Java Programming

U Hou Lok

Department of Computer Science and Information Engineering,
National Taiwan University

Java 273
22 Aug.–2 Sep., 2016

Class Information

- Lecturer: U Hou Lok
- The class website:

  http://www.csie.ntu.edu.tw/~d99922028/java2016_273T.html,

  is your source for the schedule, homework assignments, announcements, etc.
  - Lecture notes will be uploaded right before class.
  - Note that the lecture notes are organized in English.
  - The lecture notes are mostly contributed by Mr. Zheng-Liang Lu.
- You can contact me, if necessary, by
  - d99922028@csie.ntu.edu.tw, or
  - Facebook.

Time for Class

- Monday–Thursday: 13:00–16:15
- Friday: 13:00–15:00

Class Contents

1. Introduction.
2. Data Types and Operators.
3. Program Flows and Loops(I).
4. Loops(II) and Arrays(I).
5. Arrays(II) and Quiz(I).
6. Methods, Objects and Classes(I).
7. Objects and Classes(II).
8. Objects and Classes(III).
9. Exceptions, I/O Streams, Strings.
10. Quiz(II).

Prerequisites

- This class is designed for the students who are not EE/CS majors.
  - No programming experience required.
  - Would be helpful if you have some programming experiences.
  - May involve with high school math as examples.
- You should consider other classes if you want advanced computer science courses.
- 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

- Feel free to ask me to slow down if I'm speaking too fast.
- Feel free to interrupt me when you have questions.
- Assignments and in-class exercises will ask you to apply the concepts and techniques learned by now.
- Start from the basics; code from the bottom.

Learning Tips

- Start with just one programming language and master it.
- Ask lots of questions.
- Practice makes permanent (and hopefully, perfect).
- 10,000 hours, more or less.
- Grasp the fundamentals for long-term benefits.
- Code by hand.[2]
- Seek out more online resources.

[2] It 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

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

- Programming assignments (30%)
  - Would be 8 to 10 questions
  - Practice makes perfect.
- Quizzes
  - 2 quizzes, one in this Friday (1 hour), and one in next Friday (2 hours).
  - Would be 2 to 4 programming problems.
  - they are not hard.
  - Open everything.

Roll Call

```java
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[3]

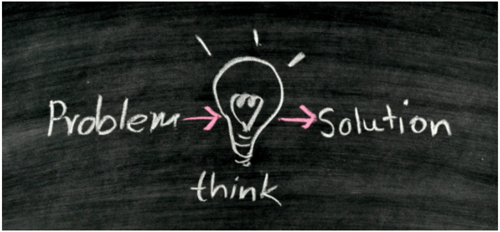
[3] Writing codes is not what the CS people work for. We are writing codes to make a better world.


PROGRAMMER
WHAT MY MOM THINKS I DO
WHAT MY FRIENDS THINK I DO
WHAT SOCIETY THINKS I DO
[JavaScript Application]
Hello, world!
OK
WHAT ARTISTS THINK I DO
WHAT I THINK I DO
WHY WON'T YOU COMPILE?
WHAT I ACTUALLY DO

In Practice

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

- The computational solution is a program.
- The program must be effective.



Programs

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...
- Properties: goal, functionality, algorithms, time and space complexity...

How and Where the Programs Run[6]

- The programs are activated from the disk into the main memory.
- Now we call them the processes.[4]
- 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.[5]
- The outputs are written back to the main memory and further stored into the disk if necessary.

[4] The "process" is a formal terminology used in OS.

[5] Limited number of registers.

[6] You may refer to any class for an introduction to computer system. For example, Introduction to Computer Science & Programming in C.

Memory Hierarchy[7]

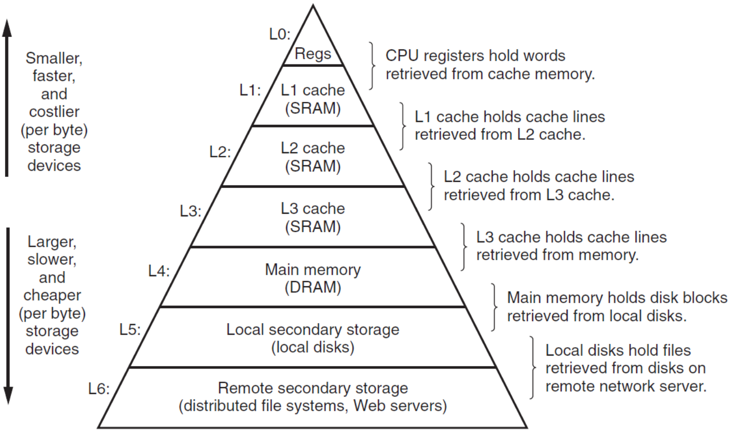


Figure 1.9 **An example of a memory hierarchy.**

[7] See Figure 1-9 in Bryant, p. 14.

Programming Languages

- 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 $\rightarrow$ 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.
  - Java: top 3
- Note that every language originates from reasons.

History[8]

- 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.

[8] See https://en.wikipedia.org/wiki/Programming_language#History.

High-level language program (in C)

