Intro to Java and OOP in Java

CSC207 -- Week 2

Course Overview

Tools (Weeks 1-4)

- Java
- Version Control
- Software Tools

Design (Weeks 5-8)

- Clean Architecture
- SOLID
- Design Patterns

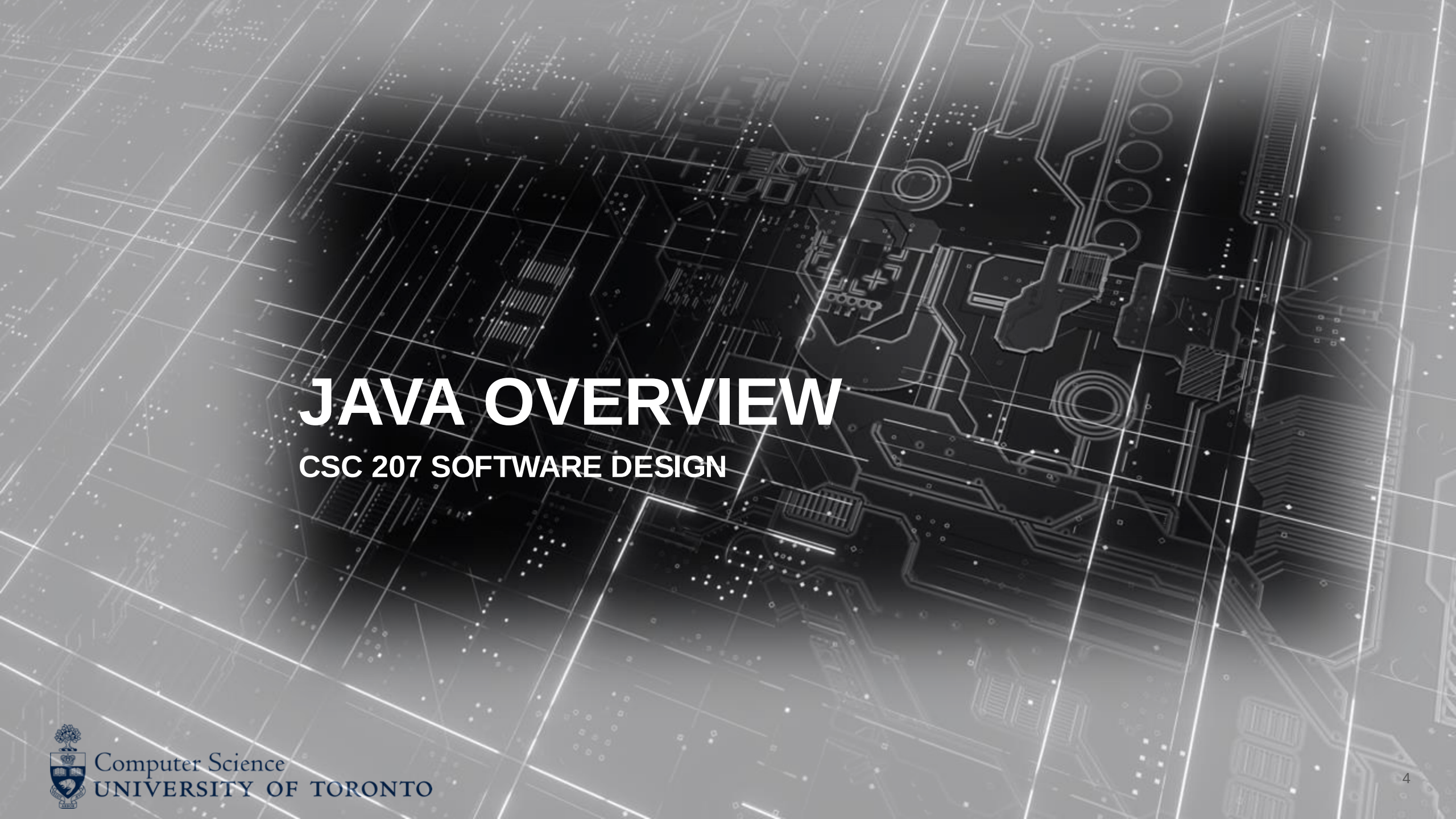
Professional Topics (Weeks 9-12)

- Ethics
- Internships
- GenAI

- Last week we introduced git, installed software, and motivated the course
- This week, we will talk about
  - running Java programs
  - OOP in Java
    - Parts of a class
    - Inheritance
    - Constructors

Questions to be answered this week...

