Building Java Programs

Chapter 1:
Introduction to Java Programming

Lecture outline

- Introduction
  - Syllabus and policies
  - What is computer science
  - Programs and programming languages

- Basic Java programs
  - Output with println statements
  - Syntax and errors
  - String literals and escape sequences

Introduction

reading: 1.1

self-check: #1-4

The Marty and Hélène show

Marty Stepp

- stepp@cs.washington.edu
- AKA 'the mentor'
- Office: CSE 466
- UW CSE Lecturer
- Likes riding in golf carts

Hélène Martin

- ln@cs.washington.edu
- Office: CSE 270
- Former CSE undergrad
- Wants to save the world
- Loves robots

Syllabus in a nutshell

- Participation points for going to section
- Textbook STRONGLY recommended
- All staff are awesome and have office hours
- jGRASP is recommended editor
- Schedule exam makeups BEFORE test
- 5 'late days'
- Don't share code; obey academic integrity policy

CSE?!

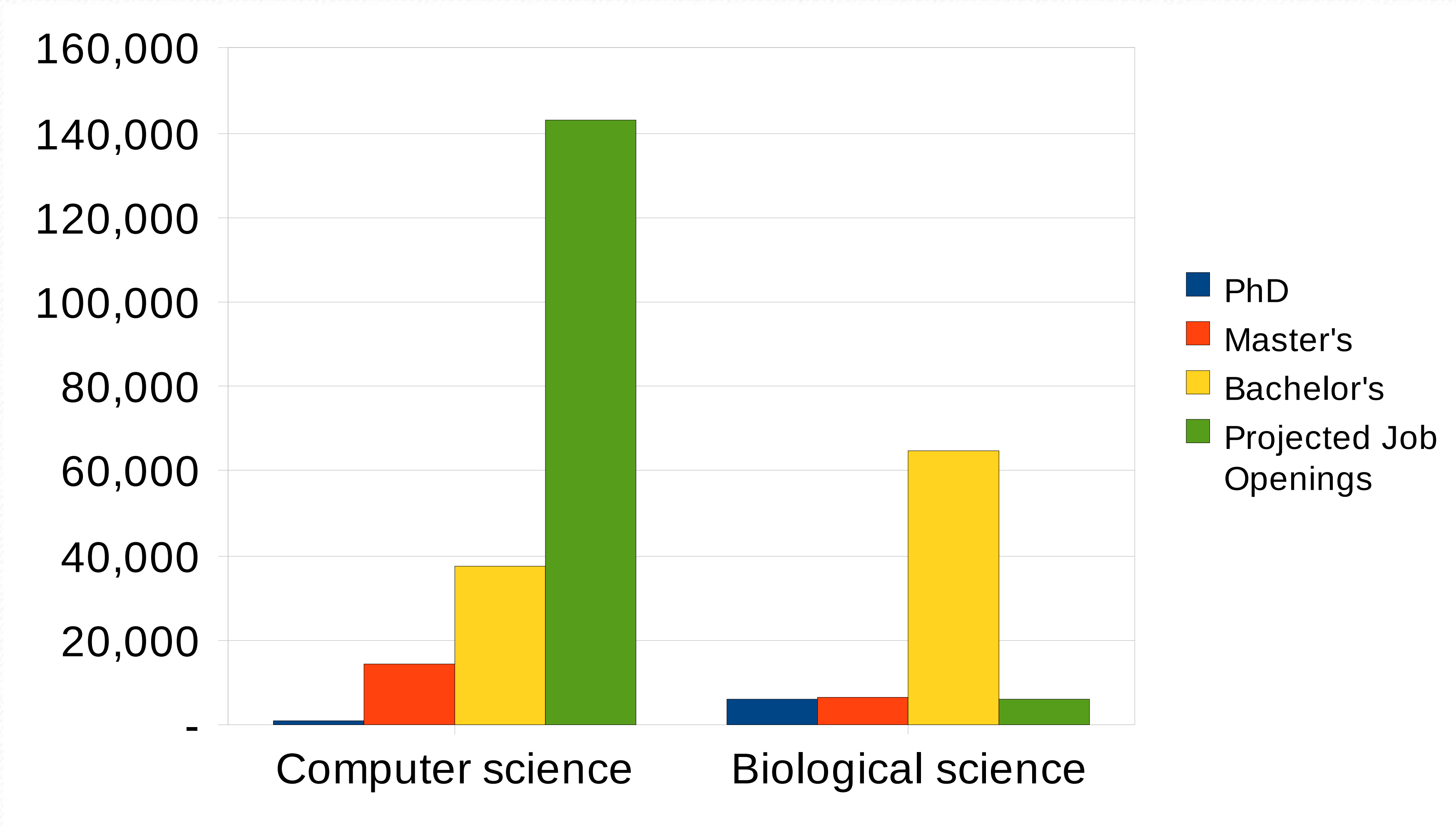
- Computer Science
  - Study of computation (*information processing*)
  - Many subfields
  - Graphics
  - Computer Vision
  - Artificial Intelligence
  - ...
- Computer Engineering
  - Overlap with CS and electrical engineering
  - Emphasis on hardware-software integration

