



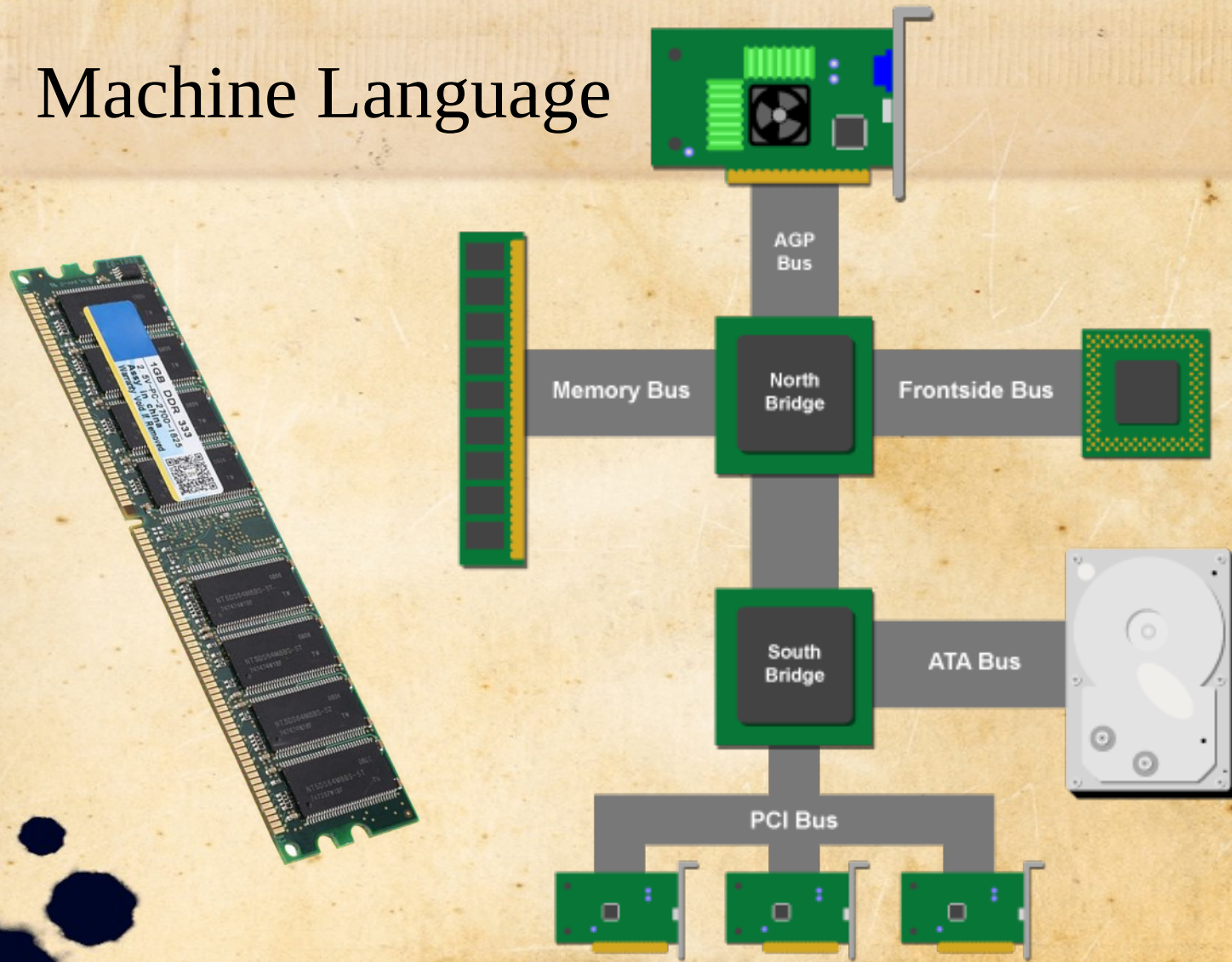
Introduction

Programming Languages

Lecture Contents

- Languages
 - Machine Language (1st Generation)
 - Assembly Language (2nd Generation)
 - High-Level Programming Languages (3rd Generation)
 - Subroutine (FORTRAN, BASIC)
 - Procedural / Functional (Pascal, C)
 - Object Oriented (C++, Java)
- Compilers and Interpreters
- Java

Machine Language



Machine Language (1st Generation)

- This is the language the computer understands
 - For a hypothetical computer, instructions could be