```
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)

```
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)

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

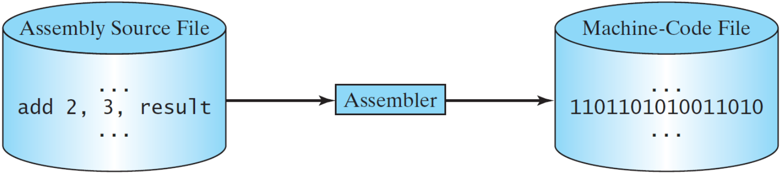
1st-Generation Programming Languages

- Computers understand instructions only in binary, which is a sequence of 0s and 1s. (Why?)
- Each computer has its own set of instructions.[9]
- 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.

[9] For example, X86 and ARM.

2nd-Generation Programming Languages

- An assembly language uses mnemonics[10] 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 find use in extremely intensive processing such as games, video editing, graphic manipulation/rendering.

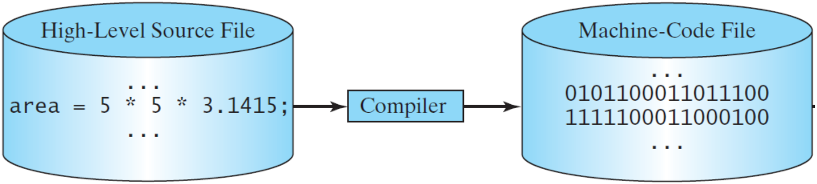
- Note that machine languages and assembly languages are also known as low-level languages.

[10] Easy to recognize and memorize.

3rd-Generation Programming Languages

- The high-level programming languages use English-like words, mathematical notation, and punctuation to write programs.
- They are closer to the human languages.
- Pros:
  - Portable, machine-independent
  - Human-friendly
- For example, C[11], C++[12], and Java[13].

[11] Dennis Ritchie (1973)
[12] Bjarne Stroustrup (1983)
[13] James Gosling (1995)

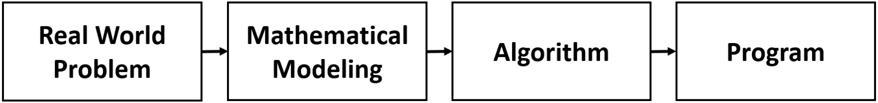


- Note that the machines understand and execute only the machine codes as before.
- The translation is accomplished by a compiler, an interpreter, or a combination of both.[14]

[14] 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.



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[16]

- An algorithm must possess the following properties[15]:
  - 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.

[15] Alan Turing (1912–1954)
[16] 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 $\leftarrow$ be the assignment operator in the following pseudo code.

```
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 $\leftarrow$ 0?

Generalize it a little bit

- Design an algorithm that sorts all the elements in a input list, A, in an ascending order.

Input: A (a list of n numbers)
Output: sorted numbers

- Can you do that? Try to think about it.

"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 (1912–1954)

- Provided a formalization of the concepts of algorithm and universal computation model which can be considered a model of a general purpose computer.
- Turing machine: a central object of study in theory of computation
  - Toy example from Google
  - Also proved that the halting problem for Turing machines is undecidable.[17]
- Turing Test: an attempt to define a standard for the so-called "intelligent" machines
- The father of computing theory and artificial intelligence (AI)
- Turing Award of ACM[18]
- The Imitation Game (2014)

[17] That is, there exist problems which are not solvable.

[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.[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?)

[19] However, now owned by Oracle Corporation, since January 2010.

