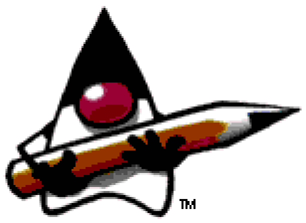
Introduction to Java Programming

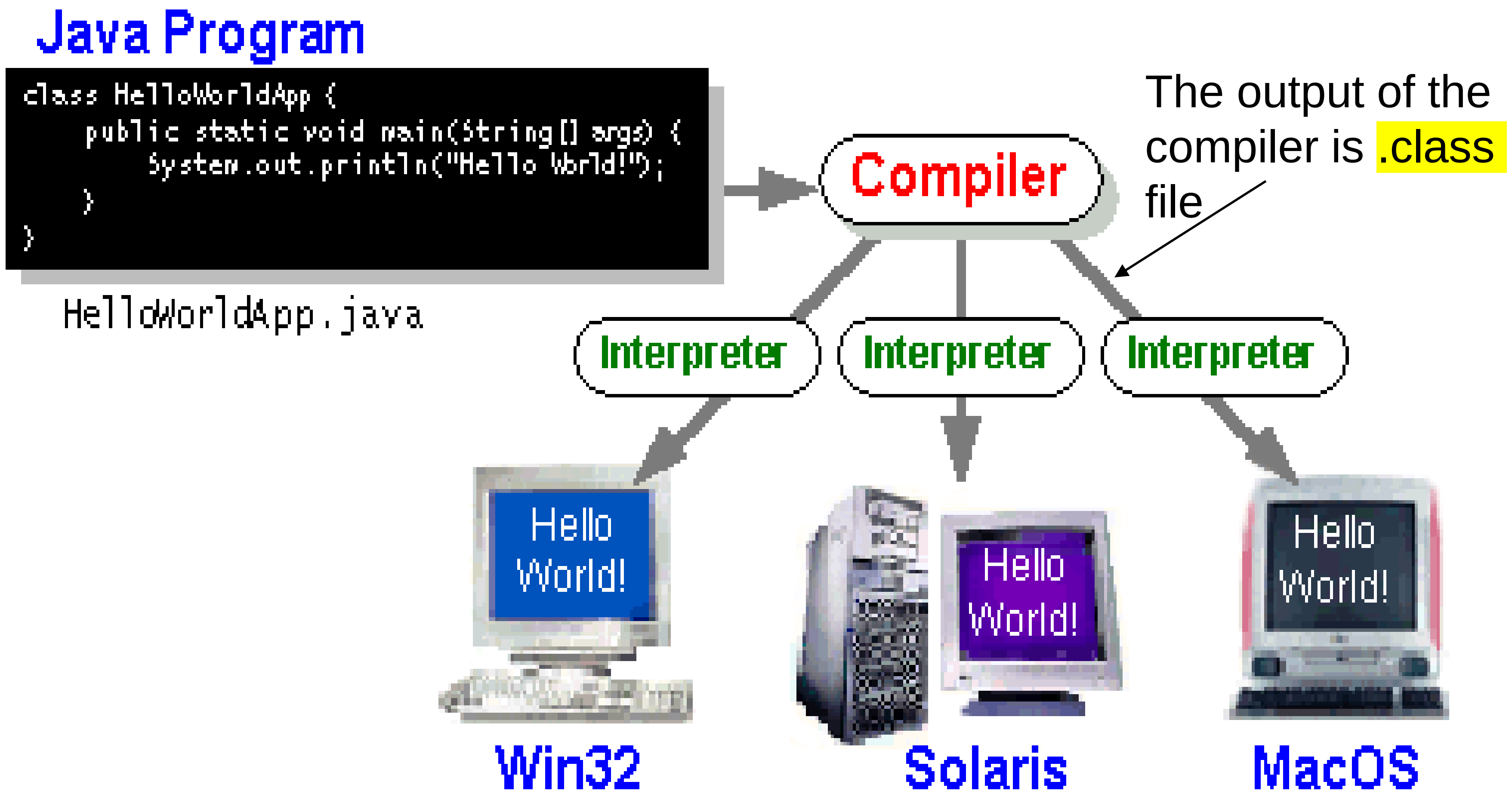
Based on slides for Building Java Programs by Reges/Stepp, found at
http://faculty.washington.edu/stepp/book/