Computer Programming I

- Computer Programming
  - Creating instructions for computers
  - Uses a language (*human-readable notation*)
  - For many, a means to an end
  - We'll explore "cool stuff"

- One
  - There's more!
  - 143 is a direct continuation
  - Programming well is a lifelong endeavor

Cool, but will I get a job?



SOURCES (via Benson Limketkai): Tabulated by National Science Foundation/Division of Science Resources Statistics; data from Department of Education/National Center for Education Statistics: Integrated Postsecondary Education Data System Completions Survey; and NSF/SRS: Sur

How to do well

- Attention to detail
- Keeping up with assignments
- Using resources
  - Office hours
  - ASK QUESTIONS
- Accepting that computers are dumb
  - They do EXACTLY what you ask them

Java

- A modern programming language
  - Sun Microsystems in 1995
  - Rich libraries
  - Cross-platform
  - Object-oriented
- Taught in 142 and 143
  - Shows basic concepts
  - Good free, cross-platform tools
  - Industry-grade language
- It's not the only language out there
  - CSE Cool Stuff

Basic Java programs with `println` statements

reading: 1.2 - 1.3

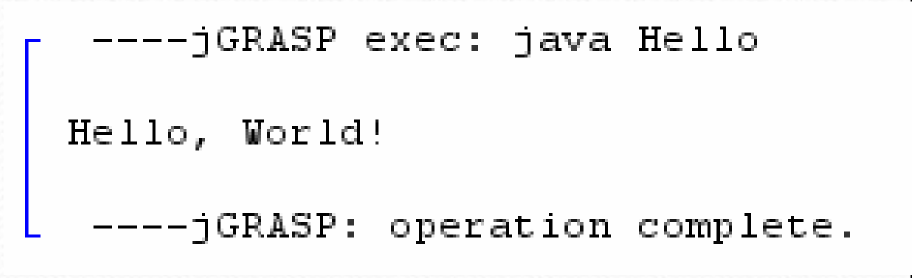
self-check: #5-15

exercises: #1-4

A basic Java program

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

- **code** or **source code**: The sequence of instructions in a program.
  - The code in this program instructs the computer to display a message.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
  - Some editors pop up the console as an external window, and others contain their own console window.

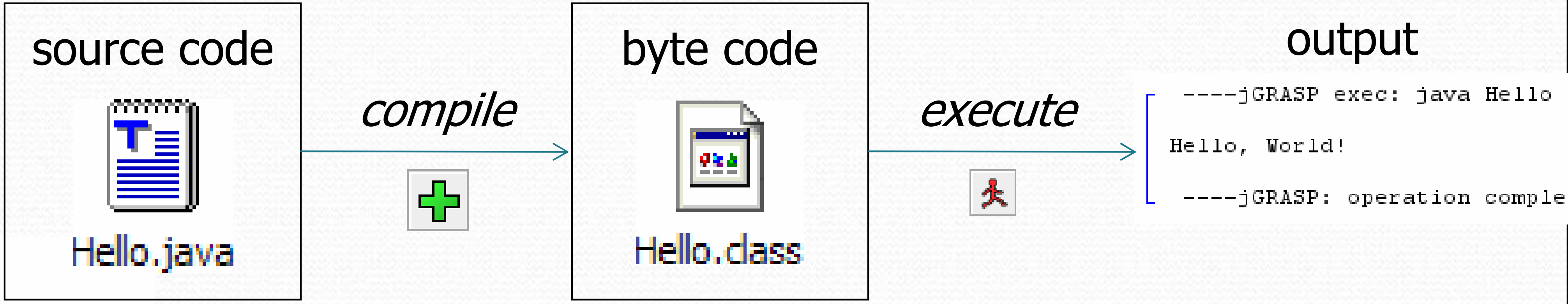


```
 ----jGRASP exec: java Hello

Hello, World!

 ----jGRASP: operation complete.
```

Compiling/running a program

Before you run your programs, you must *compile* them.

