CS-101 – WEEK 01 – INTRO TO PROGRAMMING AND JAVA

GIUSEPPE TURINI

Kettering
UNIVERSITY

TABLE OF CONTENTS

STUDY GUIDE

Study Material:

- These slides.
- Your notes.
- "Java Illuminated (5th Ed.)", chapter 1, pp. 1-38. *

Practice Exercises:

- Exercises from *: 1.7.1.1-14, 1.7.2.15-20, 1.7.3.21-30, 1.7.5.

Additional Resources:

- "Java Illuminated (5th Ed.)", appendix D, pp. 1153-1155.
- "Java Illuminated (5th Ed.)": appendix E, pp. 1157-1160.

INTRODUCTION TO CODING

Programming: Creating code instructing a computer to solve a specific problem.
For example: a ticketing kiosk, a game, a restaurant ordering system, etc.

Programming is a science. Software engineering practices ensure that programs are:

- Correct: computing the correct results or performing the correct tasks.
- Readable: easy to understand what the code does by reading the code.
- Maintainable: easy to update, and easy to extend/improve.
- Reusable: easy to use code developed to solve a problem, to solve another problem.

Programming is an exciting creative activity:

- Exciting because we solve problems!
- Creative because we create software applications from nothing.

ALGORITHM, PROGRAM, PLATFORM

Algorithm: An algorithm is a sequence of unambiguous instructions for solving a problem.

Note: Although most algorithms are intended for computer implementation, the notion of algorithm does not depend on such an assumption!

Example: A recipe for a specific cake can be considered an algorithm to solve the problem of baking that cake properly.

Program: A (computer) program is the coding of an algorithm in a specific programming language, or more in general, a sequence of instructions in a programming language that a computer can execute.

Platform: The (computing) platform used to run a program, usually including the computer hardware (HW) and its operating system (OS) and sometimes additional devices.

BASIC COMPUTER CONCEPTS

Computer Hardware (HW):

- Central Processing Unit (CPU).
- Main memory (or RAM) and other storage devices (hard drives, etc.).

Operating System (OS):

- Controls the peripheral devices (keyboard, mouse, etc.).
- Supports multiple software applications executing simultaneously.

Software application (SW):

- Coded to perform specific tasks, relies on the support of the operating system.

Computer networks:

- Networks connect 2 or more computers so they can share data or hardware.
- Local Area Networks (LANs) and the Internet.

COMPUTER HARDWARE

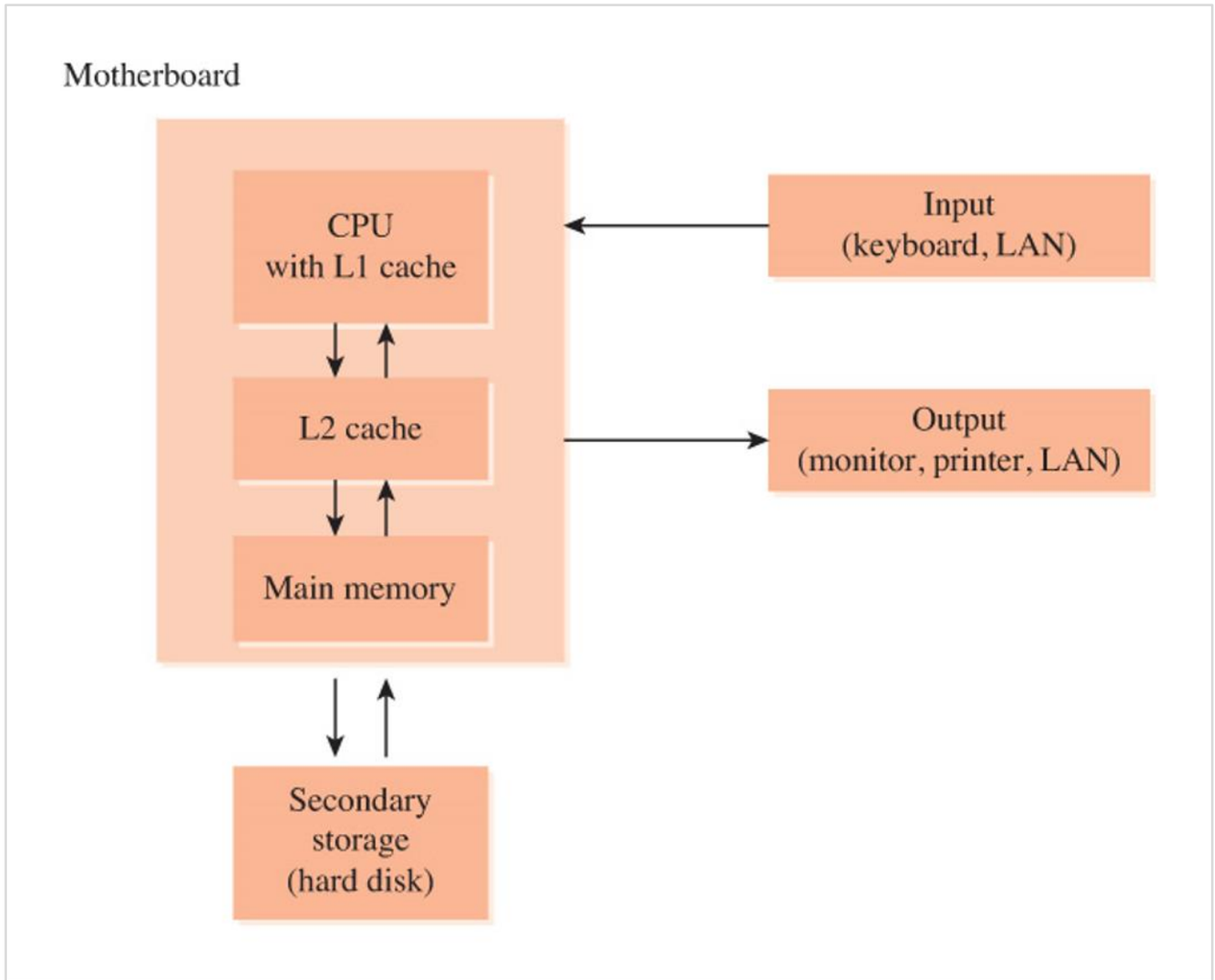
Central Processing Unit (CPU): Executes the instructions of the operating system and any software application currently running.