				Machine Code
0000	Stop			
0001	Add	3+4	→	0001 0011 0100
0010	Subtract	7-5	→	0010 0111 0101
0011	Multiply	2x3	→	0011 0010 0011
0100	Divide	6÷2	→	0100 0110 0010
0101	Square	3x3	→	0101 0000 0011

Assembly Language (2nd Generation)

- It is more human-readable, but corresponds directly to the machine code (one assembly instruction → one machine language instruction)
 - It is specific to each type of computer chip

Assembly Code		Machine Code
add 3,4	→	0001 0011 0100
sub 7,5	→	0010 0111 0101
mul 2,3	→	0011 0010 0011
div 6,2	→	0100 0110 0010
sqr 3	→	0101 0000 0011
stop		0000 0000 0000

- Translated into machine code by an *assembler*.

High-Level Languages (3rd Generation)

- Much more human-readable
- Languages are becoming more abstract with time
 - less abstract languages such as BASIC, C
 - more abstract languages such as Java and Python
- High-Level languages are translated into ***machine code*** by either a ***compiler*** or an ***interpreter*** (or a combination of them).
- One ***statement*** in a high-level language may become many ***machine code*** instructions.

High Level Languages

- BASIC

```
10 FOR I = 1 TO 10
20 GOSUB 100
30 NEXT I
40 END
100 PRINT "hello hackaday!"
110 RETURN
```

```
main:  let b2 = 15           ; set b2 value
       gosub flsh          ; call sub-procedure
       let b2 = 5          ; set b2 value
       gosub flsh          ; call sub-procedure
       end                 ; stop accidentally falling into sub

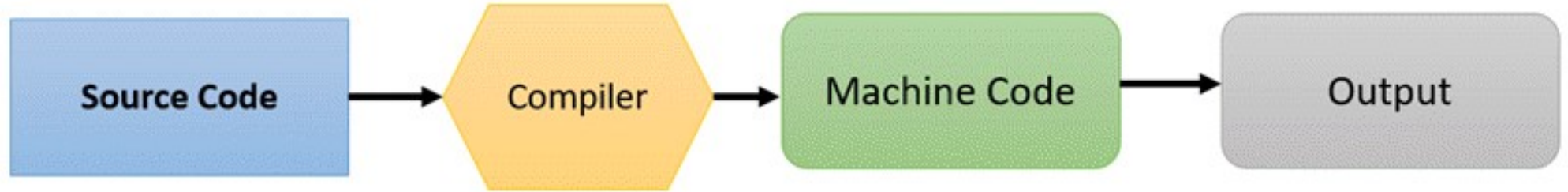
flsh:  for b0 = 1 to b2      ; define loop for b2 times
       high B.1             ; switch on output 1
       pause 500            ; wait 0.5 seconds
       low B.1              ; switch off output 1
       pause 500            ; wait 0.5 seconds
       next b0              ; end of loop
       return               ; return from sub-procedure
```


Interpreters and Compilers

- Computers only understand ***machine code***
 - Specific to each computer chip family (X86, ARM, RISC-V)
- ***Assembler***: assembly language → machine code
- ***Interpreter***:
 - Line by line, executing instructions at the same time.
- ***Compiler***: high-level language → machine code (“object code”)
 - Entire program to generate an executable file that can be run again and again.

