lecture_1:

Introduction to Java

Components of acomputer

- Central Processing Unit (CPU) - receives instructions from memory and executes them (Intel
  Core i9)
- Main Memory - stores information while the computer is on (RAM) Storage -
- stores information while the computer is on or off (1TB SSD)
- Input devices - pass information from user to computer (keyboard, touchpad) Output
- devices - pass information from computer to user (screen, printers, speakers)
  Communication devices - pass information among computing devices (modem, network
- interface card)

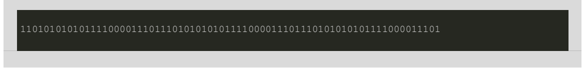
Bits and bytes

- 1 byte = 8 bits (stores a number between -128 and 127)
- 1 kilobyte = 1024 bytes (20 KB = 1 page word processing document)
- 1 megabyte = 1024 kilobytes (1 MB = 50 pages of documents)
- 1 gigabyte = 1024 megabytes (8 GB = 1 movie)
- 1 terabyte = 1024 gigabytes

Types of programming languages

Machine language

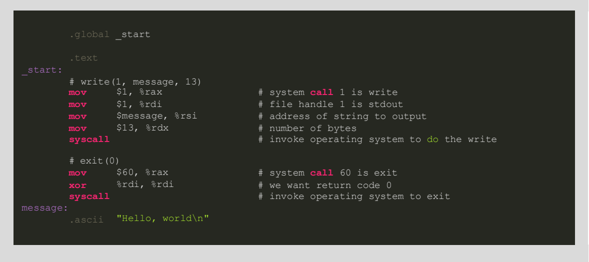
All instructions are primitive and specific to different types of computers. These instructions are in binary form. These languages are very hard for humans to read and interpret. Each processing architecture has its own machine code. For example, ARM (mobile devices) is different than Intel x64 (Macs / PCs).



```
110101010101111000011101110101010101111000011101110101010101111000011101
```

Assembly language

Assembly makes machine language easier to understand, though it is still not very human-readable. Assembly languages can be directly converted into machine language via an assembler. Because it is compiled into machine language, assembly is different for each processor architecture.



```
        .global _start

        .text
_start:
        # write(1, message, 13)
        mov     $1, %rax                # system call 1 is write
        mov     $1, %rdi                # file handle 1 is stdout
        mov     $message, %rsi          # address of string to output
        mov     $13, %rdx               # number of bytes
        syscall                         # invoke operating system to do the write

        # exit(0)
        mov     $60, %rax               # system call 60 is exit
        xor     %rdi, %rdi              # we want return code 0
        syscall                         # invoke operating system to exit
message:
        .ascii  "Hello, world\n"
```

High level languages

These languages are the ones most typically used by developers. Some examples include Java, C, C++, Python, Haskell, Go, Rust, R, etc. High level languages are easier to read and ohen have cross-platform support. For example, applications written in Java can typically run on Windows, OS X, and Linux without modification.

There are some good examples of high level langauges and their uses on pg. 8 of Liang textbook.

```
System.out.print("Hello, world!")
```

Compilers and interpreters

- An interpreter reads one statement from a source code file, executes it, and moves to the next line.
- A compiler translates the entire file into machine language before executing a single instruction. The user executes that machine language file.
- Java is a combination of compilation and interpretation. It compiles a Java source file into Java Bytecode, which is interpreted by the host machine.

Operating systems

An OS manages and controls a computer system. Windows, Mac OS, Linux, and Unix are examples. Operating systems control and monitor system activities, allocate and assign system resources, and schedule operations. The operating system communicates with the system hardware directly.

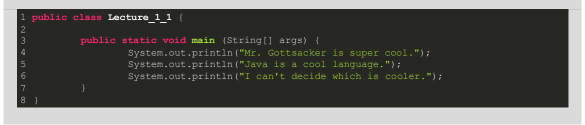
Multi- definitions

- Multiprogramming: programs operating at the same time, sharing the same CPU
- Multithreading: a single program executing multiple tasks at the same time
- Multiprocessing: similar to multithreading; uses separate processors for each program

Java

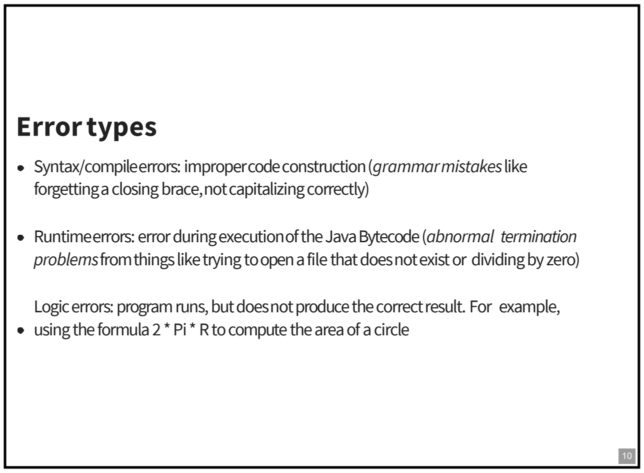
- Invented by Sun Microsystems in 1995, acquired by Oracle in 2010 Object-
- oriented
- Popular language for modem applications and sohware companies -- PowerSchool, Android apps, Twitter, Square, Hadoop
- Security problems stemming from embedding Java applets in webpages... but no
    - one does that anymore.
- JDK: Java Development Kit (compiling Java source code) JRE:
- Java Runtime Environment (running Java .class files)