Main memory (RAM): Stores both instructions and data of any software application currently running. This storage is not-persistent (data is lost whenever the computer is shut down).

Hard Disk (HD or Hard Drive): Stores data into files. This storage is persistent (data is available even when the computer is shut down).

Peripheral devices: External hardware devices usually used to provide the computer with additional functionalities. For example: a keyboard to input text data, a mouse to provide point-and-click interactions, a monitor to visualize information to the user, a printer to print data on paper, etc.

COMPUTER HARDWARE: DESIGN



COMPUTER HARDWARE: CPU

Central Processing Unit (CPU): Executes the instructions of the operating system and any software application currently running.

Example: A 3.4 GHz Intel Core i3-3240 processor.

Usually a CPU includes:

- An Arithmetic Logic Unit (ALU) (or Integer Unit, IU) which performs basic integer arithmetic and logical operations.
- A Floating-Point Unit (FPU) which performs floating-point arithmetic.
- A set of hardware registers for storing data and memory addresses.
- A control unit to sequence the instructions to be executed.
- Other supporting hardware.

The instructions to be executed usually perform: arithmetic/logic operations, transfer of data from/to memory locations, changes in the execution flow/sequence of instructions.

COMPUTER HARDWARE: CPU SPEED

The CPU speed depends on its clock cycles per second, typically measured in GHz (gigahertz). This speed represents how fast a CPU can execute operations.

Example: A CPU rated at 2.5 GHz executes 2.5 billion (2500000000) clock cycles a second.

Note: Different instructions may require a different number of clock cycles to be executed.

A program (usually consisting of multiple instructions) is executed in this way:

- First, the program is loaded into memory.
- Then, each instruction is executed using this fetch-decode-execute cycle:
  - **Fetch:** Using the program counter (that tracks instruction sequence), the CPU transfers next instruction from memory to the instruction register.
  - **Decode:** The instruction is decoded (to understand its type).
  - **Execute:** The instruction is executed.
- This fetch-decode-execute cycle repeats until the program ends.

COMPUTER HARDWARE: MEMORY UNITS

Storage devices (memory, hard disks, etc.) are usually rated in terms of their capacity in bytes.

Bit: One (1) binary digit (value 0 or 1).

Byte: Eight (8) binary digits, or 8 bits.

Kilobyte (kB): About one thousand (~1000) bytes, exactly 1024 bytes (or 2^{10} bytes).

Megabyte (MB): About one million (~1000000) bytes, exactly 2^{20} bytes.

Gigabyte (GB): About one billion (~1000000000) bytes, exactly 2^{30} bytes.

