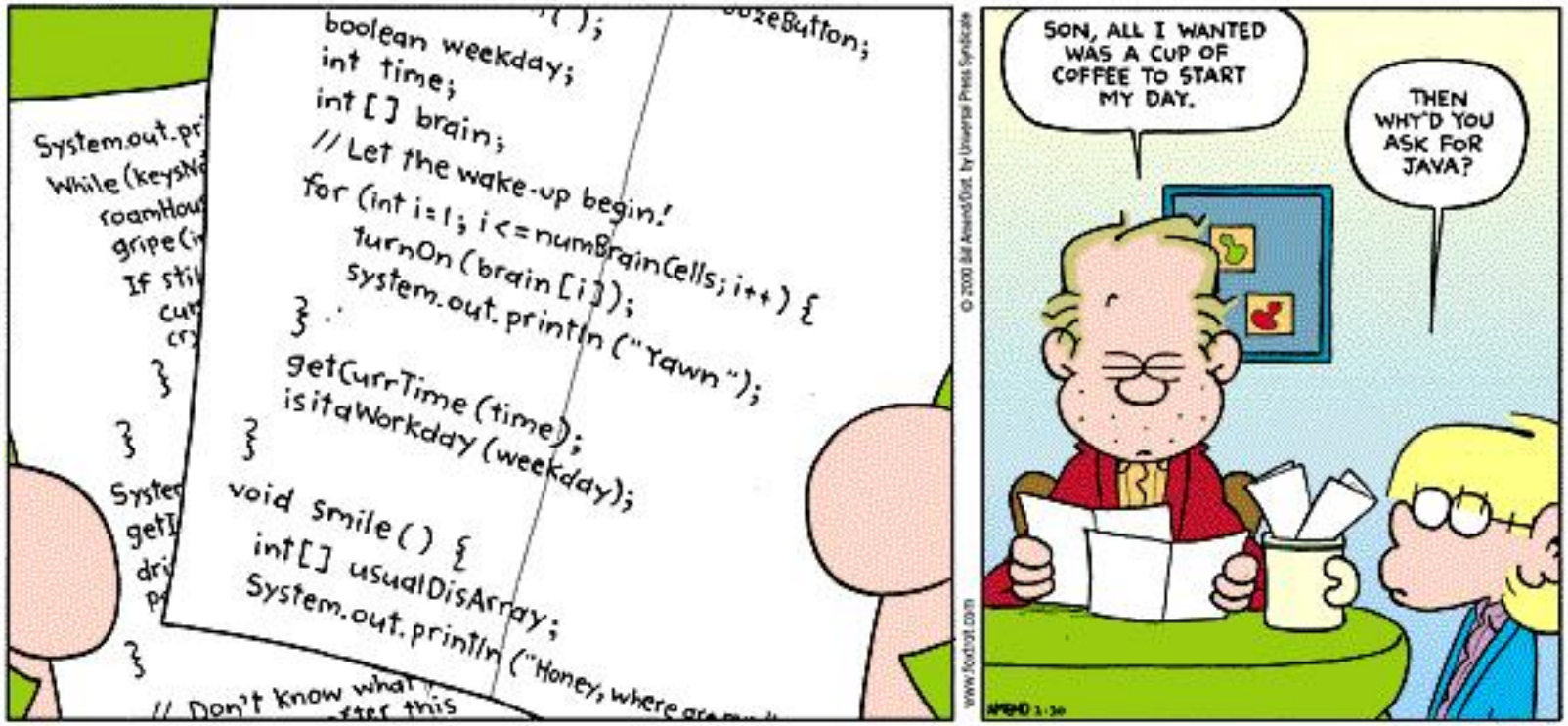
CSE 142, Fall 2011

Building Java Programs Chapter 1
Lecture 1-1: Introduction; Basic Java Programs

reading: 1.1 - 1.3

Welcome to CSE 142!



