CSE 142

Lecture 1

Course Introduction; Basic Java

Welcome

Today:

- Course mechanics
- A little about computer science & engineering (CSE)
  - And how this course relates
- *Java* programs that print text

Handouts

- Anonymous survey
- Non-anonymous survey: to your TA
- Syllabus: read it carefully and entirely

Turn in surveys at end of class

Course Information

- Two instructors; one course
- Lectures
  - Core material; see programs developed
  - Often spend time writing code rather than the slides
- Section
  - Required (attendance = 1 homework assignment)
  - Examples; practice
- Textbook
  - Required; written exactly for this course
  - Allows lectures/section to focus on design rather than details
- *Python* sessions: optional (more details later)

Course Resources

Many resources for students

- http://www.cs.washington.edu/142
  - all resources, slides, example programs, etc.
- IPL: Mary Gates room 334
- Your TA
- Instructions for downloading course software
- Videos on publisher's textbook site
- Practice-It!
- Discussion Board

Homework & Exams

- Weekly homework assignments
  - Very related to lectures, section, textbook
  - Usually due Tuesday evening (see syllabus for late days)
  - Graded by your TA
  - 50% of your grade
- Midterm (20%) and final (30%)
  - No make-ups
  - Open book, open notes
  - Samples, review sessions, etc. later
- Grade questions
  - TAs can fix obvious errors
  - For more complex issues, request a regrade

Academic Integrity

- Do not cheat in our class!
  - Vigorous enforcement out of fairness for most students
  - We use homework-comparison software
- Discussing the general terms of the homework and the overall approach is allowed and helpful, but…
  - Assignments are individual effort
  - Never share code or "walk through" a detailed approach
  - Do not get line-by-line help
  - Giving too much help is as bad as receiving it
- Full explanation in the syllabus – read it carefully

Advice / expectations, part 1

- Attend lecture on-time and well-rested
  - 9:28 / 11:28 much more useful than 9:35 / 11:35
  - Do not pack up at 10:19, 12:19 (unfair to neighbors)
- Take lecture notes or otherwise participate actively
- Keep up with textbook reading
- Work through section-handout problems
- Start homework early
- Basically, eat your vegetables ☺
  - But also have fun – this is great stuff so enjoy it!

Advice / expectations, part 2

Be open to this course *changing your life*

- Software has changed *everything*: science, engineering, social science, business, economics, advertising, politics, communication, …
  - Course is important for non-CSE majors
- Many students discover an interest in CS and the CSE major via this course
  - Example: Grossman never wrote a program before college
  - Course designed for students with no experience
  - Great to be a "tortoise" instead of a "hare"

End of mechanics

Questions about course logistics?

Welcome

Today:

- Course mechanics
- A little about computer science & engineering
  - And how this course relates
- *Java* programs that print text

Myths

- CSE majors spend lives in dark offices alone writing code
- CSE is only about lines and lines of details
- CSE is not creative
- CSE is only for people that want to work at large software companies
- All the jobs moved to Bangalore
- …

See: http://www.cs.washington.edu/whycse

Another view

"Computer science is no more about computers than astronomy is about telescopes." Edsger Dijkstra

- Computer science & engineering is about many things:
  - Automating the organization and analysis of information
    - Examples: "Degrees of Facebook separation" / "driving directions"
  - Problem solving
  - Logical thinking
  - Being very precise: software does *exactly* what you say
- With many subspecialties:
  - Graphics, robotics, computational biology, ubiquitous computing, artificial intelligence, programming languages, large-scale data processing, human-computer interaction, much, much more

Programming

- Writing programs is a big part of what CSE people do
  - Also design them, evaluate them, improve them, etc.
  - Writing programs often-but-not-always the fun part
  - Involves taking the "basic idea" and doing it *exactly*
    - A really useful skill for everyone
- CSE142 is an introduction to programming
  - How to write and debug short programs
  - Key features and techniques used in programming
  - *Specifying* and *automating* tasks

Java

- We need a *language* for writing down programs
  - English is far too imprecise
- Java is one such *programming language*
  - There are 1000s of others, but sticking with one makes sense for an introductory course
  - Python is another (optional sessions)
- Java is an industrial-strength language
  - The real world writes 1,000,000-line programs with it
  - Rich libraries (e.g., graphics) and tools (e.g., JGrasp)
  - Advantages and disadvantages for learning
    - Learning to steer on a fighter jet instead of a tricycle

Four layers

You are going to learn a lot of things

1. Fundamental ideas bigger than CSE
   Example: Break a problem into logical pieces
2. Fundamental ideas of CSE
   Example: Repeat execution with slightly different data
3. Fundamental ideas of programming
   Example: Using a *loop* to do repeated execution
4. Precise details of Java you just have to get right
   Example: Where to use (and) versus { and }

All 4 layers matter.
The layers can help organize the material.