- How do we compile and run Java programs? What is the JVM?
- What is the structure of a Java class? (declaration, variables, …)
- Since the constructor of a parent class is not inherited by a subclass, how does the parent part of a subclass instance get constructed?
- What is the difference between overriding, overloading, and inheriting a method? Write a definition for each.
- What are the different ways that a subclass instance can access private information in the parent class?
- What is casting? When do we use it? Why should we avoid using it too much?



JAVA OVERVIEW

CSC 207 SOFTWARE DESIGN

LEARNING OUTCOMES

- Be familiar with what resources are available to help you learn the fundamentals of Java this term.
- Understand how Java works at a high level.
- Be aware of what concepts you will be learning through the readings over the next couple of weeks.

REFERENCE MATERIAL ON JAVA

- The "course notes" and practice Java quizzes on Quercus:
  - https://github.com/CSC207-UofT/207-course-notes/blob/master/01-introduction-to-java.md
  - https://q.utoronto.ca/courses/353377/pages/java-practice-quizzes?module_item_id=6076633
- The official java tutorials are also a great place to learn:
  - http://docs.oracle.com/javase/tutorial/java/TOC.html
- An optional mini e-textbook:
  - https://runestone.academy/ns/books/published/java4python/index.html
- Please share any other resources you find useful on Piazza for others to benefit from.
- Ask on Piazza if you have questions or visit office hours. Everyone is here to help.

VERSION OF JAVA IN THIS COURSE

- In most materials we focus on the core language features and point to the official Java tutorials, which were written for Java 8.
- There are some nice language features in later versions of Java, but we largely won't be emphasizing them in this course. E.g., the Java 10 `var` keyword.
- We have suggested you work with Java 17, but once you start working on the project, your team can agree on which version you want to use.

RUNNING PROGRAMS

What is a program?

What does it mean to “run” a program?

To run a program, it must be translated from its high-level programming language to a low-level machine language whose instructions can be executed.

Roughly, there are two flavours of translation:

- **Interpretation**
- **Compilation**

INTERPRETED VS. COMPILED

Interpreted (e.g., Python)

- Python is a program (an interpreter) that translates and executes one statement of a program at a time.

Compiled (e.g., C)

- A C compiler translates an entire C program to an executable file. The translated program contains machine code that runs natively on an operating system.

Hybrid (e.g., Java)

- Java programs are compiled to bytecode (which is a lot like assembly code) using the Java compiler. This is an intermediate representation of the program.
- The Java Virtual Machine (JVM) is an executable that runs this intermediate bytecode, translating and executing one statement of the bytecode program at a time. As it runs, it does fancy things like look for optimizations to make.

COMPILING JAVA

- You need to compile, then run (as described above).
- If using the command line, you need to do this *manually*. For example:

First, compile using “javac”:

```
dianeh@laptop$ javac HelloWorld.java
```

This produces file “HelloWord.class”:

```
dianeh@laptop$ ls
HelloWorld.class  HelloWorld.java
```