Terabyte (TB): About one trillion (~1000000000000) bytes, exactly 2^{40} bytes.

Example: 3 MB of L1 cache memory, 8 GB of RAM memory (RAM), 1 TB of hard disk memory.

COMPUTER HARDWARE: MEMORY TYPES

CPU Cache Memory: Cache memory can store data only when power is applied to the memory but does not need to be refreshed to retain the data stored. This type of memory is located on the CPU chip and usually uses Static Random-Access Memory (SRAM) technology. Access speed fast, capacity small.

Main Memory (or RAM): Main memory (Random-Access Memory, or RAM) can store data only when power is applied to the memory and needs to be refreshed regularly to retain the data stored. This type of memory is located on the motherboard and usually uses Dynamic Random-Access Memory (DRAM) technology. Access speed moderate, capacity moderate.

Hard Drive Memory: HD memory does not need power to retain the data (but it needs power to change the data). This type of memory is located on storage devices connected to the motherboard. Access speed slow, capacity large.

COMPUTER HARDWARE: MEMORY USAGE

For the CPU to execute instructions at its fastest speed, instructions and data must be available on the CPU memory (cache memory) at that speed as well.

The CPU uses instructions and data stored in the L1 cache (Level-1 cache memory, very fast).

Whenever all L1 cache data has been used, the L1 cache is refilled by transferring data from the L2 cache (Level-2 cache memory, slower than L1 cache but larger) to the L1 cache.

Whenever all the L2 cache data has been used, the L2 cache is refilled by transferring data from the main memory (RAM, slower than L2 cache but larger) to the L2 cache.

Whenever all the main memory (RAM) data has been used, the RAM is refilled by transferring data from HD memory (hard drives, slower than RAM but larger) to the RAM.

OPERATING SYSTEM

The Operating System (OS) is a software application that:

- Controls the peripheral devices (keyboard, mouse, etc.).
- Supports multiple software applications executing simultaneously (multitasking).
- Allocates memory to each software application avoiding memory conflicts.
- Prevents applications and user from damaging the system.

The OS loads (boots) when the computer is turned on and the OS is intended to run until the computer is turned off.

Example: Microsoft Windows, Linux, Apple Mac OS X, etc.

SOFTWARE APPLICATIONS

A software application consists of a program coded to perform specific tasks, and it relies on the support of the operating system.

Example: a word processor, a spreadsheet, a computer videogame, an Internet browser.

COMPUTER NETWORKS

A computer network is a system connecting multiple computers to share data or hardware.

Local Area Network (LAN): System of connected computers (wireless or wired) located geographically close to one another. For example, a university network, or a cybersecurity laboratory network.

The Internet (The Web): A network of computer networks where servers deliver Internet content (webpages) to clients using Internet browsers.

Server: Computer/program providing services to other computers/programs (clients) in the same network.

Client: Computer/program using services provided by other computers/programs (servers) in the same network.

COMPUTER DATA

A computer processor (CPU) can only process binary numbers, so computer data are stored as binary numbers (but sometimes programs use hexadecimal values).

Binary System (Base 2): Binary numbers are expressed in the binary system (base 2), using only digits 0 and 1.

Decimal System (Base 10): Decimal numbers are expressed in the decimal system (base 10), using only digits 0-9.

Hexadecimal System (Base 16): Hexadecimal numbers are expressed in the hexadecimal system (base 16), using only digits 0-9 and A-F.

Example: The same value (x) expressed in binary, decimal, and hexadecimal.

value x in base 2 = 11110110 = $1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

value x in base 10 = 246 = $2\times10^2 + 4\times10^1 + 6\times10^0$

value x in base 16 = 0xF6 = $15\times16^1 + 6\times16^0$ (0x is default prefix, F represents 15).

BINARY NUMBERS

Binary System (Base 2): Binary numbers are expressed in the binary system (base 2), using only digits 0 and 1.

Example: The decimal value 246 in binary is 11110110.

$$11110110 = 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 246$$

2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
16,384	8,192	4,096	2,048	1,024	512	256	128	64	32	16	8	4	2	1

Powers of two and their decimal values.

Note: If not otherwise specified, we will always consider positive binary integers.

Note: Negative binary integers and signed binary floats are discussed in the appendices.

BINARY NUMBERS: CONVERSIONS

Binary to Decimal Conversion:

- Multiply each digit in the binary number by its associated power of two, considering the rightmost digit corresponding to 2^0.
- Add all these products together.