Java

- There are hundreds of **high-level** computer languages. Java, C++, C, Basic, Fortran, Cobol, Lisp, Perl, Prolog, Eiffel, Python
- The capabilities of the languages vary widely, but they all need a way to do
  - declarative statements
  - conditional statements
  - iterative or repetitive statements
- A compiler is a program that converts commands in high level languages to machine language instructions



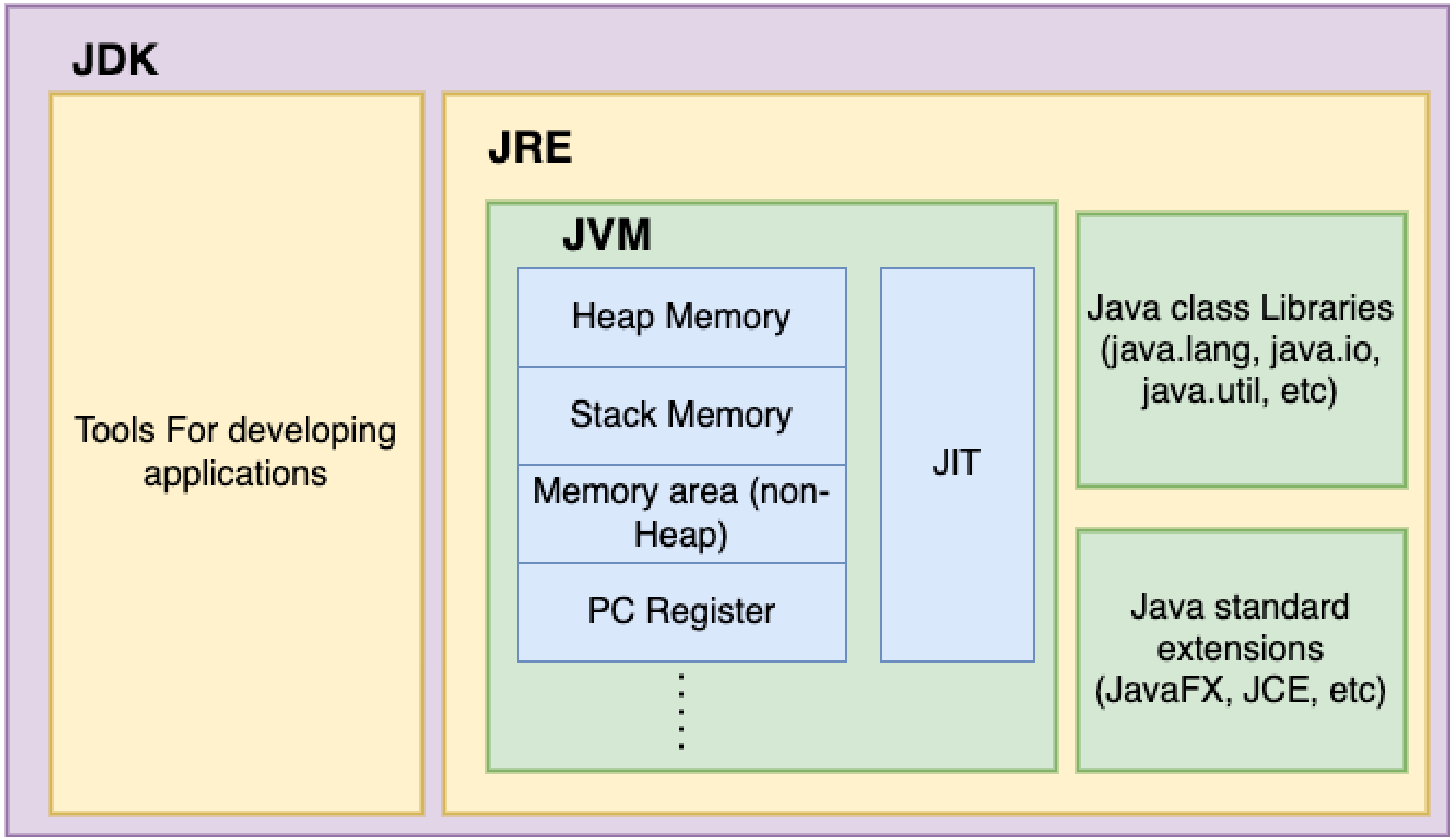
A Picture is Worth…



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

The output of the compiler is .class file

The Interpreter's are sometimes referred to as the Java Virtual Machines


JDK
Tools For developing applications
JRE
JVM
Heap Memory
Stack Memory
Memory area (non-Heap)
PC Register
JIT
Java class Libraries (java.lang, java.io, java.util, etc)
Java standard extensions (JavaFX, JCE, etc)

A Simple Java Program

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

More Definitions

- **code or source code**: The sequence of instructions in a particular program.
