



Welcome to CSE 142!

Zorah Fung

University of Washington, Winter 2017

Building Java Programs Chapter 1

Lecture 1: Introduction; Basic Java Programs

reading: 1.1 - 1.3

What is computer science?

- computers?
- science?

PROCESS

al·go·rithm:

a step-by-step procedure for solving a problem or accomplishing some end *especially by a computer*

Fields of computer science

- Graphics
- Computer Vision
- Artificial Intelligence
- Robotics
- Machine Learning
- Data Mining
- Natural Language Processing
- User Interfaces
- ...

- How does this all relate to programming?
 - This course is “Introduction to Programming I” after all.

Programming is like Legos...





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 CSE
 - ... think “computers and robots are going to take over the world. I want to befriend them so that my life will be spared.”



Syllabus Time

Tips for Success

- Come to lecture!
- Visit website often: cs.uw.edu/142
- Utilize the resources we provide you:
 - IPL (MGH 334)
 - Come visit me in Office Hours!
 - Your TA
 - Textbook
 - Slides and Lecture examples
 - Message Board
 - Practice-It! <http://practiceit.cs.washington.edu/practiceit/>
- Remember: assignments must be **your own work!**

Tips for Success (cont'd)

- Keep up with the assignments
 - The course material is cumulative
- If you don't understand something, ask questions (especially "WHY?").
 - There's no such thing as a dumb question.
 - Computers are neither magical nor mysterious. Everything can be explained!

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.
 - We will be using a programming language called Java.



Why Java?

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

Your first Java program!

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

- File must be named `Hello.java`
- What does this code *output* (print to the user) when you *run* (execute) it?

Bigger Java program!

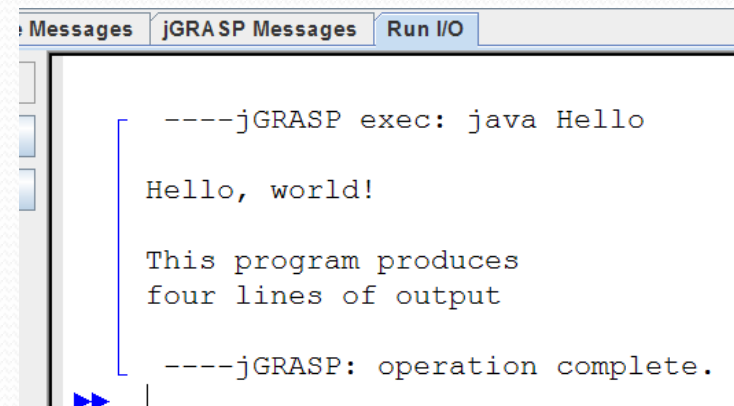
```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

- Its output:

Hello, world!

This program produces
four lines of output

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



The screenshot shows a console window with three tabs: 'Messages', 'jGRASP Messages', and 'Run I/O'. The 'Run I/O' tab is active and displays the following output:

```
----jGRASP exec: java Hello  
Hello, world!  
This program produces  
four lines of output  
----jGRASP: operation complete.
```

Running a program

1. Write it.

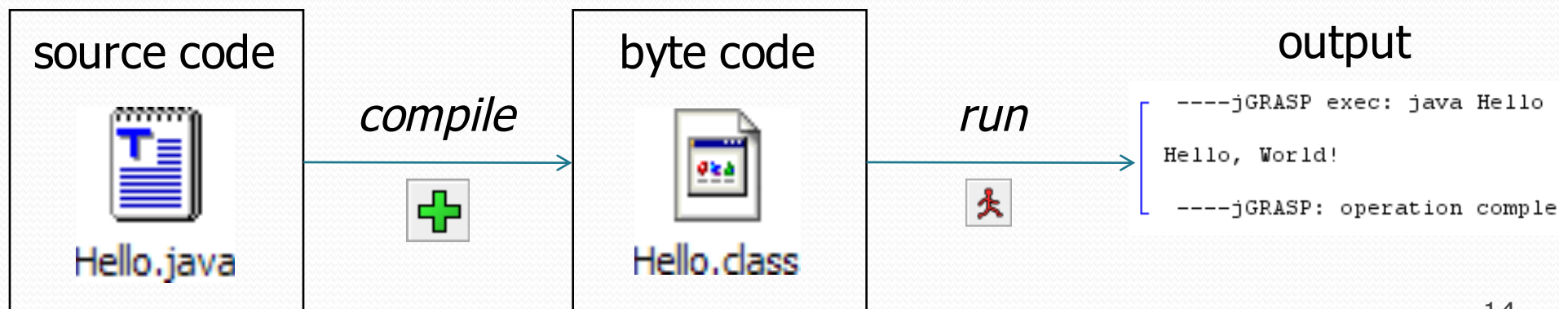
- **code** or **source code**: The set of instructions in a program.

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.



Structure of a Java program

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

class: a program

method: a named group of statements

statement: a command to be executed

- 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 HelloWorld {
```

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

System.out.println

- A statement that prints a line of output on the console.
 - pronounced "print-linn"
- 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.

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 | |

- **Note:** Because Java is case-sensitive, you could technically use `Class` or `cLaSs` as identifiers, but this is very confusing and thus **strongly discouraged**.

Syntax

- **syntax:** The set of legal structures and commands that can be used in a particular language.
 - The “spelling” and “grammar” of a programming 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
 - Class and file names do not match
 - ...

Syntax error example

```
1 public class Hello {
2     poublic static void main(String[] args) {
3         System.owt.println("Hello, world!")_
4     }
5 }
```

- **Compiler output:**

```
Hello.java:2: <identifier> expected
    poublic 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!
 - Why can't the computer just say "You misspelled 'public'"?

More on syntax errors

- Java is case-sensitive
 - Hello and hello are not the same

```
1 Public class Hello {
2     public static void main(String[] args) {
3         System.out.println("Hello, world!");
4     }
5 }
```

compiler output:

```
Hello.java:1: class, interface, or enum expected
Public class Hello {
^
1 error
```

First lesson in this class

- Computers are stupid.
- Computers can't read minds.
- Computers don't make mistakes.
- If the computer is not doing what you want, it's because **YOU** made a mistake.



Strings and escape sequences

Strings

- **string:** A sequence of text characters.
 - Starts and ends with a " (quotation mark 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 \"");
```