- **compiler**: Translates a computer program written in one language into another language.
  - Java Development Kit (JDK) includes a Java compiler.
  - **byte code**: The Java compiler converts your source code into a format named *byte code* that can be executed on many different kinds of computers.



```
 ----jGRASP exec: java Hello

Hello, World!

 ----jGRASP: operation comple
```

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");
    }
}
```

- Its output:

```
Hello, world!

This program produces
four lines of output
```

Structure of Java programs

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

- Every executable Java program consists of a **class**
  - that contains a **method** named `main`
    - that contains the **statements** (commands) to be executed

Java terminology

- **class**: A module that can contain executable code.
  - Every program you write will be a class.
- **statement**: An executable command to the computer.
- **method**: A named sequence of statements that can be executed together to perform a particular action.
  - A special method named `main` signifies the code that should be executed when your program runs.
  - Your program can have other methods in addition to `main`. (seen later)

Identifiers

- **identifier**: A name given to an item in your program, e.g:
  - a class
  - a method (next lecture)
  - a variable or constant value (Ch. 2)

- conventions for naming in Java:
  - *classes*: capitalize each word (`ClassName`)
  - *methods*: capitalize each word after the first (`methodName`)

Details about identifiers

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

- Example Java identifiers:

legal:	`susan`	`second_place`	`_myName`
	`TheCure`	`ANSWER_IS_42`	`$variable`
illegal:	`me+u`	`49er`	`question?`
	`side-swipe`	`hi there`	`ph.d`
	`jim's`	`2%milk`	`suzy@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` for the name of a class or method; Java reserves those to mean other things.
  - You could use `CHAR` or `While`, because Java is case-sensitive.

Syntax

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

- some Java syntax:
  - every basic Java statement ends with a semicolon ;
  - The contents of a class or method occur between { and }

Syntax errors

- **syntax error** or **compiler error**: A problem in the structure of a program that causes the compiler to fail.

```
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:5: ';' expected
}
^
2 errors
```

Fixing syntax errors

- Error messages can be hard to understand:

```
Hello.java:2: <identifier> expected
    pooblic static void main(String[] args) {
           ^
```

- The compiler displays the line where it found the error.

```
1 public class MissingSemicolon {
2     public static void main(String[] args) {
3         System.out.println("A rose by any other name")
4         System.out.println("would smell as sweet");
5     }
6 }
```

```
MissingSemicolon.java:4: ';' expected
System.out.println("would smell as sweet");
^
```

`System.out.println`

- `System.out.println` : A statement 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` :

  `System.out.println("`***<text>***`");`

  Prints the given message as a line of text on the console.

  `System.out.println();`

  Prints a blank line on the console.

Strings and string literals

- **string**: A sequence of text characters that can be printed or manipulated in a program.
  - sometimes also called a *string literal*
  - strings in Java start and end with quote characters "
  - Examples:

```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

String restrictions

- A string may not span across multiple lines.

```
"This is not
a legal String."
```

- A string may not contain a " character. (' is okay)

```
"This is not a "legal" String either."
"This is 'okay' though."
```

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 \"");
```

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.

- Comment, general syntax:

  `//` ***<comment text, on one line>***

  or,

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

- Examples:

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

Comments example

```
/* Suzy Student, CS 101, Fall 2019
   This program prints lyrics from my favorite song! */
public class BaWitDaBa {
    // The code to print the song on the console.
    public static void main(String[] args) {
        // first verse
        System.out.println("Bawitdaba");
        System.out.println("da bang a dang diggy diggy");

        // separate the lyrics with a blank line
        System.out.println();

        // second verse
        System.out.println("diggy said the boogy");
        System.out.println("said up jump the boogy");
    }
}
```