Course Staff

- Hélène Martin (pronounced L-N)
- Marty Stepp
  - Textbook co-author
  - Python session, labs, office hours, lots behind the scenes
- Pim Lustig (pl@cs.washington.edu)
  - Course registration, sections, etc.
- TAs
  - Your primary point of contact
  - Ask them about their experiences in CSE

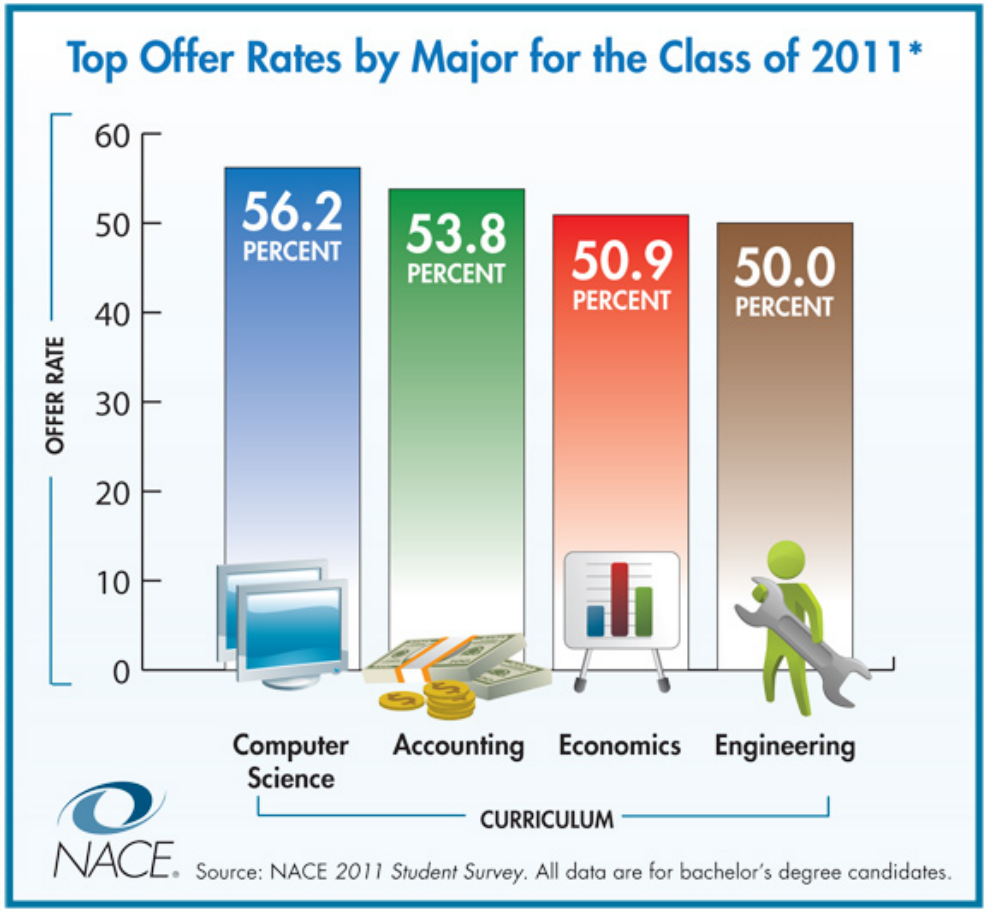
Computer Science

- Science?
  - More like engineering, art, magic…
- CS is still a young field finding itself
- CS is about PROCESS – how to accomplish a task
- Computers are a tool
  - What kinds of problems can they solve?
  - How can they be made faster, cheaper, more efficient…?