Decimal to Binary Conversion:

- Find the greatest power of two that is smaller or equal than the decimal number.
- Assign 1 as the binary digit associated with this power of two.
- Subtract this power of two from the decimal number.
- Repeat these steps with the remainder of the decimal number, until reaching 0.
- Assign 0 as the binary digit associated with all others powers of two (unused).

HEXADECIMAL NUMBERS

Sometimes, programs use values in base 16 (hexadecimal) (see color values in HTML)

Hexadecimal System (Base 16): Hexadecimal numbers are expressed in the hexadecimal system (base 16), using only digits 0-9 and A-F.

Example: The decimal value 246 in hexadecimal is 0xF6.

$0xF6 = 15 \times 16^1 + 6 \times 16^0$ (0x is default prefix, F represents 15) = 246

Note: The prefix 0x is only used in some programming languages (Java).

HEXADECIMAL NUMBERS (2)

Hex Digit	Binary Value	Hex Digit	Binary Value
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Hexadecimal values 0-9 and A-F and their binary equivalents.

HEXADECIMAL NUMBERS: CONVERSIONS

Binary to Hexadecimal Conversion (and vice versa):

- Substitute the appropriate hexadecimal digit for each set of 4 binary digits.
- For example:

(binary)	0001	1010	1111	1001	1011	0011	1011	1110
(hexadecimal)	1	A	F	9	B	3	B	E

TEXT CHARACTERS

A standard method to represent text characters is the Unicode character set (used in Java).
This set represents each text character using 16 bits (2 bytes).
So, the Unicode character set can encode up to 65536 (2^{16}) different characters.

Another popular (but old) method to represent text characters is the ASCII character set.
This set represents each text character using 7 bits.
So, the ASCII character set can encode up to 128 (2^{7}) different characters.

Note: For compatibility reasons, the first 128 characters in the Unicode character set are the same as in the ASCII character set.

Note: You can find the complete ASCII character set table in the appendices.

TEXT CHARACTERS (2)

Unicode Character	Decimal Value	Hex Value
NUL, the null character (a nonprintable character)	0	0000
*	42	002A
1	49	0031
2	50	0032
A	65	0041
B	66	0042
a	97	0061
b	98	0062
}	125	007D
delete (a nonprintable character)	127	007F

Some Unicode Characters and their decimal equivalents.

PROGRAMMING LANGUAGES

Programming Language: A programming language is a set of rules to converts instructions to machine code (command executable by a computer processor). The description of a programming language is usually split into 2 components: the syntax (form) and the semantics (meaning).

Several programming languages already exist, and new programming languages are being created every year: each designed with different goals (e.g., imperative languages, declarative languages, functional languages, etc.).

Programming languages can be categorized into 3 types:

- Machine language (or machine code).
- Assembly language (or assembler language, or asm).
- High-level language (C, C++, Java, C#, Python, etc.).

MACHINE LANGUAGE

A machine language uses binary codes to execute instructions and refer memory addresses.
Programming using a machine language is extremely challenging and time consuming.
Machine code is not portable to other computer architectures with different CPU.

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Example of machine code for a 32-bit architecture
(a 32-bit line represent 1 instruction).

ASSEMBLY LANGUAGE

An assembly language uses symbolic names (e.g., var1) for memory addresses and mnemonics for instructions (e.g., SW represent the instruction to store a value).

Assembly code is always converted into machine code before it is executed.

Assembly code is not portable to other computer architectures with different CPU.

```
lw  $to,    0($2)
lw  $t1,    4($2)
sw  $t1,    0($2)
sw  $t0,    4($2)
```

Example of assembly code
(a line represent 1 instruction).

HIGH-LEVEL LANGUAGES

A high-level language (C, C++, Java, etc.) are like the English language.

Coding using a high-level programming language is easier for a software developer.

High-level code is usually more portable among different computer architectures (CPU).

