

Building Java Programs

Chapter 1

Introduction to Java Programming

Copyright (c) Pearson 2013.
All rights reserved.

What is computer science?

- Computer Science
 - The study of theoretical foundations of information and computation and their implementation and application in computer systems. -- Wikipedia
 - Many subfields
 - Graphics, Computer Vision
 - Artificial Intelligence
 - Scientific Computing
 - Robotics
 - Databases, Data Mining
 - Computational Linguistics, Natural Language Processing ...
- Computer Engineering
 - Overlap with CS and EE; emphasizes hardware

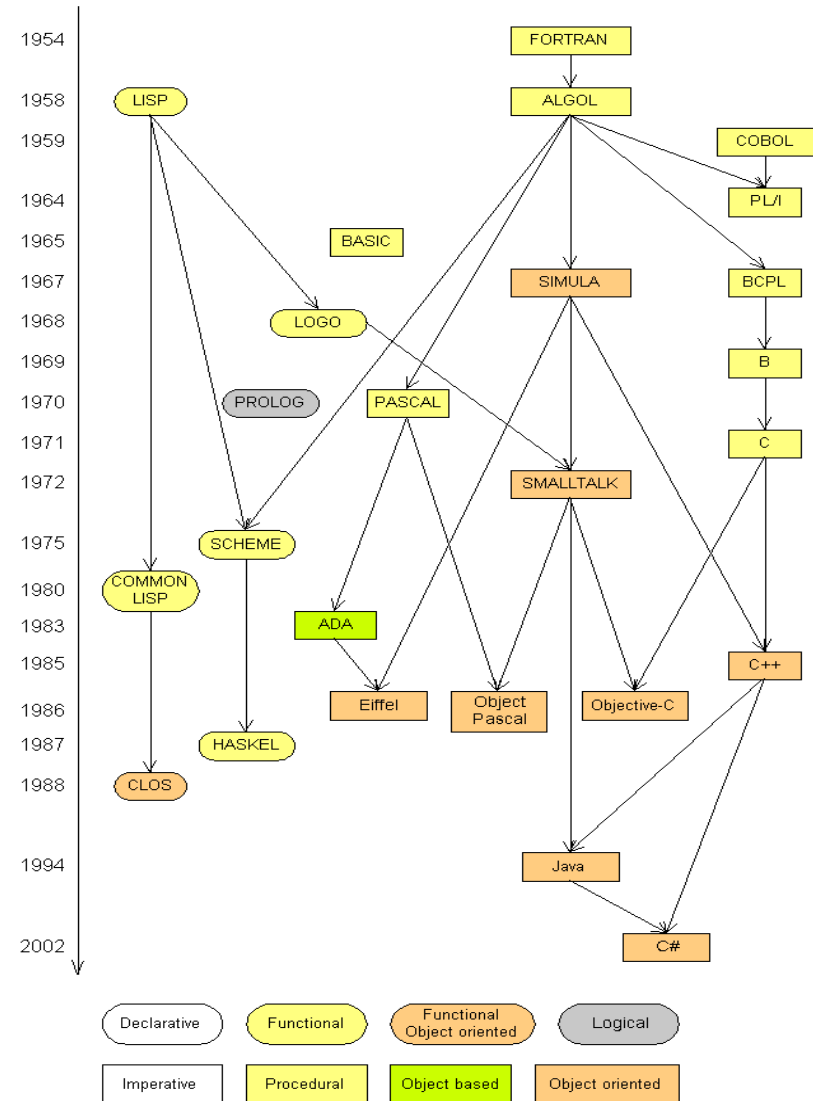
What is programming?

- **program:** A set of instructions to be carried out by a computer.
- **program execution:** The act of carrying out the instructions contained in a program.
- **programming language:** A systematic set of rules used to describe computations in a format that is editable by humans.
 - This textbook teaches programming in a language named Java.



