

# Building Java Programs

Chapter 1

Lecture 1-1: Introduction; Basic Java Programs

**reading: 1.1 - 1.3**

# 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.



# Course principles

- Lots of resources and people who want to help you
- Deliberate topic progression
- Coherence between lectures, sections, labs, homework, exams
- What you **do** will determine what you learn

# Take this course if you...

- ... like solving tricky problems
- ... like building things
- ... (will) work with large data sets
- ... are curious about how Facebook, Google, etc work
- ... have never written a computer program before
- ... are shopping around for a major
  - 142 is a good predictor of who will enjoy and succeed in CSE

# Why Java?

- Relatively simple
- Object-oriented
- Pre-written software
- Platform independent (Mac, Windows...)
- Widely used
  - #1 in popularity ie <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

# Compiling/running a program

## 1. Write it.

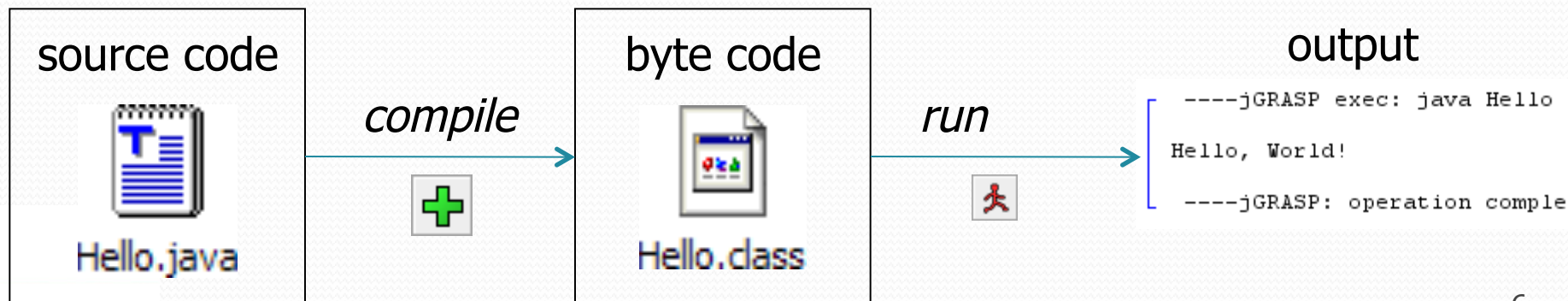
- **code** or **source code**: The set of instructions in a program.

## 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

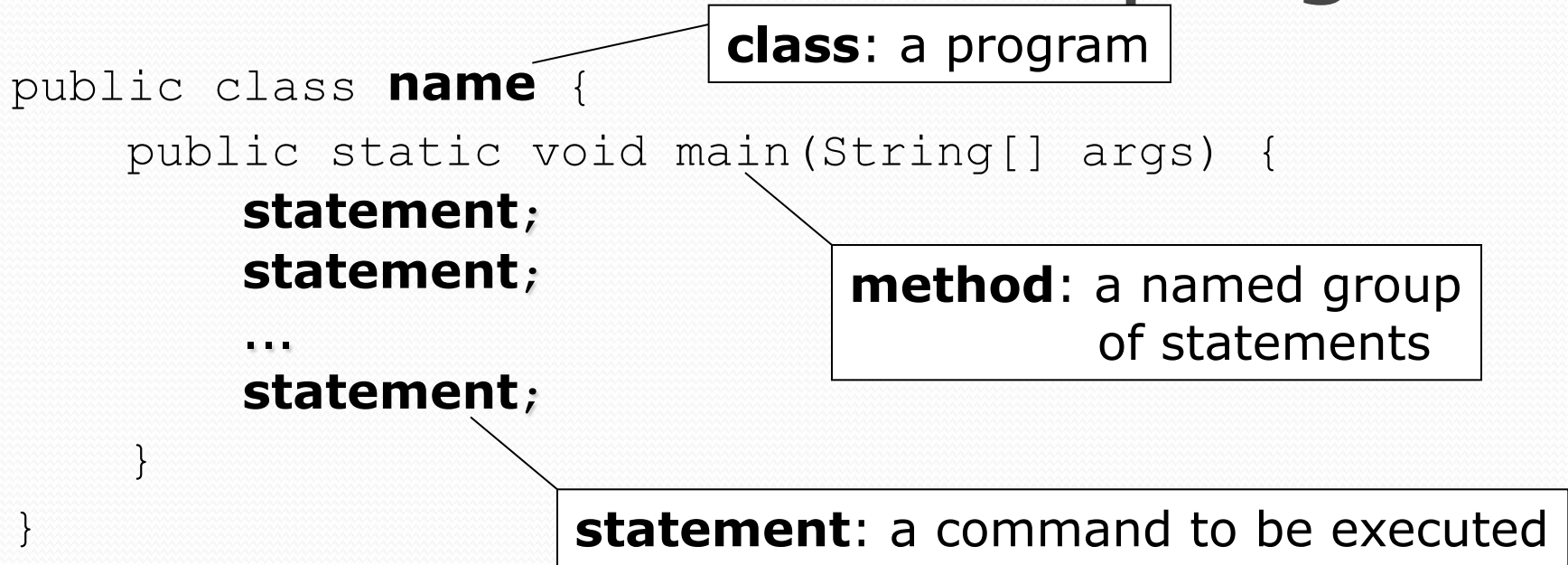
```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

- **Its output:**

Hello, world!

This program produces  
four lines of output

# Structure of a Java program



- 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	<b>public</b>	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	<b>static</b>	<b>void</b>
char	finally	long	strictfp	volatile
<b>class</b>	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

- i.e., You may not use `char` or `while` for the name of a class.

# 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.

Examples:

- 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.

`\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 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 this 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 this 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 this 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 \"");
```