High-level languages can be designed for specific uses, for example: Ruby and PHP for Internet applications, Fortran for scientific computing, and COBOL for business software.

```
public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Example of high-level code (e.g., Java code).

COMPILERS AND INTERPRETERS

High-level programming languages are compiled, interpreted, or a combination of both.

Compiler: Code written in a compiled programming language (e.g., C++) needs to be converted by a compiler (a software program) into machine code (an executable program, e.g., .exe), before it can be executed. Compiled programming languages allow development of high-performance code, but the code is usually less portable.

Interpreter: Code written in an interpreted programming language (e.g., Perl) is read and converted by an interpreter (a software program) to machine code line-by-line at execution time. Interpreted programming languages allow development of more portable code, but the code is usually less performant.

Java: Java uses a combination of a compiler and an interpreter.
A Java program is first compiled into processor-independent byte code.
This byte code is then interpreted at runtime by the Java Virtual Machine (JVM).

CONFIGURING JDK AND IDE

JDK: The Java Development Kit (JDK) is a distribution of Java by Oracle. It implements the both the Java programming language and the Java Virtual Machine (JVM). Additionally, it includes the Standard Edition (SE) of the Java API.
The OpenJDK is an open-source version of the JDK (supported by Oracle).

IDE: An Integrated Development Environment (IDE) is a software application that provide features facilitating software development. Usually, an IDE is designed for a target programming language, and includes: (1) a code editor, (2) build automation tools, and (3) a debugger. Some popular IDEs are: Visual Studio (C++, C#), Eclipse (Java), etc.

Note: The official IDE used in this course is jGRASP (www.jgrasp.org). You can find additional information on the reasons why jGRASP is the course IDE in the appendices.

Note: A student can use a different IDE (e.g., Eclipse) only if the student takes responsibility in learning how to use it independently (or with minimal assistance).

JDK CONFIGURATION

You can download the JDK from the following sources:

- Official Oracle JDK webpage (www.oracle.com/java/technologies/downloads)
- OpenJDK distribution from Adoptium (adoptium.net).
- OpenJDK bundled with jGRASP (www.jgrasp.org).

Note: Please download the proper JDK considering your platform/architecture (e.g., Linux/macOS/Windows and 32-bit/64-bit).

Note: The current JDK version is 17 (January 2022), but for this course any JDK version 8 or later will be sufficient.

After you have completed the download of the JDK, simply install it on your computer.

IDE CONFIGURATION

Note: This is a quick setup for jGRASP, other IDEs have similar configurations.

You can download jGRASP from its official webpage (www.jgrasp.org).
After you have completed the download of jGRASP, simply install it on your computer.

Note: If you download/install jGRASP bundled with the OpenJDK, the IDE is ready to go.

Note: If you install jGRASP after the JDK, it should be able to configure itself automatically.

Note: If you want/need to configure jGRASP manually, launch jGRASP and use:

Settings (menu) + Compiler Settings (menu group) + Workspace (menu item)

Then select the PATH tab (and PATHS subtab) and create a new item.
The new item should store the path of your JDK binaries folder, for example:

C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot\bin

Once the new item has been created, enable/check it, and click the Apply button.

CHECKING JDK-IDE SETUP

To check your JDK-IDE setup

- Download the course code from GitHub (github.com/turinig/cs101-cs102-cs203).
- Open the file HelloJava.java in jGRASP (or your IDE).
- Compile it, and run it.
- If successful, the details of your JDK-IDE setup show up on the console.
- Otherwise, there is a JDK-IDE configuration issue.

As an alternative, check your JDK-IDE setup by compiling/running this Java code in your IDE.

```
System.out.println("Java vendor: " + System.getProperty("java.vendor"));
System.out.println("Java version: " + System.getProperty("java.version"));
System.out.println("Java install dir: " + System.getProperty("java.home"));
System.out.println("OS name: " + System.getProperty("os.name"));
System.out.println("OS vers: " + System.getProperty("os.version"));
System.out.println("OS arch: " + System.getProperty("os.arch"));
```

INTRODUCTION TO JAVA

Java uses a combination of a compiler and an interpreter.

- A Java program is first compiled into processor-independent byte code.
- This byte code is then interpreted at runtime by the Java Virtual Machine (JVM).

To run a Java program, we only need a JVM (that simulates a virtual processor).
JVMs are available for every major computing platform.

Note: Because a Java program is interpreted at runtime, usually it runs slower than the same program implemented in C++ (compiled only).

JAVA APPS, ANDROID APPS, SERVLETS

Java programs can be coded as:

- Java applications.
- Android applications.
- Java servlets.

Java Application: A Java application runs standalone on a computer.
In this course we will focus on this kind of application.

Android Application: An Android application is a Java program developed for Android devices (mobile devices running the Android operating system).

Java Servlet: A Java servlet is launched by a web server and runs on the server (without being downloaded to the client). Typically used for web applications.

PROGRAMMING BASICS

Programming means to "teach" a computer how to solve a problem, by providing the computer a sequence of instructions to reach the solution to the problem.

To do this, we have to consider that:

- We have to be able to solve the problem ourselves, before being able to teach how to solve it to a computer!
- We have to know all the operations a computer can perform, and also the instructions a computer cannot execute!
- We have to identify a sequence of instructions (the computer can perform) to solve our problem, usually by decomposing the problem into simpler sub-tasks.

Note: Please consider that this is just a simple description of "programming", and that there are more sophisticated programming techniques that you will learn in the future.

CONTROL FLOW

Sequential Processing: It consists in coding a program as a sequence of instructions which are performed in order, one after another.

The order in which instructions are executed is critical in programming!

Control Flow: The specification of the ordering of instructions in a program. It can be done mixing 4 different strategies:

- Sequential execution.
- Method/function call.
- Selection (if, switch, etc.).
- Looping (for, while, do-while, etc.).

PSEUDOCODE

To ensure the control flow is correct, usually programmers use a tool called pseudocode.

Pseudocode: A method for describing a program (instructions and control flow) by using the English language, rather than a programming language.

Example: The pseudocode for calculating the sum of 2 numbers.