Programming languages

- Some influential ones:
 - FORTRAN
 - science / engineering
 - COBOL
 - business data
 - LISP
 - logic and AI
 - BASIC
 - a simple language



Some modern languages

- procedural languages: programs are a series of commands
 - **Pascal** (1970): designed for education
 - **C** (1972): low-level operating systems and device drivers
- functional programming: functions map inputs to outputs
 - **Haskell** (1990), **JavaScript** (1995)
- object-oriented languages: programs use interacting "objects"
 - **C++** (1985): "object-oriented" improvements to C
 - successful in industry; widely used for high-performance & embedded SW
 - **Java** (1995): designed for embedded systems, web apps/servers
 - Runs on many platforms (Windows, Mac, Linux, cell phones...)
 - Widely taught in computer science programs
 - **Python** (1991) Widely used in data engineering & data science
 - **Kotlin** (2011) Fully interoperable with Java. Adopts modern concepts for simplicity, safety, & performance
 - **Swift** (2014) Developed by Apple. Adopts modern concepts for simplicity, safety & performance

Basic Java programs with `println` statements

Compile/run a program

1. Write it.

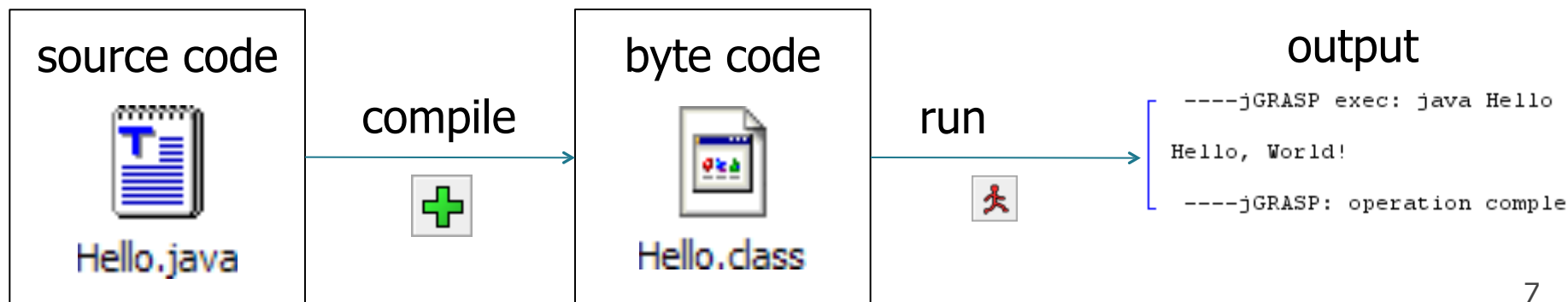
- **code** or **source code**: The set of instructions in a program.

2. Compile it.

- **compile**: Translate a program from one language to another.
- **byte code**: The Java compiler converts your code into a format named byte code that runs on many computer types.

3. Run (execute) it.

- **output**: The messages printed to the user by a program.



A Java program

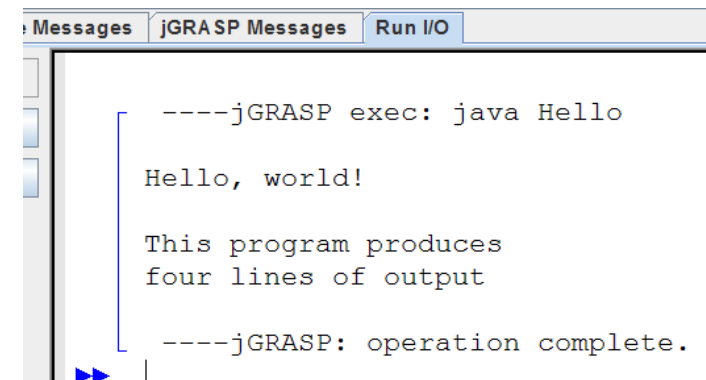
```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

- **Its output:**

Hello, world!

This program produces
four lines of output

- **console:** Text box into which the program's output is printed.



Structure of a Java program