Simple Java program



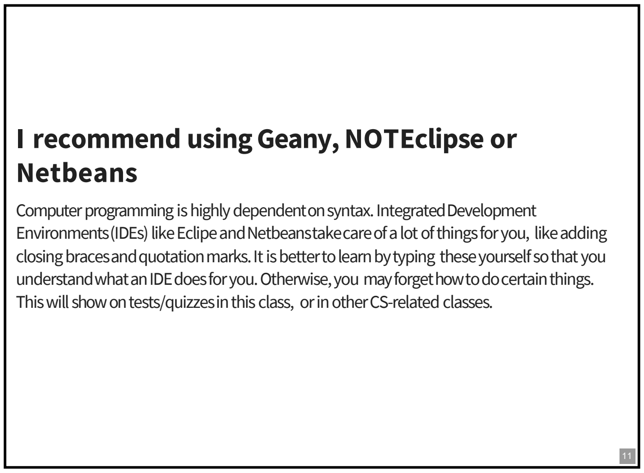
```java
public class Lecture_1_1 {

        public static void main (String[] args) {
                System.out.println("Mr. Gottsacker is super cool.");
                System.out.println("Java is a cool language.");
                System.out.println("I can't decide which is cooler.");
        }
}
```



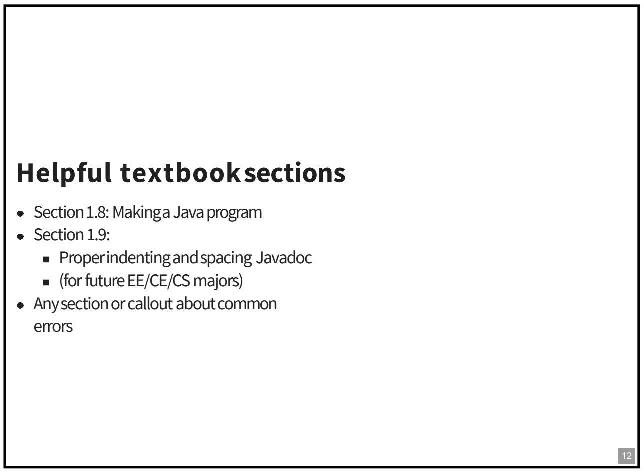
Error types

- Syntax/compile errors: improper code construction (*grammar mistakes* like forgetting a closing brace, not capitalizing correctly)
- Runtime errors: error during execution of the Java Bytecode (*abnormal termination problems* from things like trying to open a file that does not exist or dividing by zero)
- Logic errors: program runs, but does not produce the correct result. For example, using the formula 2 * Pi * R to compute the area of a circle



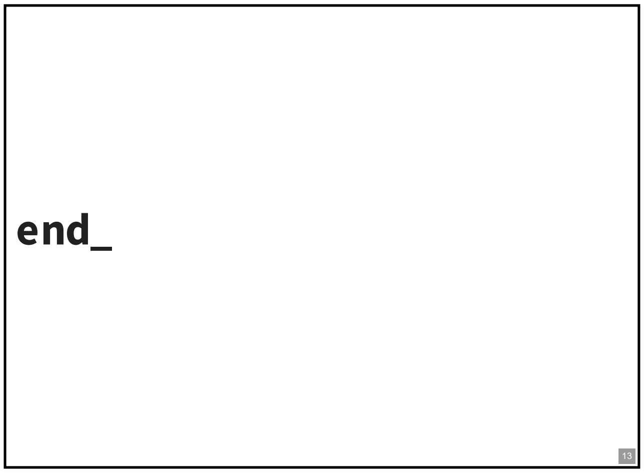
I recommend using Geany, NOTEclipse or Netbeans

Computer programming is highly dependent on syntax. Integrated Development Environments (IDEs) like Eclipe and Netbeans take care of a lot of things for you, like adding closing braces and quotation marks. It is better to learn by typing these yourself so that you understand what an IDE does for you. Otherwise, you may forget how to do certain things. This will show on tests/quizzes in this class, or in other CS-related classes.



Helpful textbook sections

- Section 1.8: Making a Java program
- Section 1.9:
  - Proper indenting and spacing Javadoc
  - (for future EE/CE/CS majors)
- Any section or callout about common errors



end_

13