Interpreters and Compilers

How Compiler Works

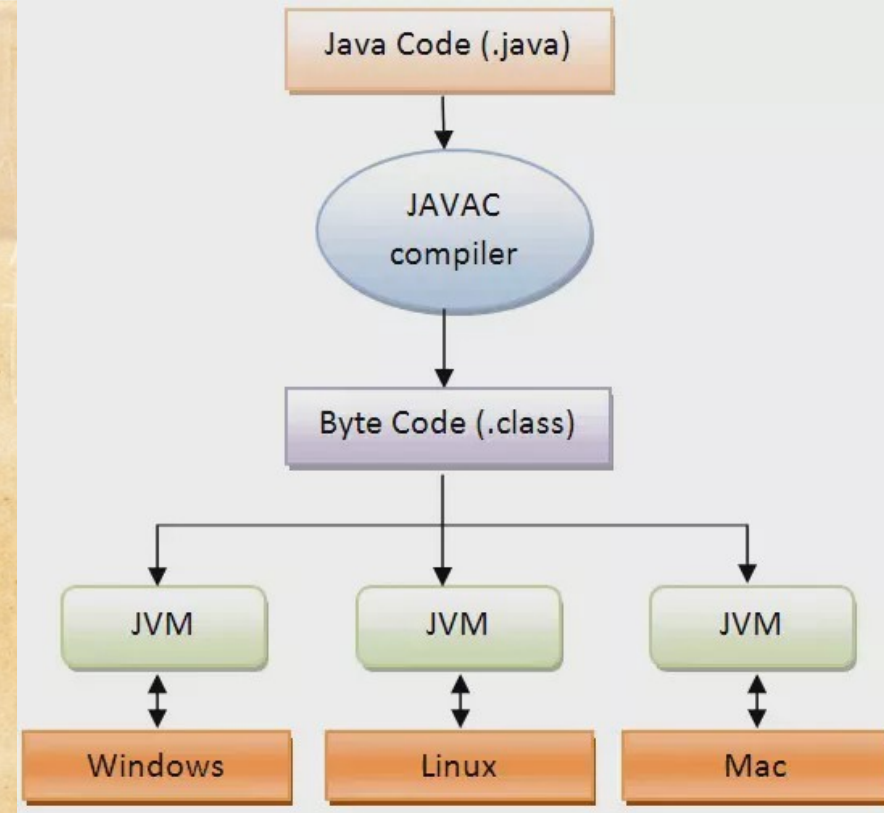


How Interpreter Works

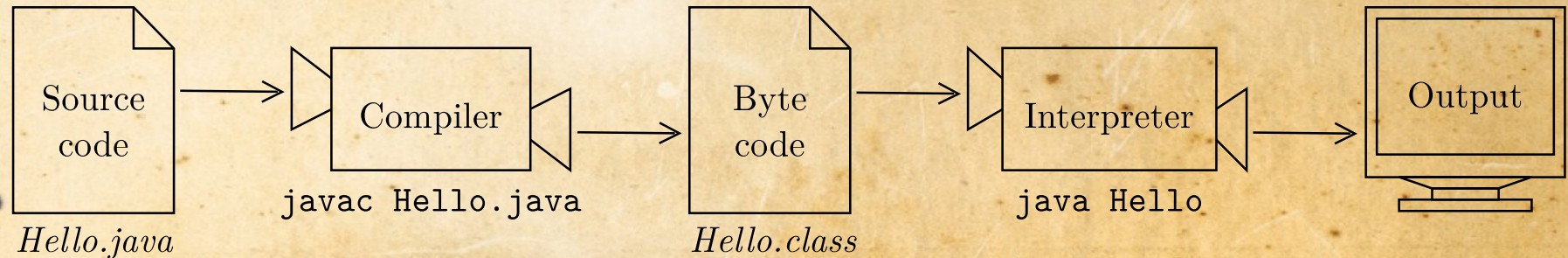
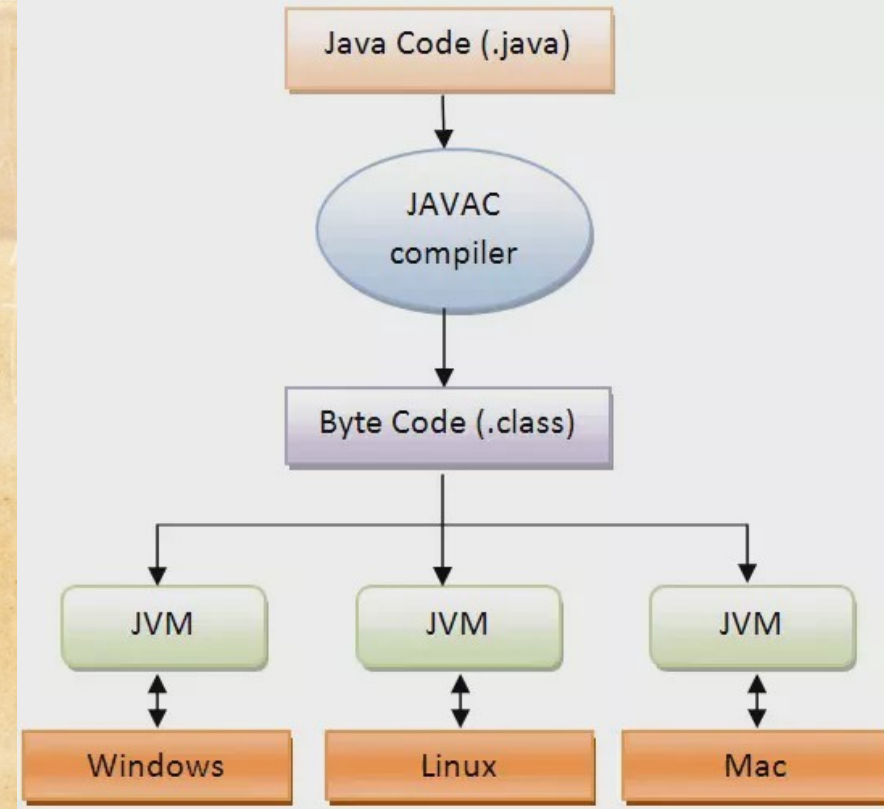


Java

- Uses both a **compiler** and an **interpreter**.
 - **javac** compiles Java *source code* into Java *bytecode*.
 - Java *bytecode* is *machine code* for a **Java Virtual Machine (JVM)**.
 - The command **java** runs the JVM.

