Building Java Programs

Chapter 1: Introduction to Java Programming

Lecture outline

- syllabus and course policies

- basic Java programs
  - programs and programming languages
  - output with `println` statements
  - syntax and errors
  - String literals and escape sequences

About me

- Marty Stepp
  - email: stepp@cs.washington.edu
  - office: CSE 466
  - phone: (206) 685-2181
- Lecturer of Computer Science

- past
  - University of Arizona 1999-2003
  - Microsoft 2003-2004
  - University of Washington Tacoma 2004-2006

Basic Java programs with `println` statements

reading: 1.1 - 1.3

Computer Science

- What is **computer science**?
  - The study of theoretical foundations of information and computation and their implementation and application in computer systems. -- *Wikipedia*

  - *Math*: number theory, graphs, computational geometry, ...
  - *Theory of computation*
  - *Data structures, algorithms, databases*
  - *Programming*: Languages, compilers, ...
  - *Software engineering*
  - *Communication and networking*
  - *Artificial intelligence*
  - *Graphics and multimedia*
  - *Scientific computing*
  - *...*

Computer programs

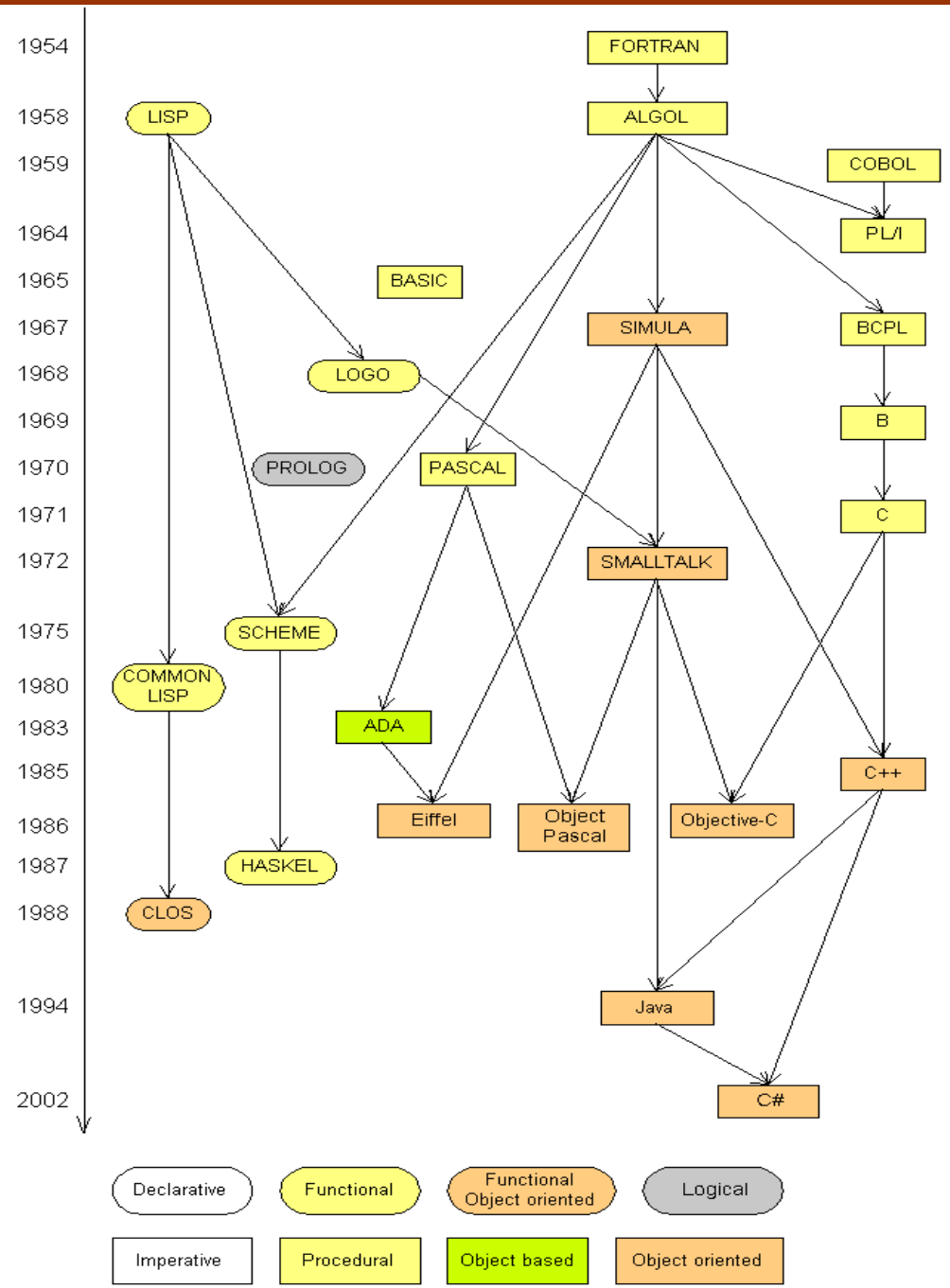
- **program**: A set of instructions to be carried out by a computer.