```
public class name {  
    public static void main(String[] args) {  
        statement;  
        statement;  
        ...  
        statement;  
    }  
}
```

The diagram illustrates the structure of a Java program with three callout boxes:

- class:** a program (points to **name**)
- method:** a named group of statements (points to `main`)
- statement:** a command to be executed (points to **statement**)

- Every executable Java program consists of a **class**,
 - that contains a **method** named `main`,
 - that contains the **statements** (commands) to be executed.

System.out.println

- A statement that prints a line of output on the console.
 - pronounced "print-linn"
 - sometimes called a "println statement" for short
- Two ways to use `System.out.println` :
 - `System.out.println("text");`
Prints the given message as output.
 - `System.out.println();`
Prints a blank line of output.

Names and identifiers

- You must give your program a name.

```
public class GangstaRap {
```

- Naming convention: capitalize each word (e.g. `MyClassName`)
- Your program's file must match exactly (`GangstaRap.java`)
 - includes capitalization (Java is "case-sensitive")

- **identifier**: A name given to an item in your program.

- must start with a letter or `_` or `$`
- subsequent characters can be any of those or a number

• **legal:** `_myName` `TheCure` `ANSWER_IS_42` `$bling$`

• **illegal:** `me+u` `49ers` `side-swipe` `Ph.D's`

Keywords

- **keyword:** An identifier that you cannot use because it already has a reserved meaning in Java.

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

Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular language.
 - Every basic Java statement ends with a semicolon ;
 - The contents of a class or method occur between { and }
- **syntax error (compiler error)**: A problem in the structure of a program that causes the compiler to fail.
 - Missing semicolon
 - Too many or too few { } braces
 - Illegal identifier for class name
 - Class and file names do not match

...

Syntax error example

```
1 public class Hello {
2     pooblic static void main(String[] args) {
3         System.owt.println("Hello, world!")__
4     }
5 }
```

- Compiler output:

```
Hello.java:2: <identifier> expected
      pooblic static void main(String[] args) {
        ^
```

```
Hello.java:3: ';' expected
    }
    ^
```

```
2 errors
```

- The compiler shows the line number where it found the error.
- The error messages can be tough to understand!

Strings

- **string**: A sequence of characters to be printed.
 - Starts and ends with a " quote " character.
 - The quotes do not appear in the output.
 - Examples:

```
"hello"
```

```
"This is a string.  It's very long!"
```

- **Restrictions:**
 - May not span multiple lines.

```
"This is not  
a legal String."
```

- May not contain a " character.

```
"This is not a "legal" String either."
```

Escape sequences

- **escape sequence:** A special sequence of characters used to represent certain special characters in a string.

<code>\t</code>	tab character
<code>\n</code>	new line character
<code>\"</code>	quotation mark character
<code>\\</code>	backslash character

- **Example:**

```
System.out.println("\\hello\nhow\tare \"you\"?\\\\\");
```

- **Output:**

```
\hello
how      are "you"?\\
```


Comments

- **comment:** A note written in source code by the programmer to describe or clarify the code.
 - Comments are not executed when your program runs.
- Syntax:
 - // comment text, on one line**
 - or,
 - /* comment text; may span multiple lines */**
- Examples:
 - // This is a one-line comment.**
 - /* This is a very long
multi-line comment. */**

Using comments

- Where to place comments:
 - at the top of each file (a "comment header")
 - at the start of every method (seen later)
 - to explain complex pieces of code
- Comments are useful for:
 - Understanding larger, more complex programs.
 - Multiple programmers working together, who must understand each other's code.

Comments example

```
/* Suzy Student, CS 101, Fall 2019
   This program prints lyrics about ... something. */

public class BaWitDaBa {
    public static void main(String[] args) {
        // first verse
        System.out.println("Bawitdaba");
        System.out.println("da bang a dang diggy diggy");
        System.out.println();

        // second verse
        System.out.println("diggy said the boogy");
        System.out.println("said up jump the boogy");
    }
}
```