  - The code in this program instructs the computer to print a message of **Hello, world!** on the screen.
- **output**: The messages printed to the computer user by a program.
- **console**: The text box or window onto which output is printed.

Compiling and Running

- **Compiler**: a program that converts a program in one language to another language
  - compile from C++ to machine code
  - compile Java to *bytecode*

- **Interpreter**: A converts one instruction or line of code from one language to another and then executes that instruction
  - When java programs are run the bytecode produced by the compiler is fed to an interpreter that converts it to machine code for a particular CPU

The command line

To run a Java program using your Command Prompt:

- change to the directory of your program

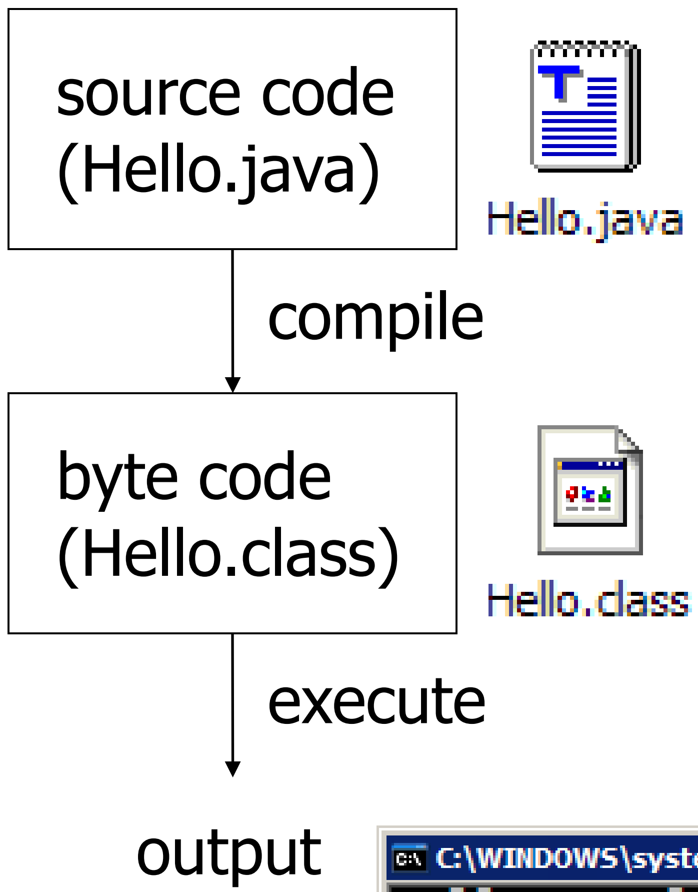
  ```
  cd
  ```

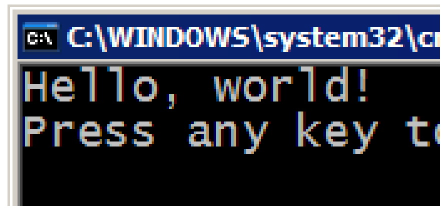
- compile the program

  ```
  javac Hello.java
  ```