```
read first number
read second number
set total to (first number + second number)
output total
```

Note: Luckily the rules for writing pseudocode are not rigid, but remember to consider that the instructions must be doable (executable) by a computer!

CONTROL FLOW: METHOD CALL

Method: In programming, we can bundle/encapsulate some code in a method. In coding a method, we have to specify its computation (instructions) and its inputs/outputs.

Method Call: Instruction used to execute the code in a method (calling a method).

Argument Passing: In a method call, we have to specify the inputs (if any) to that method (the values on which to perform the method computation.

When a method completes its computation, the execution continues at the next instruction after the method call, and the program can now use the method output/result.

Example: Pseudocode for calculating the square root of a number.

```
read a number
call the square-root-method, passing the number and receiving the result
output the result
```

CONTROL FLOW: SELECTION

Selection: A selection is used to make different choices depending on some conditions. For example: if condition is true, do this; otherwise, do that.

Example: Pseudocode for checking if a number is positive or negative.

```
read a number
if the number is greater than or equal to 0
   write “Input number is positive.”
otherwise
   write “Input number is negative.”
```

Note: The indentation of the code in the if-otherwise is used to highlight the different instructions executed depending on the result of the condition.

Note: Programmers use indentation to make it easier to see the control flow of a program!

CONTROL FLOW: LOOPING

Loop: A loop (or iteration) statement allows the repetition of some code (some instructions) until a specified condition is reached.
For example: repeat-doing this until condition is true.

Example: Pseudocode for adding a group of numbers in input.

```
set total to 0
read the first number
repeat-doing
   add last number to total
   read the next number
until there is a number to read
output total
```

Note: Please consider that there are several different types of loop statements (for, while, do-while, foreach, etc.), each with its own characteristics (advantages, disadvantages, syntax); so, be ready to learn all loop statements and use them properly!

CODING A JAVA APPLICATION

Coding (successfully) a Java application consists of at least these steps:

- **Write code:** Java source code is stored in a text file with extension .java
- **Compile code:** Compiling the code creates one (or more) files with extension .class (and with the same name of some code elements) and storing processor-independent byte codes.
- **Run (compiled) code:** Running the compiled code means that the JVM interprets at runtime the compiled code and translates it into machine-level instructions for the processor used by your computer.

Note: Later, we will see the differences between running the code by using an IDE, and running the code without using an IDE (e.g., by creating a JAR file).

YOUR FIRST JAVA PROGRAM

This is an example of your first Java program to print a message on the console.

```
// Example 1.5 from "Java Illuminated (5th Ed.)", chapter 1, page 25.
// Note: the class name has to match the file name!
public class Example_01_05__FirstJavaProgram {

   // The Java main method is the entry point of any Java program.
   // Its syntax is always this (you can only change the name of args).
   public static void main( String[] args ) {
      // Method call to Java code to print the input argument on the console.
      System.out.print("Programming is not a spectator sport!");
      // Method call to Java code to exit this program providing a status code
      // (0 means successful termination, non-zero means abnormal termination).
      System.exit(0);
   }