- **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.

Languages

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



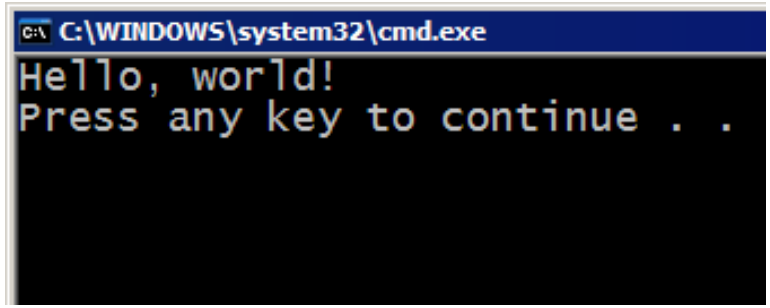
Some modern languages

- *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
  - **Lisp** (1958) / **Scheme** (1975), **ML** (1973), **Haskell** (1990)
- *object-oriented languages*: programs use interacting "objects"
  - **Smalltalk** (1980): first major object-oriented language
  - **C++** (1985): "object-oriented" improvements to C
    - successful in industry; used to build major OSes such as Windows
  - **Java** (1995): designed for embedded systems, web apps/servers
    - Runs on many platforms (Windows, Mac, Linux, cell phones...)
    - The language taught in this textbook

A basic Java program

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

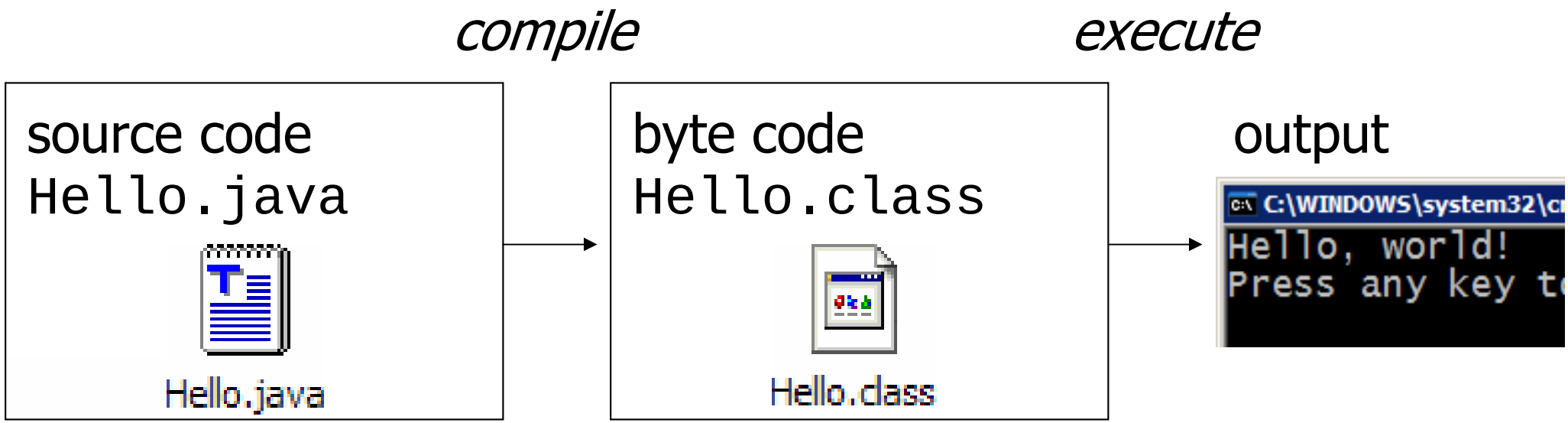
- **code** or **source code**: The sequence of instructions in a program.
  - The code in this program instructs the computer to display a message of **Hello, world!** on the screen.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
  - Some editors pop up the console as an external window, and others contain their own console window.



