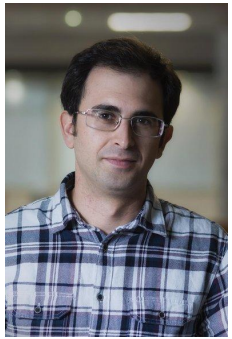
Java Essentials

Instructor: Vahe Pezeshkian

TA: Margarita Zargaryan

sourcemind



Vahe Pezeshkian

Amirkabir University of Technology (Tehran Polytechnic)

BSc IT Engineering

AUA

MSc Computer & Information Science

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

EngD Software Technology

Adobe

Sr. Software Engineer

vaheh.pezeshkian@gmail.com

https://www.linkedin.com/in/vpezeshkian/

sourcemind

Margarita Zargaryan

AUA

BS in Computer and Information Science

AMOTEK TECHNOLOGIES

Java Software Engineer

sourcemind

Hello Java!

- Introduction to Java and JVM
- Writing Java programs
- Java program constructs
- Java stack, heap, and references
- Java class library

OOP Basics

- Introduction to OOP
- Java OOP by practice

Java Core Powers

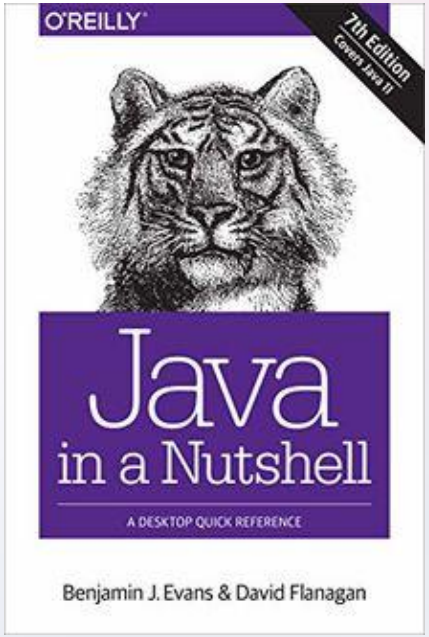
- Exceptions
- JUnit
- Enums
- Generics
- Java collections
- Streams
- Java I/O

Design Patterns

- Creational
- Behavioral
- Structural
- Practice with Java Core & Patterns

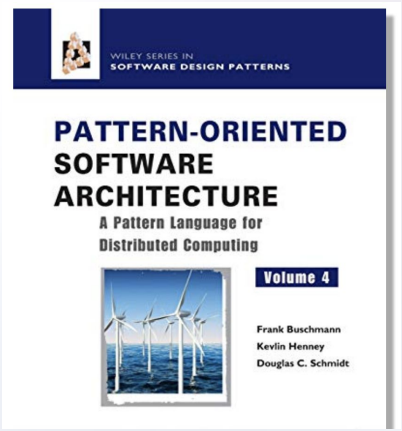
Materials: Java

- Slides
- Exercises
- Homework
- Additional Reading: Java in a Nutshell (7th ed.)



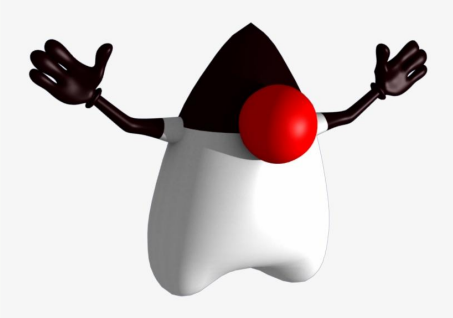
Materials: Design Patterns

- Refactoring Guru: Design Patterns
- Head First Design Patterns
- Design Patterns (Gang of four)
- Pattern-oriented Software Architecture vol. IV



sourcemind

Session 1: Hello Java!



sourcemind

The Java world

- The Java language
- The Java Virtual Machine (JVM)
- The Java ecosystem

1. The Java language

- 1996: Java 1.0 *Introduced JVM and Java language*
  Syntax similar to C/C++ on purpose
  Designed with backwards-compatibility in mind
- 1998: Java 1.2 *Collections API*
- 2004: Java 5 *Enums, generics*
- 2014: Java 8 *Lambda expressions, Streams API*
- ...
- 2018: Java 11 (LTS)
- ...
- **2021: Java 17 (LTS)**

