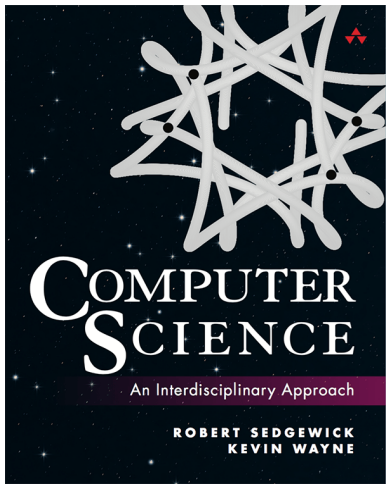

1.1 Your First Java Program: Hello World

In this section, our plan is to lead you into the world of Java programming by taking you through the three basic steps required to get a simple program running. As with any application, you need to be sure that Java is properly installed on your computer. You also need an editor and a terminal application. Here are system specific instructions for three popular home operating systems. [Mac OS X · Windows · Linux]

Programming in Java.

We break the process of programming in Java into three steps:

1. *Create* the program by typing it into a text editor and saving it to a file named, say, `MyProgram.java`.
2. *Compile* it by typing "`javac MyProgram.java`" in the terminal window.
3. *Execute* (or *run*) it by typing "`java MyProgram`" in the terminal window.

The first step creates the program; the second translates it into a language more suitable for machine execution (and puts the result in a file named `MyProgram.class`); the third actually runs the program.

- *Creating a Java program*. A program is nothing more than a sequence of characters, like a sentence, a paragraph, or a poem. To create one, we need only define that sequence characters using a text editor in the same way as we do for email. HelloWorld.java is an example program. Type these character into your text editor and save it into a file named `HelloWorld.java`.

```
public class HelloWorld {
   public static void main(String[] args) {
      // Prints "Hello, World" in the terminal window.
      System.out.println("Hello, World");
   }
}
```

- *Compiling a Java program*. A *compiler* is an application that translates programs from the Java language to a language more suitable for executing on the computer. It takes a text file with the `.java` extension as input (your program) and produces a file with a `.class` extension (the computer-language version). To compile `HelloWorld.java` type the boldfaced text below at the terminal. (We use the `%` symbol to denote the command prompt, but it may appear different depending on your system.)

```
% javac HelloWorld.java
```

  If you typed in the program correctly, you should see no error messages. Otherwise, go back and make sure you typed in the program exactly as it appears above.
- *Executing (or running) a Java program*. Once you compile your program, you can execute it. This is the exciting part, where the computer follows your instructions. To run the `HelloWorld` program, type the following in the terminal window:

```
% java HelloWorld
```

  If all goes well, you should see the following response

```
Hello, World
```

- *Understanding a Java program*. The key line with `System.out.println()` prints the text "Hello, World" in the terminal window. When we begin to write more complicated programs, we will discuss the meaning of `public`, `class`, `main`, `String[]`, `args`, `System.out`, and so on.
- *Creating your own Java program*. For the time being, all of our programs will be just like `HelloWorld.java`, except with a different sequence of statements in `main()`. The easiest way to write such a program is to:
  - Copy `HelloWorld.java` into a new file whose name is the program name followed by `.java`.
  - Replace `HelloWorld` with the program name everywhere.
  - Replace the print statement by a sequence of statements.

Errors.

Most errors are easily fixed by carefully examining the program as we create it, in just the same way as we fix spelling and grammatical errors when we type an e-mail message.

- *Compile-time errors*. These errors are caught by the system when we compile the program, because they prevent the compiler from doing the translation (so it issues an error message that tries to explain why).
- *Run-time errors*. These errors are caught by the system when we execute the program, because the program tries to perform an invalid operation (e.g., division by zero).
- *Logical errors*. These errors are (hopefully) caught by the programmer when we execute the program and it produces the wrong answer. Bugs are the bane of a programmer's existence. They can be subtle and very hard to find.

One of the very first skills that you will learn is to identify errors; one of the next will be to be sufficiently careful when coding to avoid many of them.

Input and output.

Typically, we want to provide *input* to our programs: data that they can process to produce a result. The simplest way to provide input data is illustrated in UseArgument.java. Whenever this program is executed, it reads the command-line argument that you type after the program name and prints it back out to the terminal as part of the message.

```
% javac UseArgument.java
% java UseArgument Alice
Hi, Alice. How are you?
% java UseArgument Bob
Hi, Bob. How are you?
```

Exercises

1. Write a program TenHelloWorlds.java that prints "`Hello, World`" ten times.
5. Modify UseArgument.java to make a program UseThree.java that takes three names and prints out a proper sentence with the names in the reverse of the order given, so that for example, "`java UseThree Alice Bob Carol`" gives "`Hi Carol, Bob, and Alice.`".

Web Exercises

1. Write a program Initials.java that prints your initials using nine rows of asterisks like the one below.

```
**        ***    **********      **            *            **
**      ***      **         **     **          ***          **
**    ***        **          **     **        ** **        **
**  ***          **           **     **      **   **      **
*****            **           **      **    **     **    **
**  ***          **           **       **  **       **  **
**    ***        **          **          ****         ****
**      ***      **         **            **           **
**        ***    **********                *            *
```

2. Describe what happens if, in HelloWorld.java, you omit
    1. `main`
    2. `String`
    3. `HelloWorld`

    4. `System.out`
    5. `println`
3. Describe what happens if, in HelloWorld.java, you omit
    1. the `;`
    2. the first `"`
    3. the second `"`
    4. the first `{`
    5. the second `{`
    6. the first `}`
    7. the second `}`
4. Describe what happens if, in HelloWorld.java, you misspell (by, say, omitting the second letter)
    1. `main`
    2. `String`
    3. `HelloWorld`
    4. `System.out`
    5. `println`
5. I typed in the following program. It compiles fine, but when I execute it, I get the error `java.lang.NoSuchMethodError: main`. What am I doing wrong?

```
public class Hello {
   public static void main() {
      System.out.println("Doesn't execute");
   }
}
```

Answer: you forgot the `String[] args`. It is required.

Last modified on August 02, 2016.