Compiling/running a program

Before you run your programs, you must *compile* them.

- **compiler**: Translates a computer program written in one language into another language.
  - Java Development Kit includes a Java compiler.
  - **byte code**: The Java compiler converts your source code into a format named *byte code* that can be executed on many different kinds of computers.



Another Java program

```
public class Hello2 {
    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");
    }
}
```

- The code in this program instructs the computer to print four messages on the screen.
  - Its output:

```
Hello, world!

This program produces
four lines of output
```

Structure of Java programs

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

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

Java terminology

- **class**: A module that can contain executable code.
  - Every program you write will be a class.

- **statement**: An executable command to the computer.

- **method**: A named sequence of statements that can be executed together to perform a particular action.

  - A special method named `main` signifies the code that should be executed when your program runs.

  - Your program can have other methods in addition to `main`. (seen later)

Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular programming language.

- some Java syntax:
  - every basic Java statement ends with a semicolon ;
  - The contents of a class or method occur between { and }

Syntax errors

- **syntax error** or **compiler error**: A problem in the structure of a program that causes the compiler to fail.
  - If you type your Java program incorrectly, you may violate Java's syntax and cause a syntax error.

```
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:5: ';' expected
}
^
2 errors
```

Fixing syntax errors

- Error messages do not always help us understand what is wrong:

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

  - We'd have preferred a friendly message such as, "*You misspelled public*"

- The compiler does tell us the line number on which it found the error...
  - But it is not always the true source of the problem.

```
1  public class MissingSemicolon {
2      public static void main(String[] args) {
3          System.out.println("A rose by any other name")
4          System.out.println("would smell as sweet");
5      }
6  }
```

```
MissingSemicolon.java:4: ';' expected
System.out.println("would smell as sweet");
^
```

System.out.println

- `System.out.println` : A statement to instruct the computer to print 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 a line of text on the console.

  `System.out.println();`
  - Prints a blank line on the console.

Strings and string literals

- **string**: A sequence of text characters that can be printed or manipulated in a program.
  - sometimes also called a *string literal*
  - strings in Java start and end with quotation mark " characters

  - Examples:

```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

String restrictions

- A string may not span across multiple lines.

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

- A string may not contain a " character. (' is okay)

```
"This is not a "legal" String either."
"This is 'okay' though."
```

Escape sequences

- A string can represent certain special characters by preceding them with a backslash \ (this is called an **escape sequence**).
  - `\t` tab character
  - `\n` new line character
  - `\"` quotation mark character
  - `\\` backslash character

  - Example:
    ```
    System.out.println("\\hello\nhow\tare \"you\"?\\\\");
    ```
  - Output:
    ```
    \hello
    how     are "you"?\\
    ```

Questions

- What is the output of each of the following `println` statements?

```
System.out.println("\ta\tb\tc");
System.out.println("\\\\");
System.out.println("'");
System.out.println("\"\"\"");
System.out.println("C:\nin\the downward spiral");
```

- Write a `println` statement to produce the following line of output:

```
/ \ // \\ /// \\\
```

Answers

- Output of each `println` statement:

```
        a        b        c
\\
'
"""
C:
in       he downward spiral
```

- `println` statement to produce the line of output:

```
System.out.println("/ \\ // \\\\ /// \\\\\\");
```

Questions

- What `println` statements will generate the following output?

```
This program prints a
quote from the Gettysburg Address.

"Four score and seven years ago,
our 'fore fathers' brought forth on
this continent a new nation."
```

- What `println` statements will generate the following output?

```
A "quoted" String is
'much' better if you learn
the rules of "escape sequences."

Also, "" represents an empty String.
Don't forget: use \" instead of " !
'' is not the same as "
```

Answers

- `println` statements to generate the output:

```
System.out.println("This program prints a");
System.out.println("quote from the Gettysburg Address.");
System.out.println();
System.out.println("\"Four score and seven years ago,");
System.out.println("our 'fore fathers' brought forth on");
System.out.println("this continent a new nation.\"");
```

- `println` statements to generate the output:

```
System.out.println("A \"quoted\" String is");
System.out.println("'much' better if you learn");
System.out.println("the rules of \"escape sequences.\"");
System.out.println();
System.out.println("Also, \"\" represents an empty String.");
System.out.println("Don't forget: use \\\" instead of \" !");
System.out.println("'' is not the same as \"");
```