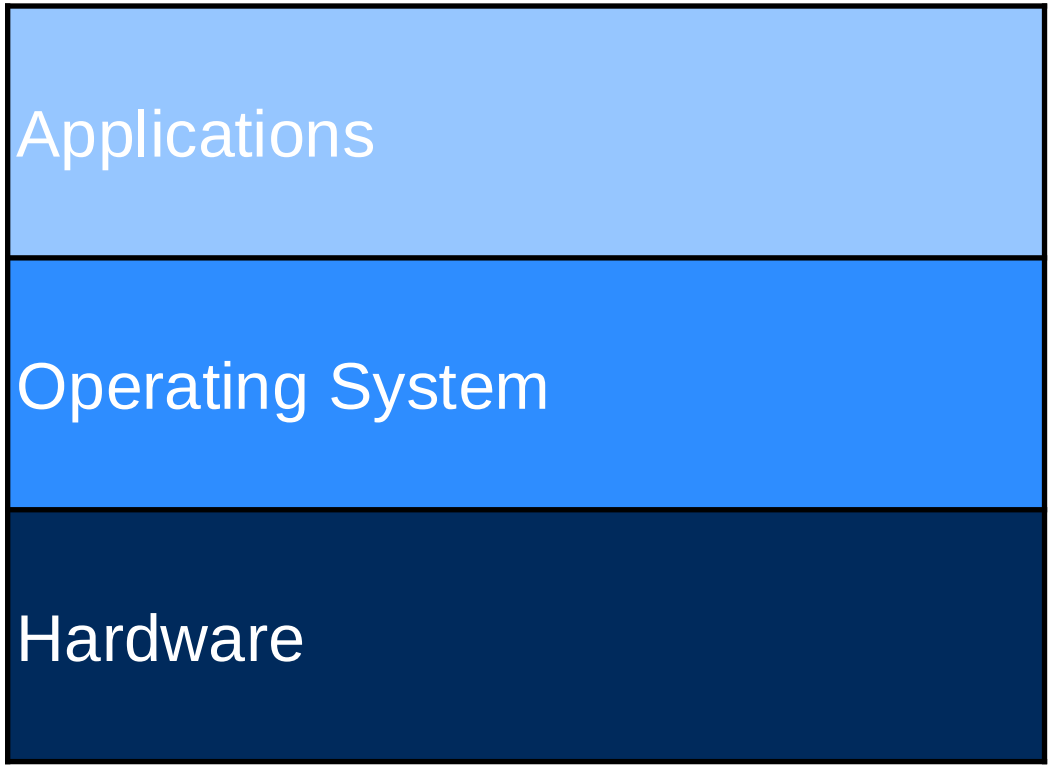
Now, run the program using “java”:

```
dianeh@laptop$ java HelloWorld
Hello world!
```

- **Modern IDEs (like IntelliJ) do this for you with the help of a build system (e.g. maven or gradle)**
  - Build systems take care of the build process, so that you can focus on the code!
- But you should have some idea of what’s happening under the hood!
  - If you go on to take CSC209, you will learn C and get much more practice with compiling code.