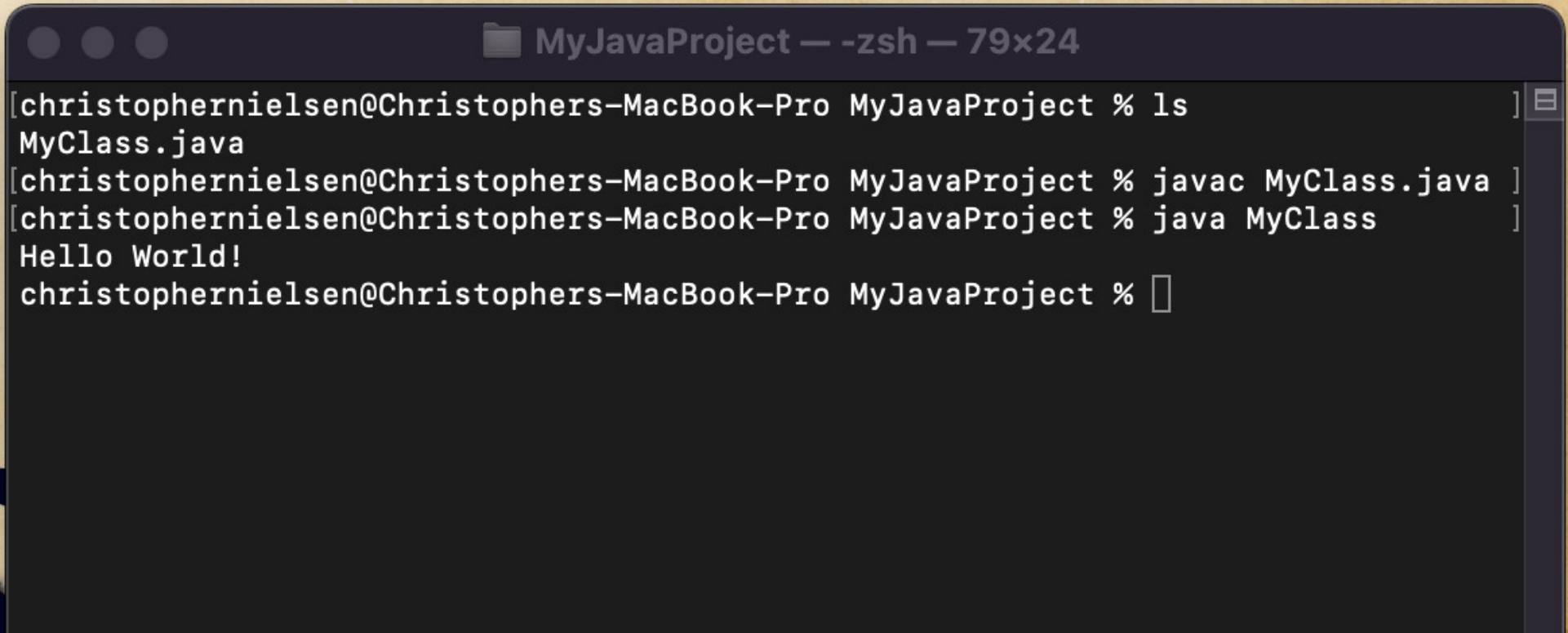


- Uses both a **compiler** and an **interpreter**.
 - **javac** compiles Java *source code* into Java *bytecode*.
 - Java *bytecode* is *machine code* for a **Java Virtual Machine (JVM)**.
 - The command **java** runs the JVM.