}
```

Note: The name of the code file storing this Java code must match the class name, so it must be named Example_01_05__FirstJavaProgram.java (or the compilation will fail).

PROGRAMMING ERRORS

These are the 3 main types of programming errors:

- **Logic error:** An error in the design of the program, usually leading to an incorrect solution. Most of the logic errors are not found by the compiler, and must be found by testing the program.
- **Compile error:** An error in the syntax or spelling of the code, always leading to a failed compilation. All compile errors are found by the compiler, and must be fixed to achieve a successful compilation.
- **Runtime error:** An error in the design of the program, usually leading to incorrect handling of special cases. Runtime errors are reported by the JVM, and should be fixed to improve the robustness of a program.

Note: Always read the diagnostic information reported whenever a compile/runtime error is detected, because usually it is useful in fixing the error!

BASIC DEBUGGING

Debugging: The activity to fix errors (bugs) that must be performed whenever errors are detected (by the programmer/compiler/JVM).

This is a brief list of the main tasks needed to debug your code:

- **Find error line:** This is the first task needed to try to fix an error. Use error diagnostic information and/or include in code temporary statements to find the code line generating the error.
- **Find error cause:** This is the second task needed to try to fix an error. Figure out "when" and "why" the error is occurring. Include in code temporary statements to find the error cause.
- **Fix error:** This is the last task needed to try to fix an error. Depending on the error, the fix could be trivial or require extensive changes to the program.

BASIC TESTING

Any non-trivial program should be tested properly before declaring it production-ready.

Testing: Verify that a program generates the correct solution to the target problem in all possible scenarios. In particular, ensure that the program works correctly in special cases (e.g., boundary conditions).

Testing usually leads to the discovery of logic errors.

APPENDICES

Appendices

- Negative Binary Integers
- Signed Binary Floats
- ASCII Character Set
- jGRASP Rationale
- Making a JAR File

NEGATIVE BINARY INTEGERS

The standard method to represent negative integers in binary is called Two's Complement.

For a binary integer value, the leftmost digit/bit is reserved for the sign (sign bit).

- If the sign bit is 0, then the integer is positive.
- If the sign bit is 1, then the integer is negative.

Additionally, negative integers are represented using the Two's Complement.
So, to calculate the value of a negative integer, we first calculate its two's complement.

Two's Complement: The two's complement (c) of a binary number (n) is a number that added to (n) yields a sum consisting of all 0 and a carry bit of 1 at the end.
You can calculate (c) by inverting all bits of (n) and then adding 1.

Example: Original negative binary number = (n) = 1111 1111 1101 1010 = -c = -38

Two's complement of (n) = (c) = 0000 0000 0010 0110 = 38

NEGATIVE BINARY INTEGERS (2)

We have seen that converting to decimal the two's complement (c) of a negative binary number (n), and setting the value to be negative, will give us the decimal value of (n).

Example: To understand how the negative integer -5 is stored in memory.
First convert the integer 5 (without the sign) into binary (0101).
Then invert the two's complement by adding 1 and inverting all bits (1001).
So, the binary number 1001 represents in memory the negative integer value -5.

Max Positive Binary Value Using 16 Bits

Considering that the leftmost digit/bit must be 0, the max is: 0111 1111 1111 1111 = 32767

Min Positive Binary Value Using 16 Bits

Considering that the leftmost digit/bit (of our original negative integer) must be 1, then the greatest two's complement we can have is: 1000 0000 0000 0000 = 32768
This two's complement represents the negative integer -32768.

NEGATIVE BINARY INTEGERS (3)