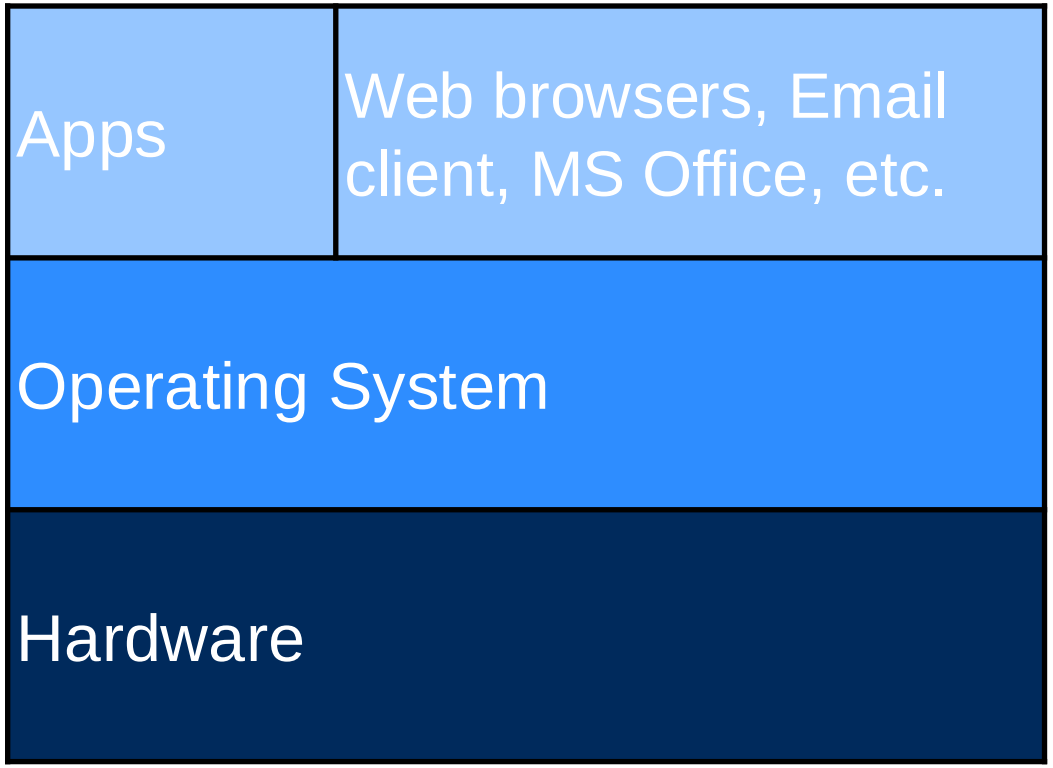
COMPUTER ARCHITECTURE

- 108/148 and 110/111: we focused on applications.
- The operating system (OS) manages the various running applications and helps them interact with the hardware (**CSC209H**, CSC369H).
- The OS works directly with the hardware (**CSC258H**, CSC369H).



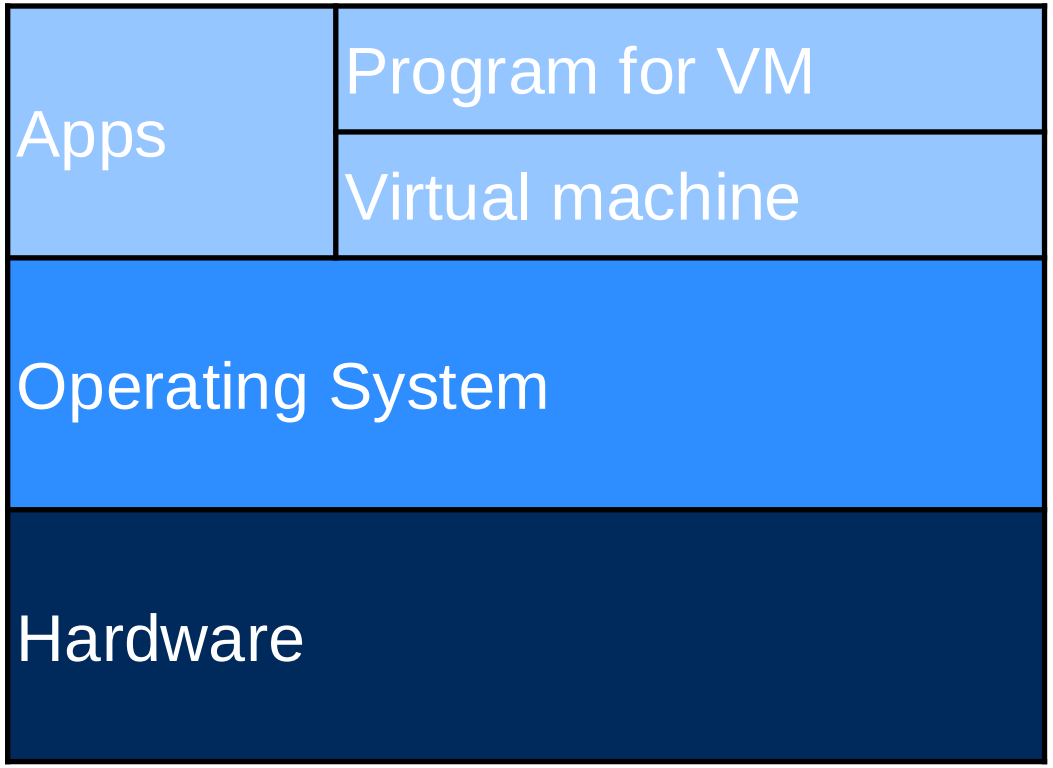
COMPILED APPLICATIONS

- Programs written in languages like C, Objective C, C++, Pascal, and Fortran are **compiled** into OS-specific applications.
- Compilation involves turning human-readable programs into OS-specific machine-readable code (CSC258H, CSC369H, CSC488H).