Take this course if you…

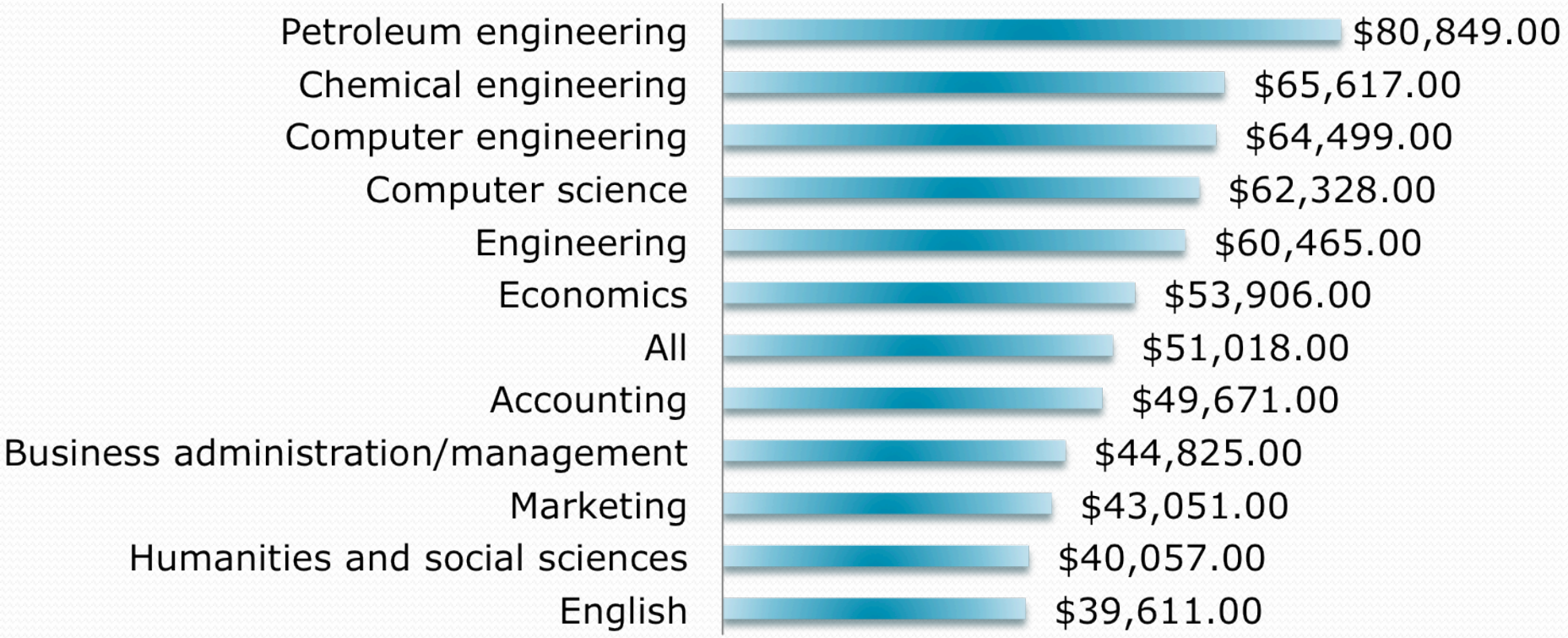
- … like solving tricky problems
- … like building things
- … (will) work with large data sets
- … are curious about how Facebook, Google, etc work
- … have never written a computer program before
- … are shopping around for a major
  - 142 is a good predictor of who will enjoy and succeed in CSE