Java

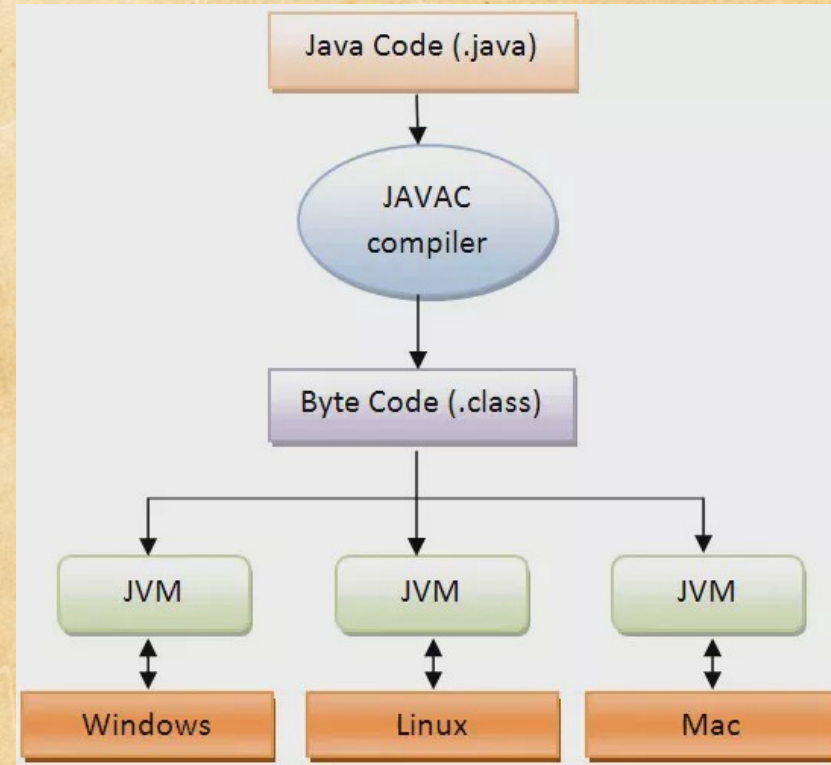
- Uses both a *compiler* and an *interpreter*.

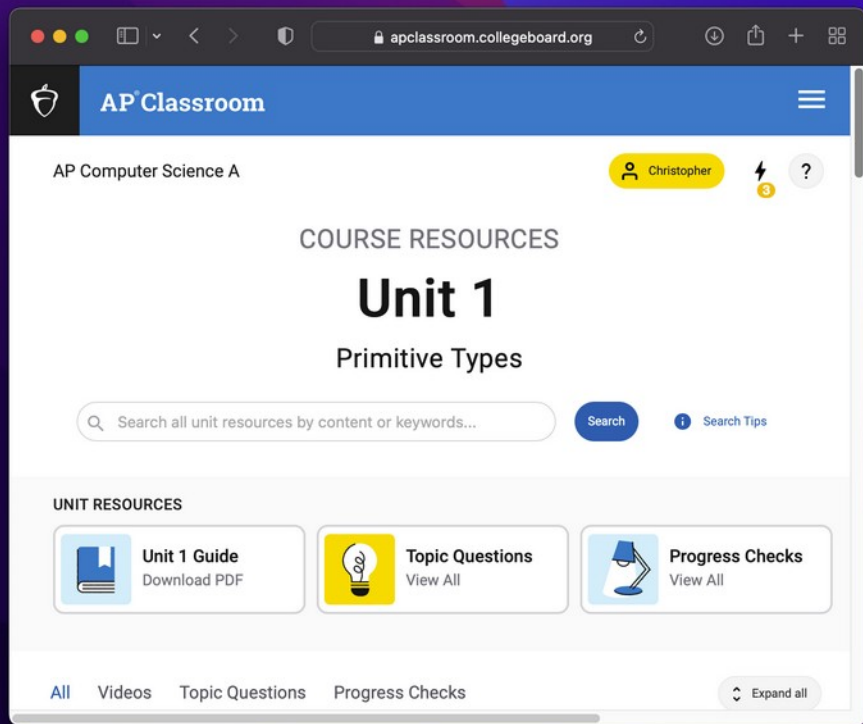
A terminal window titled "MyJavaProject — -zsh — 79x24" is shown. The window has a dark background and a light-colored text. The text inside the terminal shows a series of commands and their outputs. The first command is "ls", which lists "MyClass.java". The second command is "javac MyClass.java", which compiles the file. The third command is "java MyClass", which runs the program and outputs "Hello World!". The prompt "christophernielsen@Christophers-MacBook-Pro MyJavaProject %" is visible at the end of each line of input.

```
MyJavaProject — -zsh — 79x24
[christophernielsen@Christophers-MacBook-Pro MyJavaProject % ls
MyClass.java
[christophernielsen@Christophers-MacBook-Pro MyJavaProject % javac MyClass.java
[christophernielsen@Christophers-MacBook-Pro MyJavaProject % java MyClass
Hello World!
christophernielsen@Christophers-MacBook-Pro MyJavaProject % ]
```