Java Virutal Machine (JVM)[22]

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

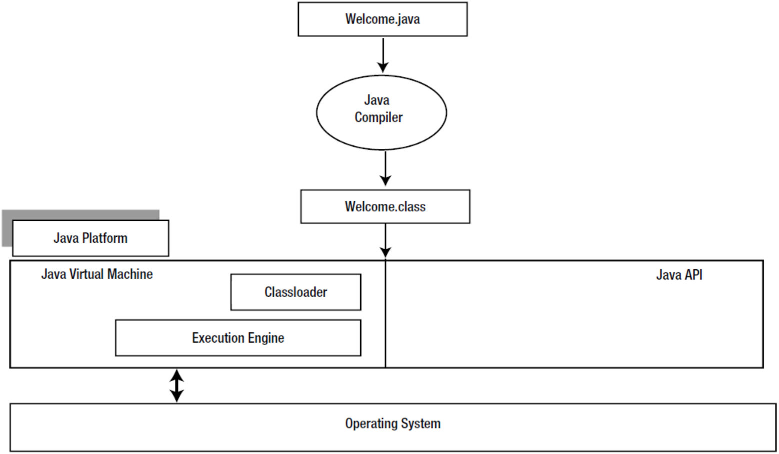
- JVM is a program, not a physical machine.
- The job of JVM is to translate the bytecodes into machine codes according to the platform it is running on.[20]
- To enhance the security, the JVM verifies all bytecodes before the program is executed.[21]
- "No user program can crash the host machine."

[20] Herein we mean operating systems, say Linux.

[21] However, there are a number of possible sources of security vulnerabilities in Java applications. See here.

[22] See JVM.

Compiling and Running a Java Program[23]



[23] See Figure 2-19 in Sharan, p. 59.

Integrated Development Environment (IDE)

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.
- **System.out.println**() is a method within *System.out*, which is automatically imported by default.

A useful tutorial of Eclipse can be found here.

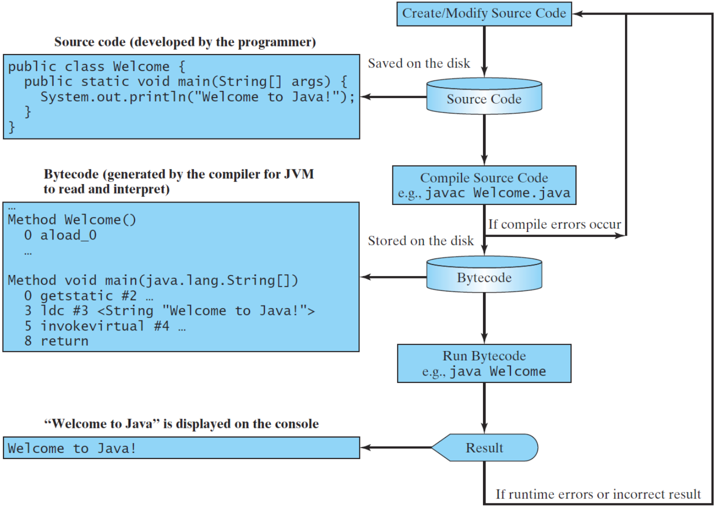
Public Classes

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 jave file.

[24]We will visit the access control later when it comes to encapsulation.

How To Run A Java Program[25]



[25] See Figure 1.14 in YDL, p.20.

Table of Special Characters[26]

Character	*Name*	*Description*
`{}`	Opening and closing braces	Denote a block to enclose statements.
`()`	Opening and closing parentheses	Used with methods.
`[]`	Opening and closing brackets	Denote an array.
`//`	Double slashes	Precede a comment line.
`" "`	Opening and closing quotation marks	Enclose a string (i.e., sequence of characters).
`;`	Semicolon	Mark the end of a statement.

[26]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."

– Edsger W. Dijkstra (1930–2002)

Programming Styles

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.

Problem Set

Exercise 1.1 (Display five messages)

Write a program that displays Welcome to Java five times.

Exercise 1.2 (Display a pattern)

Write a program that displays the following pattern:

```
    J     A     V     V    A
    J    A A     V   V    A A
J   J   AAAAA     V V    AAAAA
 J J   A     A     V    A     A
```

Exercise 1.3 (Print a table)

Write a program that displays the following table:

a	a^2	a^3
1	1	1
2	4	8
3	9	27
4	16	64

```java
class Lecture2 {

"Data types, Variables, and Operators"

}

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

Example

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

Recall that a program comprises data and algorithms.

- How to store the data?
  $\rightarrow$ variable, date type
- How to compute the area?
  $\rightarrow$ arithmetic operators
- How to show the result?
  $\rightarrow$ **System.out.println**()

```java
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 as Box

Variable Declaration

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

```java
    ...
    int x; // x is a variable declared an interger type.
    ...
```

- 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 date 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.

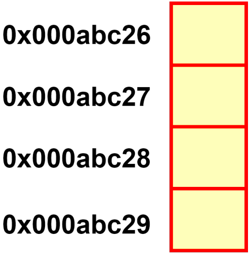
[30] The letter "A" and "a" are different.

Reserved Words[31]

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

[31] See Appendix A in YDL, p. 1253.

