Data Type
  - boolean
  - Numeric Type
    - Integral Type
      - byte
      - short
      - int
      - long
      - char
    - Floating-Point Type
      - float
      - double

---

See Figure 3-4 in Sharan, p. 67.
## Integers

<table>
<thead>
<tr>
<th>Name</th>
<th>Width</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>64</td>
<td>$-9,223,372,036,854,775,808$ to $9,223,372,036,854,775,807$</td>
</tr>
<tr>
<td>int</td>
<td>32</td>
<td>$-2,147,483,648$ to $2,147,483,647$</td>
</tr>
<tr>
<td>short</td>
<td>16</td>
<td>$-32,768$ to $32,767$</td>
</tr>
<tr>
<td>byte</td>
<td>8</td>
<td>$-128$ to $127$</td>
</tr>
</tbody>
</table>

- The most commonly used integer type is `int`.
- If the integer values are larger than its feasible range, then a **overflow** occurs.