The two’s complement is not the only method to represent negative binary integers!

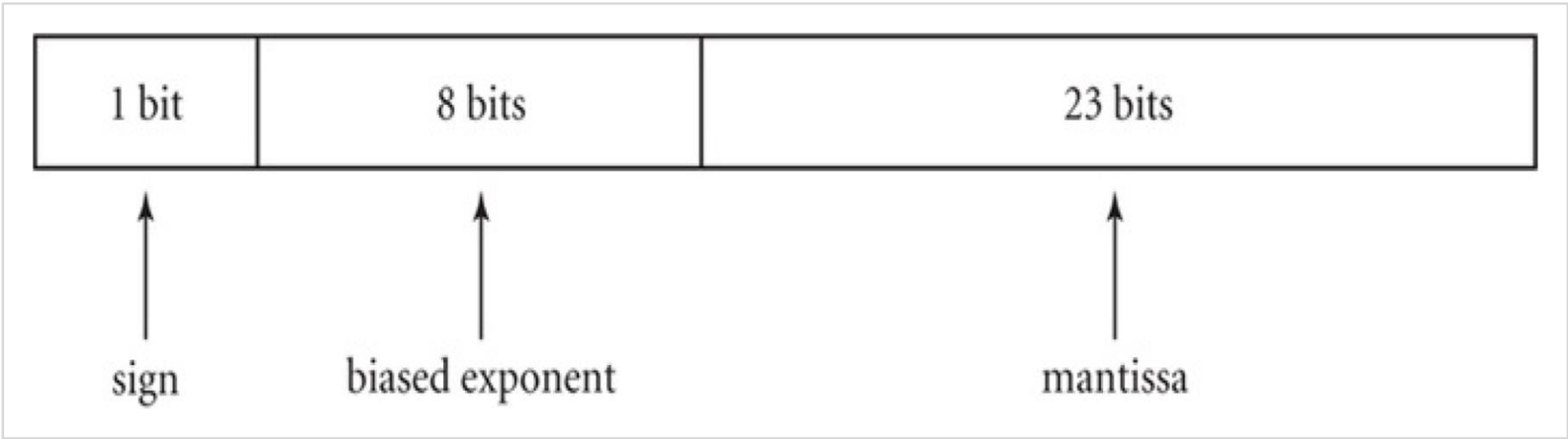
Compared to other methods for representing signed binary integers, the two's complement has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers.
(As long as the inputs are represented in the same number of bits as the output, and any overflow beyond those bits is discarded from the result.)

Also, the two's complement method has no representation for negative zero (-0), and thus does not suffer from its associated difficulties.

SIGNED BINARY FLOATS

The standard representation for signed binary floats is the IEEE 754.

Note: Here we discuss details on IEEE 754 for single-precision (32 bits) signed binary floats. IEEE 754 for double-precision (64 bits) signed binary floats includes minor changes.



Single-precision (32 bits) signed binary float in memory.

Any float value is defined in the standard form: $(-1)^{sign} \times (1 + 0.mantissa) \times 2^{(biased\ exponent\ -\ 127)}$

SIGNED BINARY FLOATS (2)

Signed Decimal Float to Single-Precision Signed Binary Float Conversion:

- Suppose that the float value to convert is: -5.375
- Convert to binary the integer part (without the sign): 5 = 101
- Convert to binary the fractional part: .375 = 0.25 +0.125 = $0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$ = .011
- The float value can now be represented in binary: -5.375 = -101.011
- Now, we can convert the binary representation in the standard form

  $(-1)^{sign} \times (1 + 0.mantissa) \times 2^{(biased\ exponent\ -\ 127)}$

  $-5.375 = -101.011 = -1.01011 \times 2^2$

  $(-1)^1 \times (1 + 0.01011) \times 2^{(129\ -\ 127)}$
- Finally, we get: 1 10000001 01011000000000000000000 (32 bits in total)

  sign = 1 (1 bit)

  biased exponent = 129 = 1000 0001 (8 bits)

  mantissa = 01011000000000000000000 (23 bits)

ASCII CHARACTER SET

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	“	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	‘	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

ASCII character set table.

JGRASP RATIONALE

These are the main reasons why jGRASP has been selected as the official course IDE:

- jGRASP is a simple and lightweight IDE.
- jGRASP is a comprehensive, platform-independent tool (runs on all platforms).
- jGRASP supports multiple programming languages (Java, C++, Python).
- jGRASP is designed for teaching/learning (lets the coder do the coding).
- jGRASP includes automatic software visualization capabilities.
- jGRASP can generate UML class diagrams to understand inter-class dependencies.
- jGRASP provides dynamic views for primitives and objects.
- jGRASP includes a string debugger helping developers to study/review the code.
- jGRASP has its own set of plugins to locate bugs, check style, etc.
- jGRASP has a complete documentation of user guides and tutorials.

MAKING A JAR FILE

JAR File: Java programs are typically distributed as JAR files (Java Archive).
The JAR format (.jar extension) allows the compression of multiple files into one.
The .jar extension is associated by default with the JVM.
So, double-clicking on a .jar file will launch its application directly.

To create a JAR file in jGRASP:

- Create a new folder for the JAR file and all its code files (.java and .class files).
- Create a new project in jGRASP (menu Project).
- Add all the code files needed to the project (.java and .class files).
- Open the project in jGRASP, and create a JAR file (menu Project).
- In JAR creation dialog 1: set type to JAR, and select Sources and Classes as files.
- In JAR creation dialog 2: select class with main function and create JAR.
- A file .jar should have been created in the same folder of the project.

MAKING A JAR FILE (2)

To test a JAR file, you can:

- Open (extract) it in jGRASP to inspect its content.
- Open (extract) it in any compression software (e.g., zip, etc.) to inspect its content.
- Launch it directly by double-clicking it in your operating system.
- Launch it inside jGRASP by double-clicking it from the jGRASP interface.
- Launch it from the command line with: java –jar FirstJARFile.jar