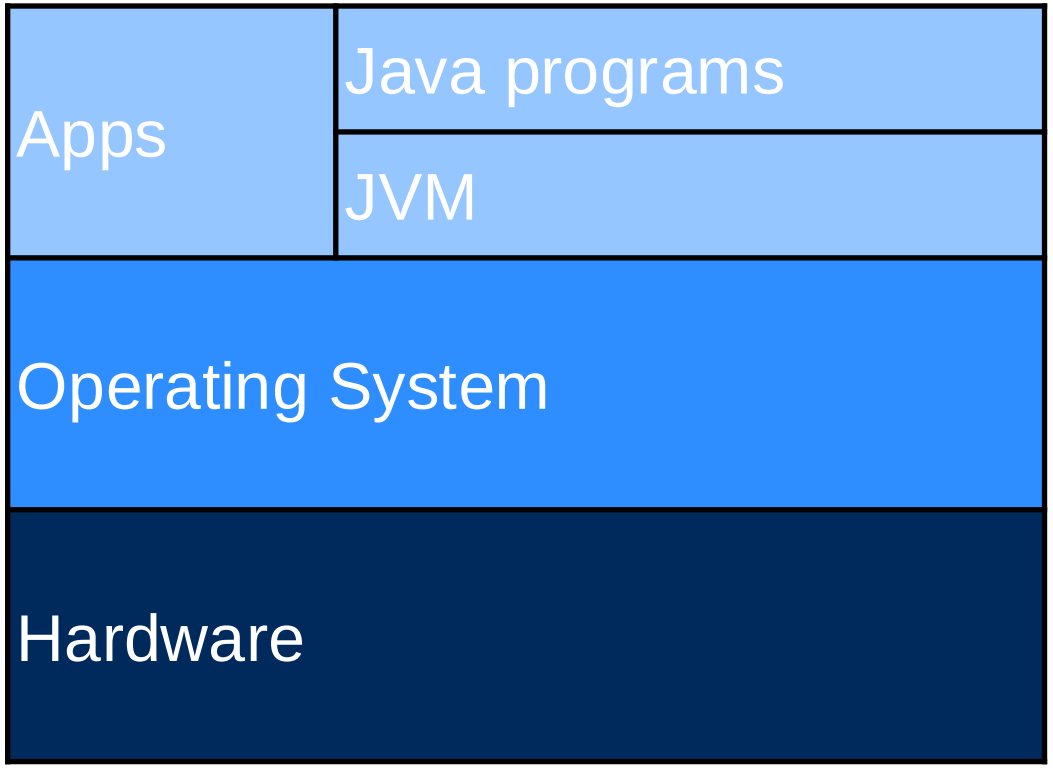
VIRTUAL MACHINE ARCHITECTURE

- **Virtual machine (VM)**: an application that pretends to be a computer.
- For example, each of Java, Python, and Scheme have their own VM.
- The VM application is written for each OS so that programs are portable across operating systems.



JAVA ARCHITECTURE

- **JVM**: an application that is something like an operating system
- Java programs are compiled to an intermediate state called **byte code**: machine code for the JVM
- The same compiled Java program can run in any JVM on any OS!



- JVMs optimize the byte code as it runs (CSC324H, CSC488H) and can even be as fast as code written in C!

NOTABLE ASPECTS OF JAVA YOU'LL LEARN THE NEXT COUPLE WEEKS

- types
- primitives
- casting
- generics
- exceptions
- Interfaces
- overloading
- memory model
- junit
- javadoc
- constructors
- access modifiers
- packages