Chapter 1
Review

• What is computer science?
• Last time, we designed a simple *algorithm*
  • What is an algorithm?
• What did we learn in doing so?
OK. Let’s talk about how computers work. What are the main parts of a computer?
Every computer has a *central processing unit* (CPU) that executes sequences of simple instructions (aka programs).
The CPU can execute programs that are contained in RAM.
When a program is not executing, its instructions will be stored on disk.
What is a program?
A program is a sequence of instructions to accomplish a specific task (or tasks).
Programming is providing sequences of (very) basic instructions to accomplish tasks.
Tasks

• Many programs do only one task, like “edit image files”
  • This task is as complicated as you want to make it, though!
• Some programs can do more than one task, like Internet Explorer
How do we provide instructions to the computer?
We can write programs in a *programming language*, like Java, C, Basic, or assembly.
Programs in *high-level languages* are easy for humans to read but can’t run directly on a CPU.
Programs in *machine language* can execute directly on a CPU (but are very hard for humans to read).
Solution: translate from high-level to machine language.
How do we do this?

answer: not by hand!
(if we can help it)
The Java compiler translates from Java code into Java virtual machine code.
The Java compiler translates from Java code into Java virtual machine code.

FROM A HIGH-LEVEL LANGUAGE...

...TO A LOW-LEVEL LANGUAGE!
Great! So we can write our program in Java and have the Java compiler translate it into machine code.

(The Java Virtual Machine is actually a program that acts like a CPU.)
What happens if we make a mistake?
What happens if we make a mistake?

(What kinds of mistakes could we make?)
Two kinds of errors

• “Syntax errors,” or compile-time errors, prevent the Java compiler from translating your program
  • example: omitting a semicolon

• “Logic errors,” or run-time errors, cause your program to do the wrong thing
  • example: giving the wrong directions
Compile-time errors prevent the compiler from “making sense of” your program (and translating it).
“Compile-time errors”

• If your run all of your wordstogether no one will understand you.

• Ungrammatical sentences to understand hard can be.

• Write complete sentences, which are
A program that makes sense but does the wrong thing contains run-time errors.
Examples

• Can be subtle:
  • “Heat water to 100 degrees C before adding yeast”

• Or drastic:
  • use your imagination here

• Also consider programs that just don’t meet requirements
We can fix errors by applying the *edit-compile-test cycle*. 
Edit-compile-test

- **EDIT**: we write our program in a text editor
- **COMPILE**: we try and translate our program to machine code with a compiler
  - if there are compile-time errors, edit again!
- **TEST**: we run our program to determine whether or not it behaves as expected
  - if there are run-time errors, edit again!
You can do every step of the edit-compile-test cycle within the Eclipse program!
Programs you’ll use while programming

• A text editor to edit program text
• A compiler to translate from Java code to Java virtual machine code
• A virtual machine to run the program
• (The Eclipse IDE combines all three)
public class Hello {
    public static void main(String [] args) {
        String greeting =
            "Hello, world!";
        String yellingGreeting =
            greeting.toUpperCase();

        System.out.println(greeting);
        System.out.println(yellingGreeting);
    }
}

What do you suppose this program does?
Some review questions
What are the basic parts of a computer?
What is a program?
What programs would you use while programming?
What kinds of errors can occur in our programs?