

OOP Using JAVA

03 – Your First Java Program

Dr. Mostafa A. El-Hosseini

melhosseini@eng.kfs.edu.eg

Recap

- » JAVA merits
- » Object-oriented
- » JAVA Code
- » Objects – Classes – Methods – main –
- » Running Java

AGENDA

- » Library
- » Java Specification
- » API – Application Programming Interface
- » Keywords
- » Identifiers
- » Comments
- » Your First Java Program

Java uses libraries

- » The Java language is **quite small**. Much of the language's functionality is achieved through **libraries**.
- » Two types of Java libraries:
 - » **Core** libraries that are part of the Java Development Kit (**JDK**), and
 - » **optional library additions**.
- » The core libraries must be present in every implementation of Java, while the optional libraries can be present or not.

The grammar and the common names

- » The people at Sun Microsystems who created Java thought of Java as coming in two parts. Just as **English has its grammar and commonly used names**,
- » the Java programming language has its
 - » **specification** (its grammar) and
 - » **Application Programming Interface API**(its commonly used names).
- » Whenever I write Java programs, I keep two important pieces of documentation — one for each part of the language — on my desk:

The grammar and the common names

- » **The Java Language Specification**: This includes rules like, “Always put an open parenthesis after the word for” and “Use an asterisk to multiply two numbers.”
- » **API** contains thousands of tools that were added to Java after the language’s grammar was defined.
 - » The tools include a routine named **pow** that can raise 5 to the 10th power for you.
 - » **JFrame** displays a window on your computer’s screen.
 - » **Other tools** listen for the user’s button clicks, query databases, and do all kinds of useful things.

The grammar and the common names

- » **The Language Specification**, is relatively small. That doesn't mean you won't take plenty of time finding out how to use the rules in the Language Specification. Other programming languages, however, have double, triple, or ten times the number of rules.
- » The **API** — can be intimidating because it's so large.
- » The **API** contains at least 3,000 tools and keeps growing with each new Java language release.
- » Well, the good news is that you don't have to memorize anything in the API.
- » You can look up the stuff you need to use in the documentation and ignore the stuff you don't need

API

- » The API is just a set of classes and other tools that were created by ordinary programmers who happen to participate in the official JCP — the Java Community Process.
- » Unlike the tools that you create yourself, the tools in the API are distributed with every version of Java

The words in a Java program

1 Keywords:

- » A **keyword** is a word that has its own special meaning in the Java programming language, and that **meaning doesn't change from one program to another**.
- » Examples of keywords in Java include **if**, **else**, and **do**.
- » The **JCP** committee members, who have the final say on what constitutes a Java program, have **chosen all Java's keywords**

The words in a Java program

② Identifiers:

- » An **identifier** is a name for something. The identifier's meaning can change from one program to another, but some identifiers' meanings tend to change more than others.
 - » **Identifiers created by you and me**: you create new names for classes and other things that you describe in your programs.
 - » Of course, you may name something **Prime**, and another guy uses **Prime** to refer to another thing
 - » That's okay because Java doesn't have a predetermined meaning for the word **Prime**

Identifier

- » **Identifiers from the API**: The JCP members have created names for many things and thrown at least 3,000 of these names into the Java API.
- » The **API** comes with each version of Java, so these names are available to anyone who writes a Java program. Examples of such names are **String**, **Integer**, **JWindow**, **JButton**, **JTextField**, and **File**

HINTS

- » Every complete Java statement ends with a **semicolon**.
- » In a Java program, **curly braces** enclose meaningful units of code.
- » You, the programmer, can (and should) indent lines so that other programmers can see the outline form of your code at a glance
- » **If you don't indent** lines of code in an informative manner, **your program will still work correctly**, but neither you nor any other programmer will be able to figure out what you were thinking when you wrote the code.

HINTS

Curly Braces for a Java Class

```
class Displayer {  
  
    public static void main(String args[]) {  
        System.out.println("You'll love Java!");  
    }  
}
```

Curly Braces for a Java Method

```
class Displayer {  
  
    public static void main(String args[]) {  
        System.out.println("You'll love Java!");  
    }  
}
```

Adding Comments to your code

» A comment is a special section of text inside a program. It is text whose purpose is to help people understand the program. A comment is part of a good program's documentation.