Welcome

Today:

- Course mechanics
- A little about computer science & engineering
  - And how this course relates
- *Java* programs that print text

Basic Java programs with `println` statements

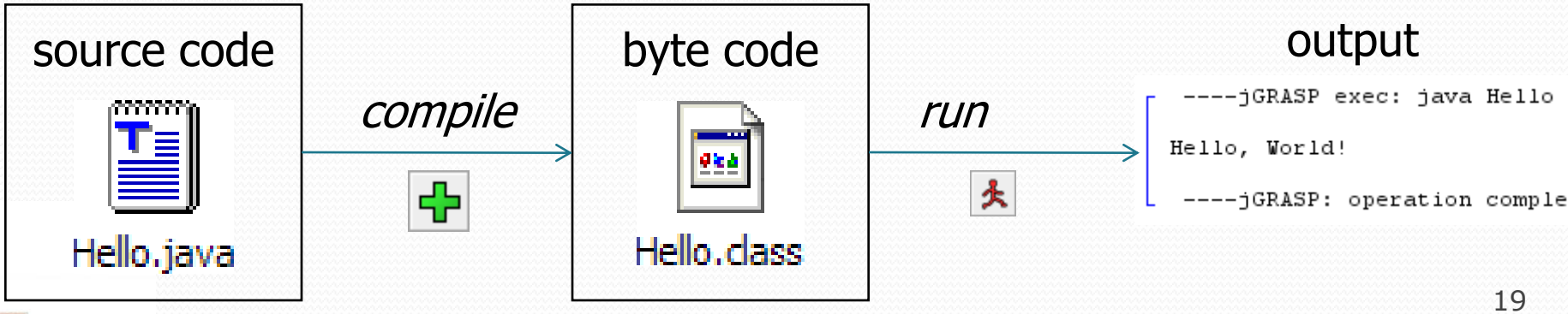
reading: 1.2 - 1.3

self-check: #5-14

exercises: #1-4

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

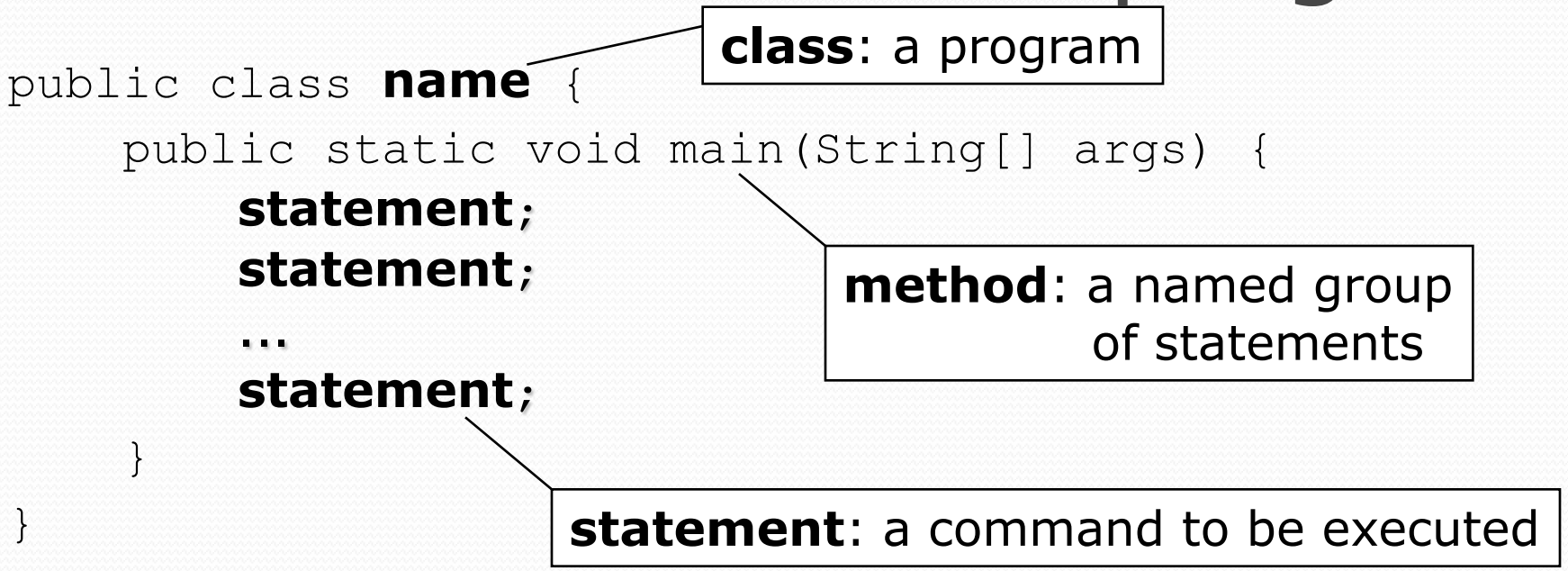
```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

- 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 Lecture1 {
```

  - Naming convention: capitalize each word (e.g. `MyClassName`)
  - Your program's file must match exactly (`Lecture1.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`
  - Cannot be "keywords" (see the text)

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 \"");
```

Some concepts

- Writing your first Java program involves a lot of details
  - **`public static void main (String [] args) {…}`**
  - where to put semicolons
  - Escape sequences
  - etc.

- A couple key concepts
  - *Execute a sequence* of commands
  - *Represent* strings using particular rules
    - Important because you might want characters literally that mean something else in Java
    - This literal vs. not distinction is common in programming