Variable as Alias of Memory Address



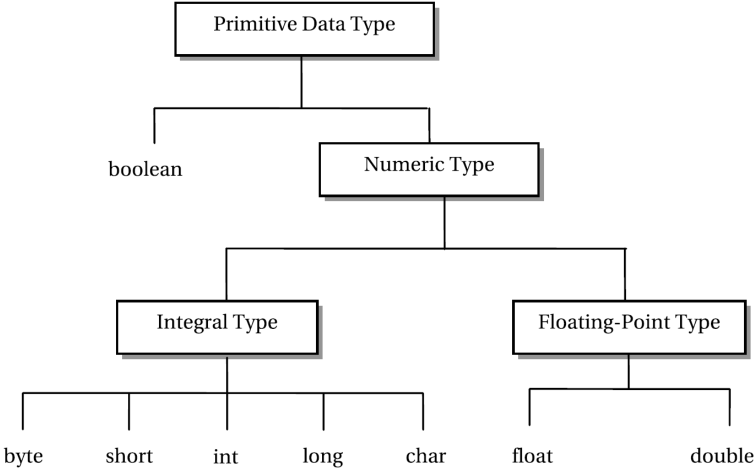
- The number 0x000abc26 stands for one memory address in hexadecimal (0 – 9, and a – f).[32]
- The variable x itself refers to 0x000abc26 in the program after compilation.

[32] See `https://en.wikipedia.org/wiki/Hexadecimal`.

Data Types

- Java is a strongly typed programming language.
  - Every variable has a type.
  - Every (mathematical) expression has a type.
- Note that you cannot change the type of the variable after declaration.
  - So "strongly-typed."
- There are two categories of date types: primitive data types, and reference data types.

Primitive Data Types[33]



[33] See Figure 3-4 in Sharan, p. 67.

Integers

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,767
byte	8	–128 to 127

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

Floating Points

Name	Width in Bits	Approximate Range
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038

- Floating points are used when evaluating expressions that require fractional precision.
  - All transcendental math functions, such as sin(), cos(), and sqrt(), return double values.
- The performance for the double values is actually faster than that for float values on modern processors that have been optimized for high-speed mathematical calculations.
- Be aware that floating-point arithmetic can only approximate real arithmetic. (Why?)

Example: 0.5 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 = 0?

```java
public class FloatingPointsDemo {
public static void main(String args[]) {
System.out.println(0.5 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
}
}
```

- The result is surprising. (Why?)
- You may try this decimal-binary converter.
- This issue is not only associated decimal numbers, but also big integers.
- So double values are not reliable if the program runs for a high-precision calculation.

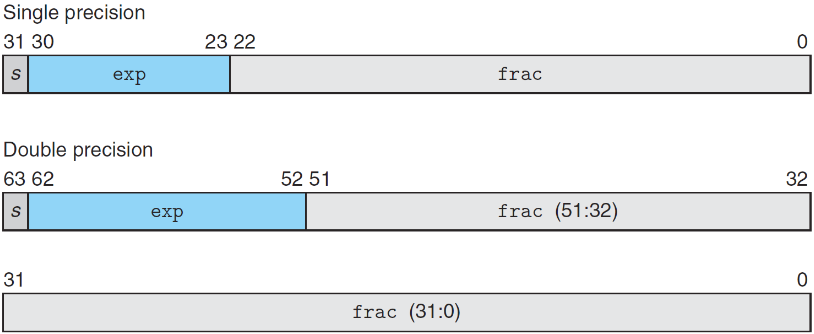
IEEE Floating-Point Representation[34]

$$x = (-1)^s \times M \times 2^E$$

- The sign s determines whether the number is negative ($s = 1$) or positive ($s = 0$).
- The significand M is a coefficient.
  - e.g., $s = 1, M = 12345, E = -3$, then the number is -12.345.
- The exponent E weights the value by a (possibly negative) power of 2.

[34] William Kahan (1985); Aka IEEE754.

Illustration[35]



- That is why we call a double value.

[35] See Figure 2-31 in Byrant, p. 104.

Assignments

- An assignment statement designates a value to the variable.

```java
int x; // make a variable declaration
...
x = 1; // assign 1 to x
```

- The equal sign ($=$) is used as the assignment operator.
  - For example, is the expression $x = x + 1$ correct?
  - Direction: from the right-hand side to the left-hand side
- To assign a value to a variable, you must place the variable name to the left of the assignment operator.[36]
  - For example, $1 = x$ is wrong.
  - 1 cannot be resolved to a memory space.

[36] x can be a l-value and r-value, but 1 and other numbers can be only r-value but not l-value. See Value.