Java

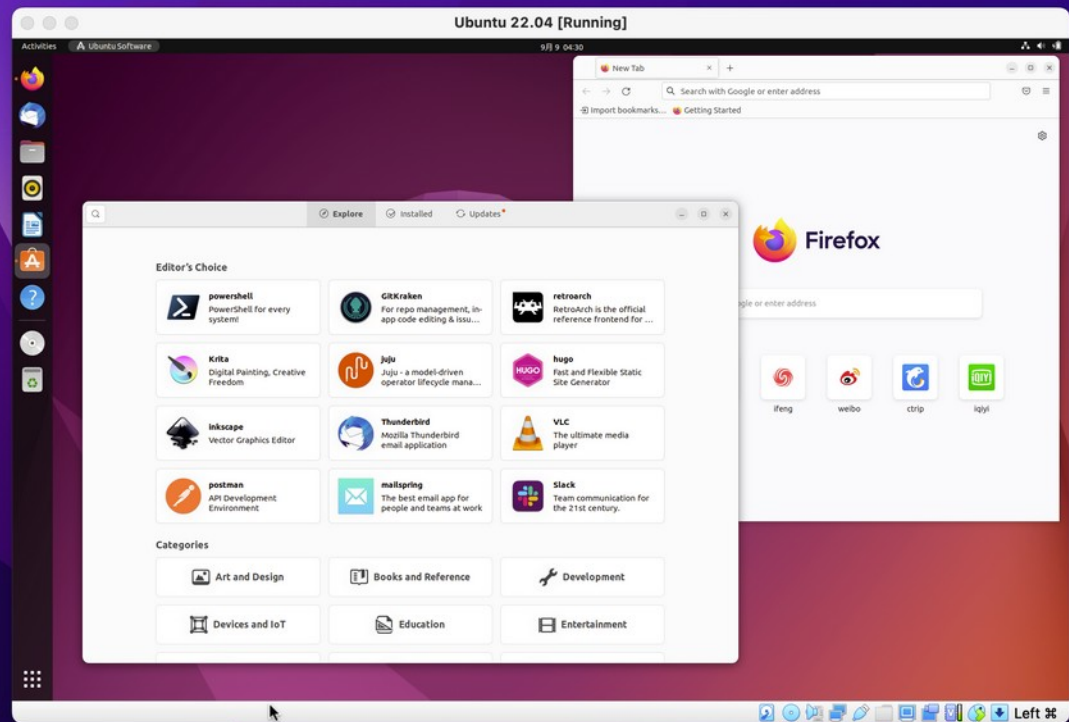
- The big benefit of programming in Java:
 - A compiled java program can run without modification on any machine that has a JVM
 - Operating systems such as Windows, MacOS, Linux, Android, ...
 - Hardware architectures such as X86, ARM, PowerPC, MIPS, RISC-V, ...
- It should be faster than other interpreted languages since *byte code* is very low level and requires little translation work





The screenshot shows the AP Classroom website interface. At the top, the URL is `apclassroom.collegeboard.org`. The header includes the AP Classroom logo and a user profile for "Christopher". The main content area is titled "COURSE RESOURCES" and "Unit 1 Primitive Types". Below this is a search bar with the text "Search all unit resources by content or keywords...". Under the "UNIT RESOURCES" section, there are three cards: "Unit 1 Guide Download PDF", "Topic Questions View All", and "Progress Checks View All". At the bottom, there are tabs for "All", "Videos", "Topic Questions", and "Progress Checks", along with an "Expand all" button.

A virtual machine running
the Linux operating system



The screenshot shows a virtual machine window titled "Ubuntu 22.04 [Running]". The desktop environment is Ubuntu 22.04, featuring a dark theme and a sidebar with application icons. A "New Tab" window is open in the background. In the foreground, the "Ubuntu Software" application is open, displaying a grid of installed and available applications. The "Editor's Choice" section includes applications like powershell, GitKraken, retroarch, Krita, Jujv, hugo, Inkscape, Thunderbird, VLC, postman, mailspring, and slack. Below this, there are categories such as Art and Design, Books and Reference, Development, Devices and IoT, Education, and Entertainment. The bottom of the window shows a dock with various system icons and a "Left" button.



Introduction

Programming Languages