Jobs before graduation



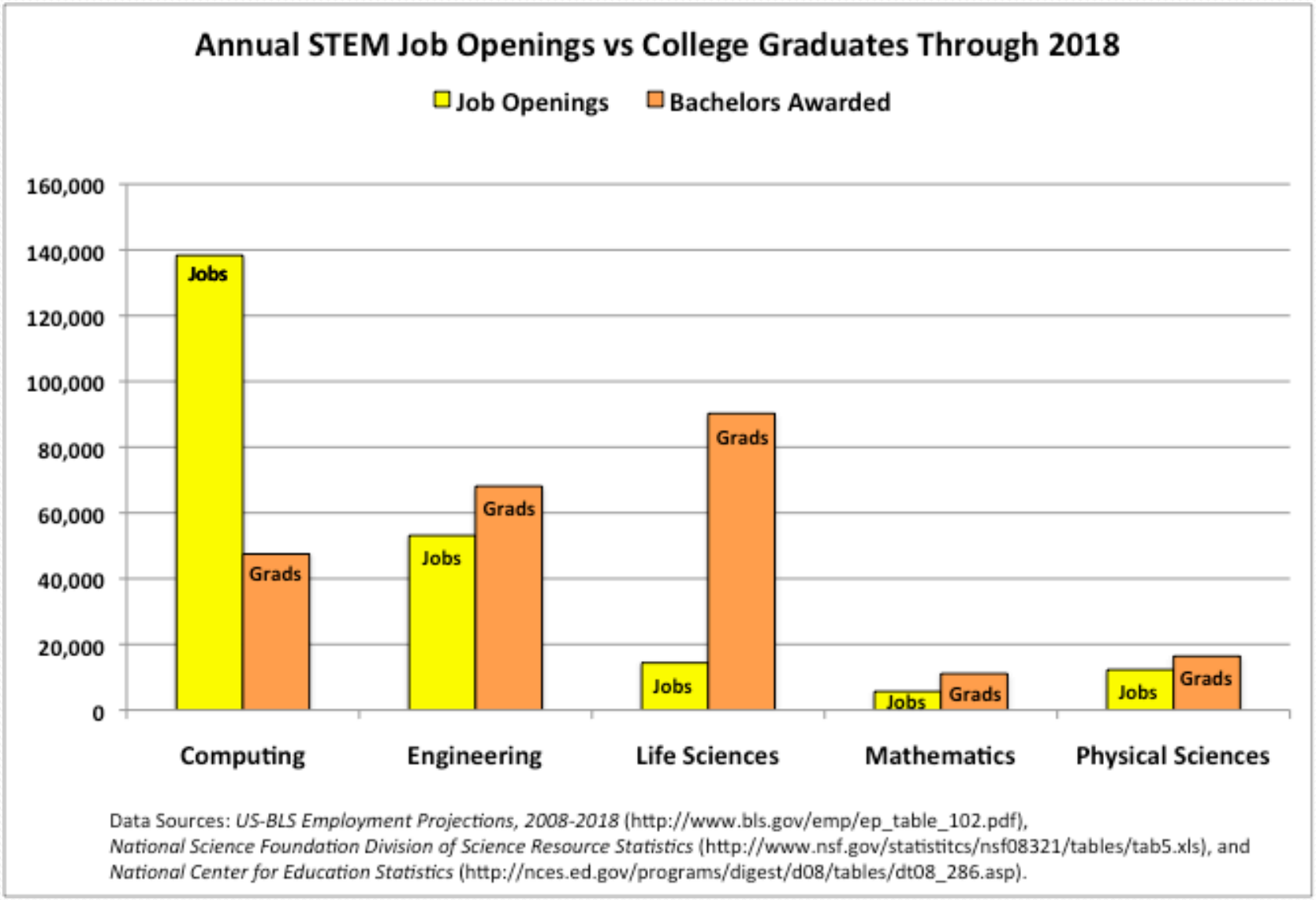
- English: 23.5%
- Healthcare: 28.7%

Starting salaries

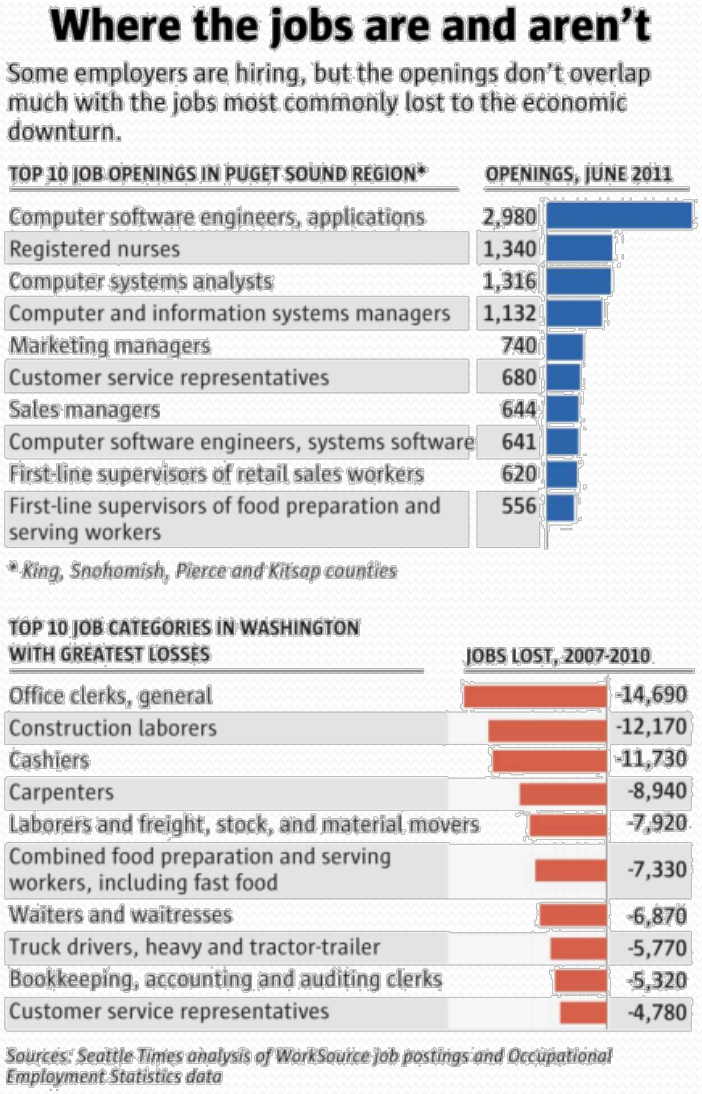


Source: Summer 2011 Salary Survey, National Association of Colleges and Employers. Data are for Bachelor's Degree candidates.

High-demand for talent



Jobs in the Seattle area



Diverse opportunities

- Software shops (Microsoft, Amazon, Google, Facebook…)

- Hard sciences (computational biology…)
- Engineering (simulations…)
- Healthcare (data management…)
- Education (math…)
- International development (data gathering…)

Course goals