① **Traditional comments:** The comment begins with `/*` and ends with `*/`. Everything between the opening `/*` and the closing `*/` is for human eyes only

Comments

- ② **End-of-line comments**: starts with two slashes `//`, and goes to the end of a line of type. Once again, no text inside the end-of-line comment gets translated by the compiler.
- ③ **Javadoc comments**: A javadoc comment begins with a slash and two asterisks (`/**`).
 - A **javadoc** comment is a special kind of traditional comment.
 - A **javadoc** comment is meant to be read by people who never even look at the Java code
 - a certain program called **javadoc** can find all the **javadoc** comments in the Listing and turn these comments into a **nice looking Web page**

Comments

Three Kinds of Comments

```
/*
 * Listing 3-6 in "Java For Dummies"
 *
 * Copyright 2006 Wiley Publishing, Inc. All rights reserved.
 */

/**
 * The Displayer class displays text on the computer screen.
 *
 * @author Barry Burd
 * @version 1.6 02/21/06
 * @see java.lang.System
 */
class Displayer {

    /**
     * Execution of the program starts at this main method.
     *
     * @param args (See Chapter 11.)
     */
    public static void main(String args[]) {
        System.out.println("I love Java!"); //Changed to "I"
    }
}
```

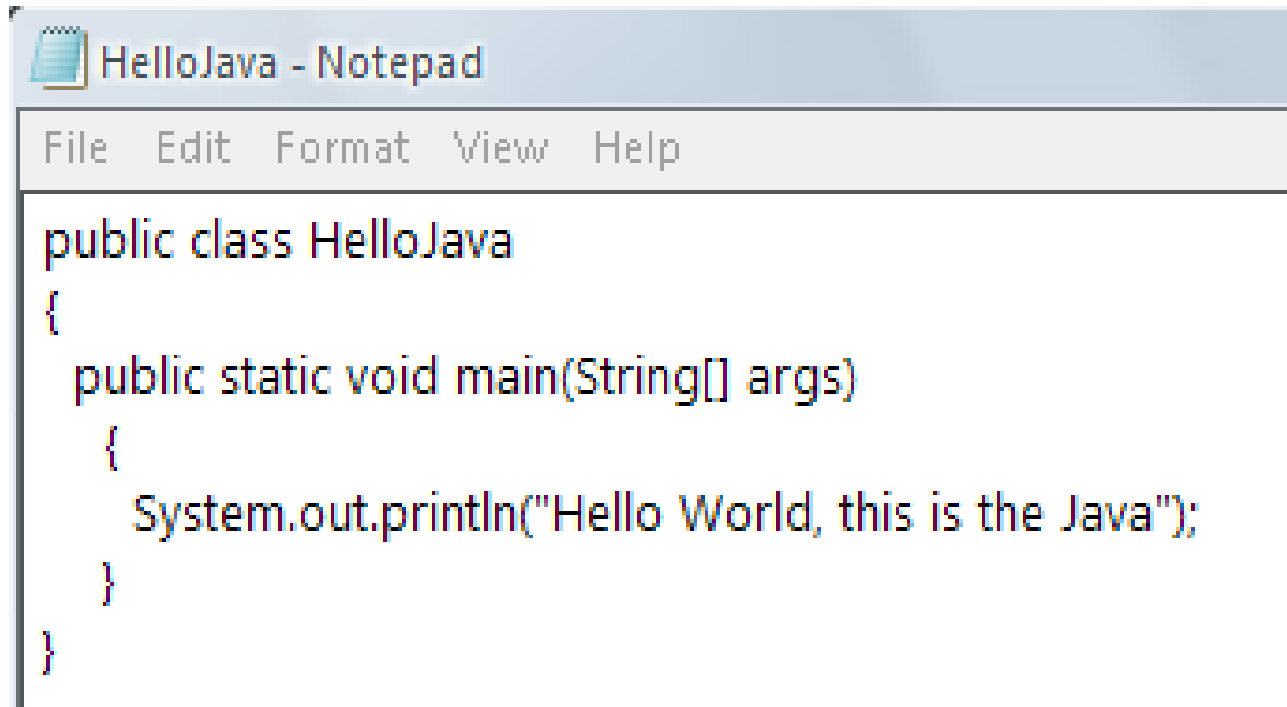
Your First Java Program

- ① Installing the JDK: <http://java.sun.com>
- ② Selecting an Editor: One component that is not included in the JDK is a text editor.
 - o You need a text editor for creating the Java source used by your programs. There are several text editors available in a PC. Edit is the MS-DOS editor while Notepad and WordPad work in Windows
 - o Notepad and WordPad are Microsoft editors that run in Windows. WordPad is the more powerful one.
 - o But they do not save a file with the extension **.java**.
 - o **Enclose the filename in double quotation** marks when saving the source file to disk