- execute the program

  ```
  java Hello
  ```



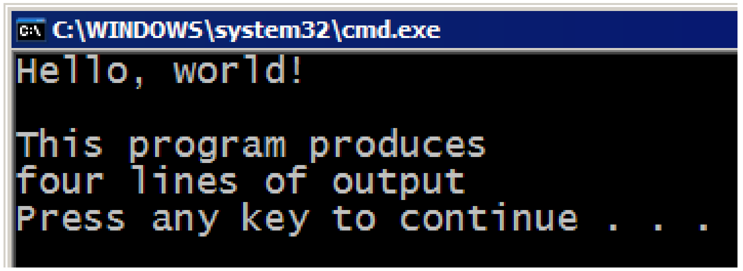


```
Hello, world!
Press any key t
```

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.



```
Hello, world!

This program produces
four lines of output
Press any key to continue . . .
```

Structure of Java programs

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

- Every executable Java program consists of a **class**...
  - that contains a **method** named `main`...
    - that contains the **statements** to be executed
- The previous program is a class named `Hello`, whose `main` method executes one statement named `System.out.println`

Java terminology

- **class**:
  (a) A module that can contain executable code.
  (b) A description of a type of objects. (seen later)
- **statement**: An executable piece of code that represents a complete command to the computer.
  - every basic Java statement ends with a semicolon `;`
- **method**: A named sequence of statements that can be executed together to perform a particular action or computation.

Syntax and syntax errors

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

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

Compiler Output

- The program on the previous slide produces the following output when we attempt to compile it

compiler output:

```
H:\summer\Hello.java:2: <identifier> expected
    pooblic static void main(String[] args) {
            ^
H:\summer\Hello.java:5: ';' expected
}
^
2 errors
Tool completed with exit code 1
```

System.out.println

- Java programs use a statement called `System.out.println` 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` :
  - 1. `System.out.println("`***<Message>***`");`
    - Prints the given message as a line of text on the console.
  - 2. `System.out.println();`
    - Prints a blank line on the console.

Static method syntax

- The structure of a static method:

```
public class <Class Name> {

    public static void <Method name> () {
        <statements>;
    }

}
```

Methods calling each other

- One static method may call another:

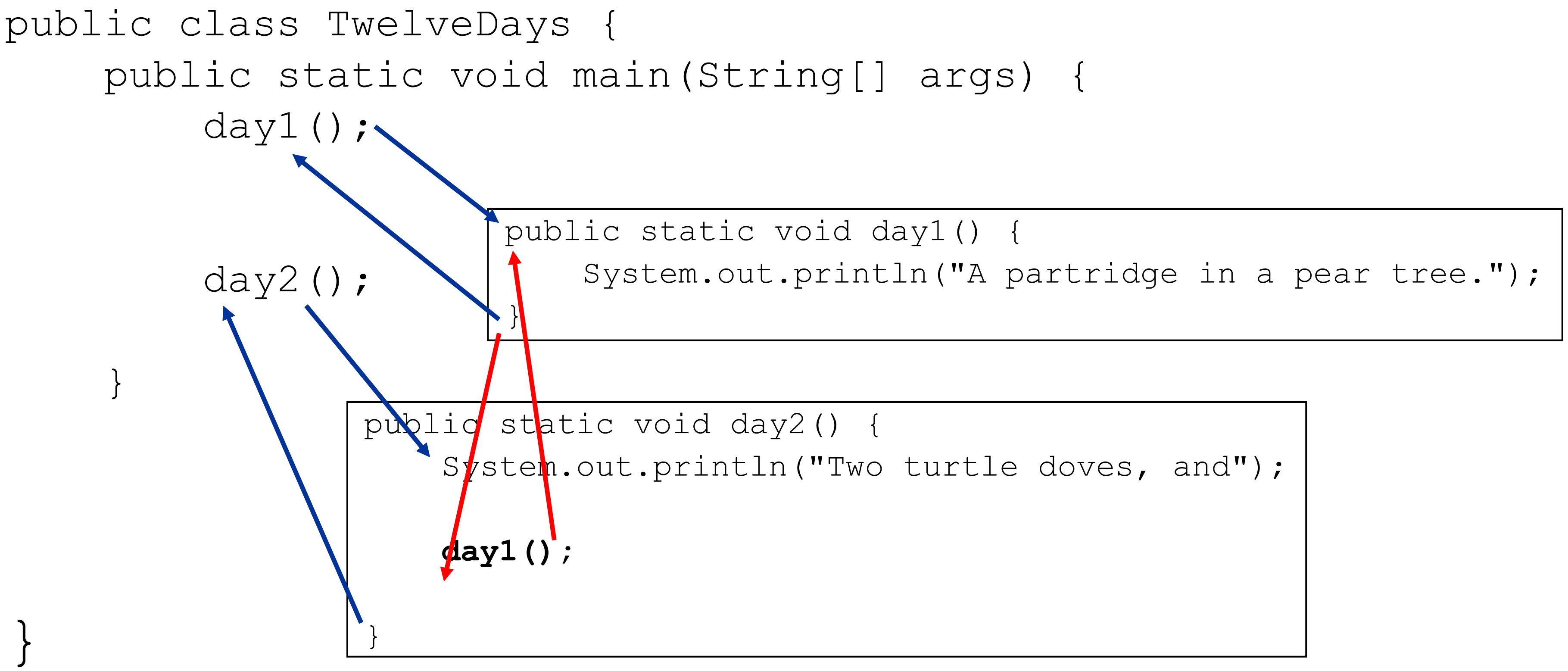
```
public class TwelveDays {
    public static void main(String[] args) {
        day1();
        day2();
    }

    public static void day1() {
         System.out.println("A partridge in a pear tree.");
    }

    public static void day2() {
         System.out.println("Two turtle doves, and");
         day1();
    }
}
```

Program's output:

```
A partridge in a pear tree.
Two turtle doves, and
A partridge in a pear tree.
```

Control flow of methods

- When a method is called, a Java program 'jumps' into that method, executes all of its statements, and then 'jumps' back to where it started.



```
public class TwelveDays {
    public static void main(String[] args) {
        day1();

        day2();

    }
}
```

```
public static void day1() {
    System.out.println("A partridge in a pear tree.");
}
```

```
public static void day2() {
    System.out.println("Two turtle doves, and");

    day1();

}
```

Identifiers

- **identifier**: A name that we give to a piece of data or part of a program.
  - Identifiers are useful because they allow us to refer to that data or code later in the program.
  - Identifiers give names to:
    - classes
    - methods
    - variables (named pieces of data; seen later)

- The name you give to a static method is an example of an identifier.
  - What are some other example identifier we've seen?

Details about identifiers

- Java identifier names:
  - first character must a letter or _ or `$`
  - following characters can be any of those characters or a number
  - identifiers are case-sensitive; `name` is different from `Name`

- Example Java identifiers:
  - legal:
    ```
    olivia        second_place    _myName
    TheCure       ANSWER_IS_42    $variable
    ```
  - illegal:
    ```
    me+u          :-)             question?
    side-swipe    hi there        ph.d
    belles's      2%milk
    kelly@yahoo.com
    ```

Keywords

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

- Complete list of Java keywords:

`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`	

- You may not use `char` or `while` or `this` or any other keyword for the name of a class or method; Java reserves those words to mean other things.
  - You could use CHAR, While, or ThIs, because Java is case-sensitive. However, this could be confusing and is not recommended.

Comments

- **comment**: A note written in the source code by the programmer to make the code easier to understand.
  - Comments are not executed when your program runs.
  - Most Java editors turn your comments a special color to make it easier to identify them.
- Comment, general syntax:

  `/*` ***<comment text; may span multiple lines>*** `*/`

  or,

  `//` ***<comment text, on one line>***
- Examples:
  - `/* A comment goes here. */`

  -
    ```
    /* It can even span
    multiple lines. */
    ```
  - `// This is a one-line comment.`