- By the end of the course, you will:
  - write medium-scale programs to solve real problems
  - know some of the kinds of problems computers can solve
  - recognize beautiful code
  - recognize ugly hacks

What is 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.

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

Why Java?

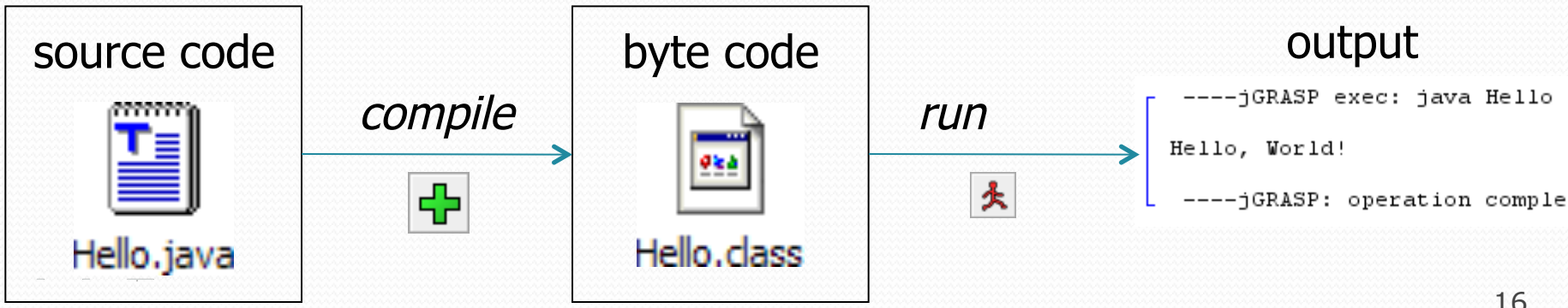
- Relatively simple
- Object-oriented
- Pre-written software
- Platform independent (Mac, Windows…)
- Widely used
  - #1 in popularity ie http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Basic Java programs with `println` statements