When to Declare Variables: Two "Before" Rules

- Every variable has a scope.
  - The scope of a variable is the range of the program where the variable can be referenced.[37]
- A variable must be declared before it can be assigned a value.
  - In practice, do not declare the variable until you need it.
- A declared variable must be assigned a value before it can be used.[38]

---

[37] The detail of variable scope is introduced later.

[38] In symbolic programming, such as Mathematica and Maple, a variable can be manipulated without assigning a value. For example, you can calculate $x + x$ by the aforementioned languages which return $2x$ as the result.

Arithmetics Operators[39]

Name	*Meaning*	*Example*	*Result*
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

- Note that the operator depends on the operands.
- So the result to division of two integers is still an integer because the fractional part is truncated.

[39]See Table 2-3 in YDL, p. 46.

Tricky Pitfalls

- Can you explain this result?

```java
...
double x = 1 / 2;
System.out.println(x); // output?
...
```

- Revisit $0.5 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 = 0.$[40]

```java
...
System.out.println(1 / 2 - 1 / 10 - 1 / 10 - 1 / 10 - 1 / 10 - 1 /
    10); // output 0; however, this is not the real solution to the
     original problem.
...
```

[40]Contribution by Mr. xxx (APcomSci267) on June 7, 2016.

Type Conversion and Compatibility

- If a type is compatible to another, then the compiler will perform the conversion implicitly.
  - For example, the integer 1 is compatible to a double value 1.0.
- However, there is no automatic conversion from double to int. (Why?)
- To do so, you must use a **cast**, which performs an explicit conversion for compilation.
- Similarly, a long value is not compatible to int.

Casting

```java
    ...
    int x = 1;
    double y = x; // compatible; implicit conversion
    x = y; // incompatible; need an explicit conversion by casting
    x = (int) y; // succeed!!
    ...
```

- Note that the Java compiler does only type-checking but no real execution before compilation.
- In other words, the values of x and y are unknown until they are really executed.

Type Conversion and Compatibility (concluded)

- small-size types $\rightarrow$ large-size types
- small-size types $\not\leftarrow$ large-size types (need a cast)
- simple types $\rightarrow$ complicated types
- simple types $\not\leftarrow$ complicated types (need a cast)

Characters

- A character stored in the machine is represented by a sequence of 0s and 1s.
  - For example, ASCII code. (See the next page.)
- The char type is a 16-bit unsigned primitive data type.
- Java uses Unicode to represent characters.
  - Unicode defines a fully international character set that can represent all of the characters found in all human languages.

ASCII (7-bit version)

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Example

- Characters can also be used as an integer type on which you can perform arithmetic operations.
- For example,

```
...
// A single-quoted value is the char type.
char x = 'a';
System.out.println(x + 1); // output 98!!
System.out.println((char)(x + 1)); // output b
...
```

- Note that a double-quoted string is a **String** object, totally different from char values.

Boolean Values

- The program is supposed to do decision making by itself, for example, Google Driverless Car.
- To do this, Java has the boolean type flow controls (selections and iterations).
  - Only two possible values, true and false.
- Note that a boolean value cannot be cast into a value of another type, nor can a value of another type be cast into a boolean value.

Rational Operators[41]

Java Operator	*Mathematics Symbol*	*Name*
`<`	$<$	less than
`<=`	$\leq$	less than or equal to
`>`	$>$	greater than
`>=`	$\geq$	greater than or equal to
`==`	$=$	equal to
`!=`	$\neq$	not equal to

- These operators take two operands.
- Rational expressions return boolean values.
- Note that the equality comparison operator is double equality sign (==), not single equality sign (=).

[41] See Table 3-1 in YDL, p. 82.

Example

```java
    ...
    int x = 2;
    boolean a = x > 1;
    boolean b = x < 1;
    boolean c = x == 1;
    boolean d = x != 1;
    boolean e = 1 < x < 3; // sorry?
    ...
```

- Be aware that e is logically correct but syntactically wrong.
- Usually, the boolean expression consists of a combination of rational expressions.
  - For example, $1 < x < 3$ should be $(1 < x)\&\&(x < 3)$, where && refers to the AND operator.

Logical Operators[42]

Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

[42] See Table 3-2 in YDL, p. 102.

Truth Table

- Let X and Y be two Boolean variables.
- Then the truth table for each logical operators is as follows:

X	Y	!X	X&&Y	X ‖ Y	X $\wedge Y$
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

- Note that the instructions of computers, such as arithmetic operations, are implemented by logic gates.[43]

[43] See any textbook for digital circuit design.