Your First Java Program

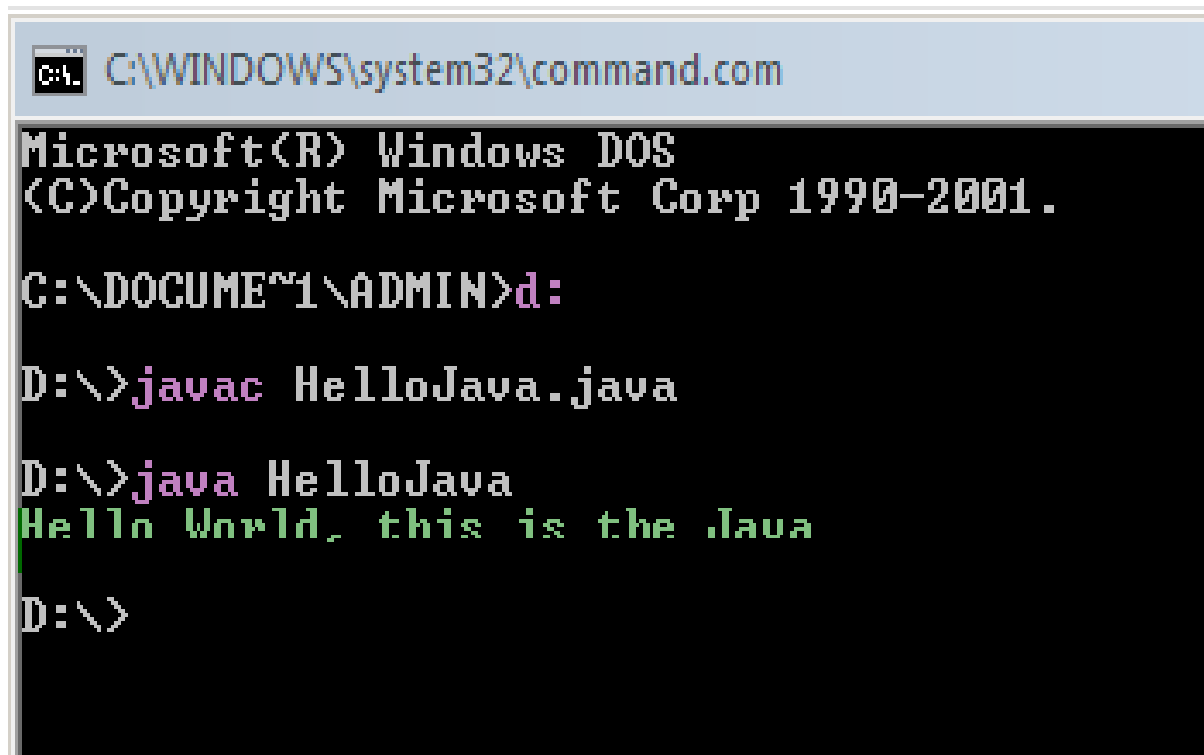
- ③ Write your program `HelloJava.java`
- ④ The Java program development software that comes with the JDK must be run from the MS DOS command line
 - If you have correctly entered the `PATH` and `CLASSPATH` commands in your `autoexe.bat` file, the system will be able to locate the development software
- ⑤ Open a window with the MS DOS command prompt.
- ⑥ Log on to the directory where the source file that you intend to compile is located.
- ⑦ Execute the Java compiler from the MS DOS command prompt.

Your First Java Program



```
public class HelloJava
{
    public static void main(String[] args)
    {
        System.out.println("Hello World, this is the Java");
    }
}
```

Your First Java Program



```
C:\WINDOWS\system32\command.com
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-2001.
C:\DOCUME~1\ADMIN>d:
D:\>javac HelloJava.java
D:\>java HelloJava
Hello World, this is the .Java
D:\>
```