reading: 1.2 - 1.3

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

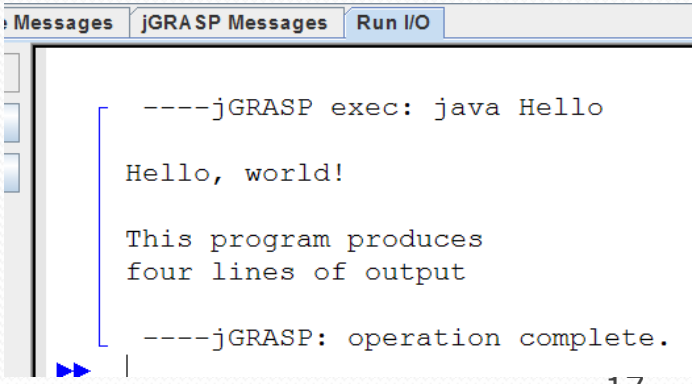
```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

- Its output:

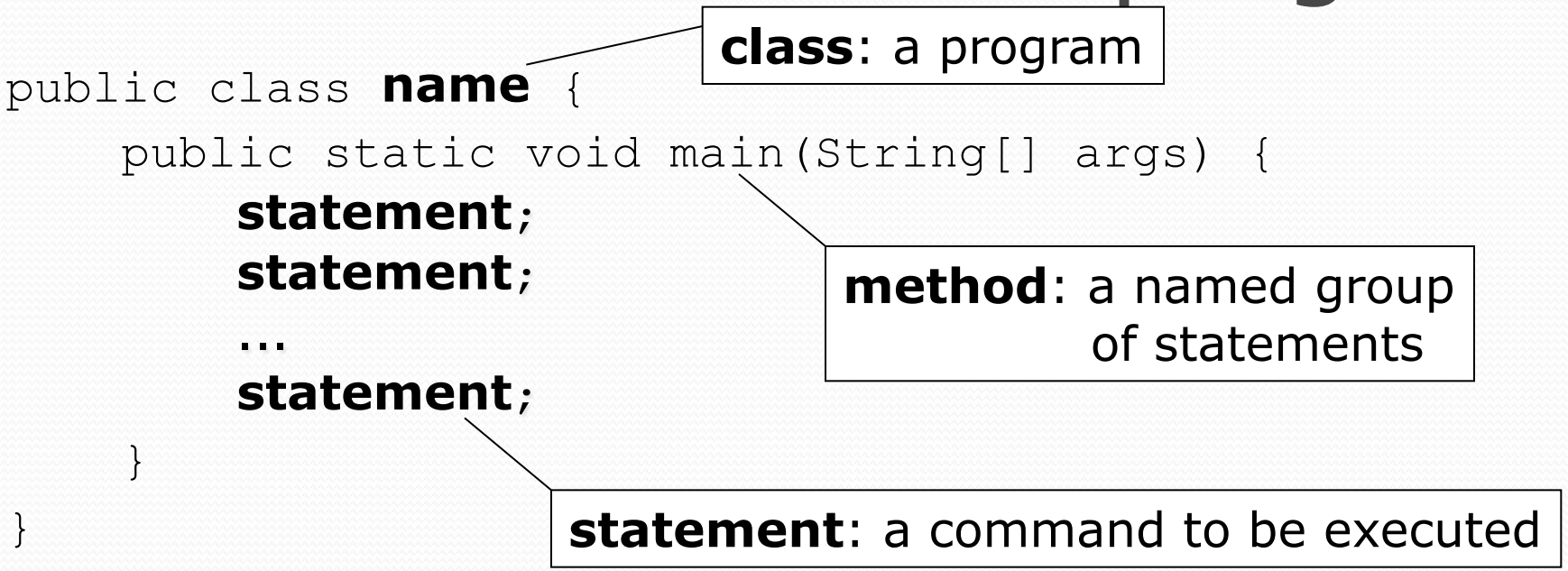
```
Hello, world!

This program produces
four lines of output
```

- **console**: Text box into which the program's output is printed.



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.

Names and identifiers

- You must give your program a name.

```
public class IrishPoetry {
```

  - Naming convention: capitalize each word (e.g. `MyClassName`)
  - Your program's file must match exactly (`IrishPoetry.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	**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	

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.
  - 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!

System.out.println

- A statement that prints a line of output on the console.
  - pronounced "print-linn" (NOT 'print-L-N')
  - 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.

Strings and escape sequences (section)

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 quote is from
Irish poet Oscar Wilde:

"Music makes one feel so romantic
- at least it always gets on one's nerves -
which is the same thing nowadays."
```

- 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 quote is from");
System.out.println("Irish poet Oscar Wilde:”);
System.out.println();
System.out.println("\"Music makes one feel so romantic");
System.out.println("- at least it always gets on one's nerves -");
System.out.println("which is the same thing nowadays.\"");
```

- `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 \"");
```