JVM Versions

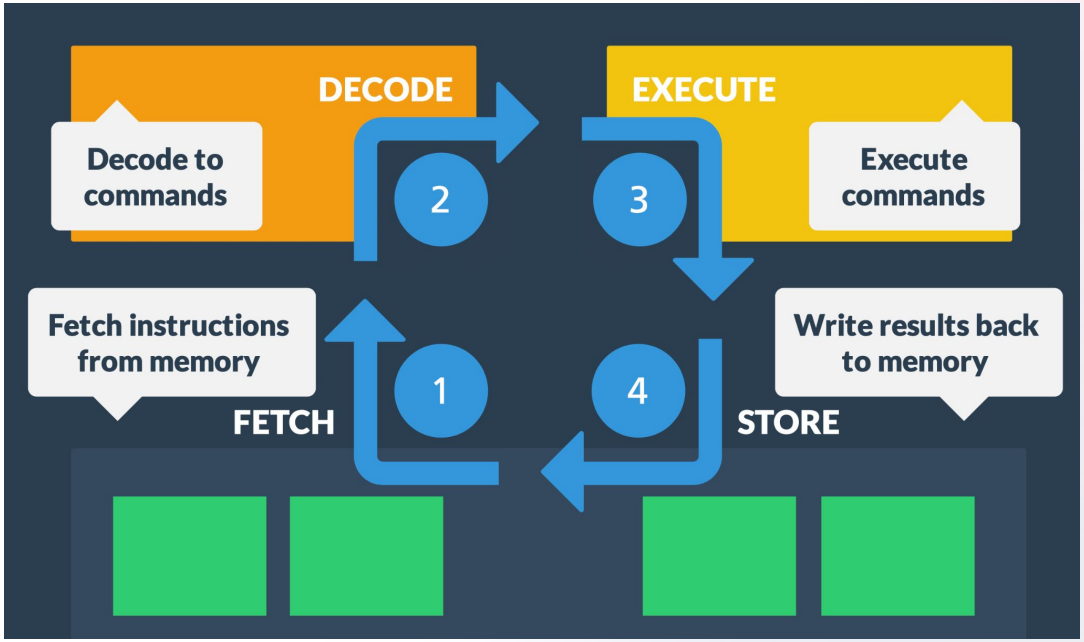
- New version: Every 6 month
- Long-Term Support (LTS) every 3 years
- Go to:

https://openjdk.org/projects/jdk/17/

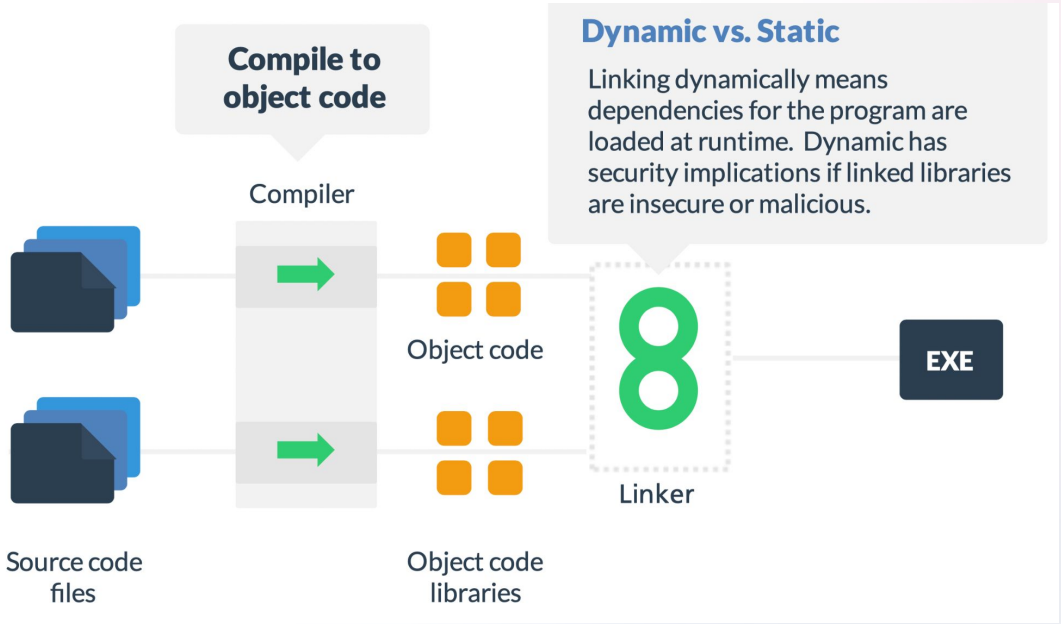
Status: What phase is the JDK version in?
Features: What to expect from that version? (e.g. JDK 11 ZGC exp. → JDK 15 final)
Schedule: Phase 1, 2, ..., GA

How programs execute?

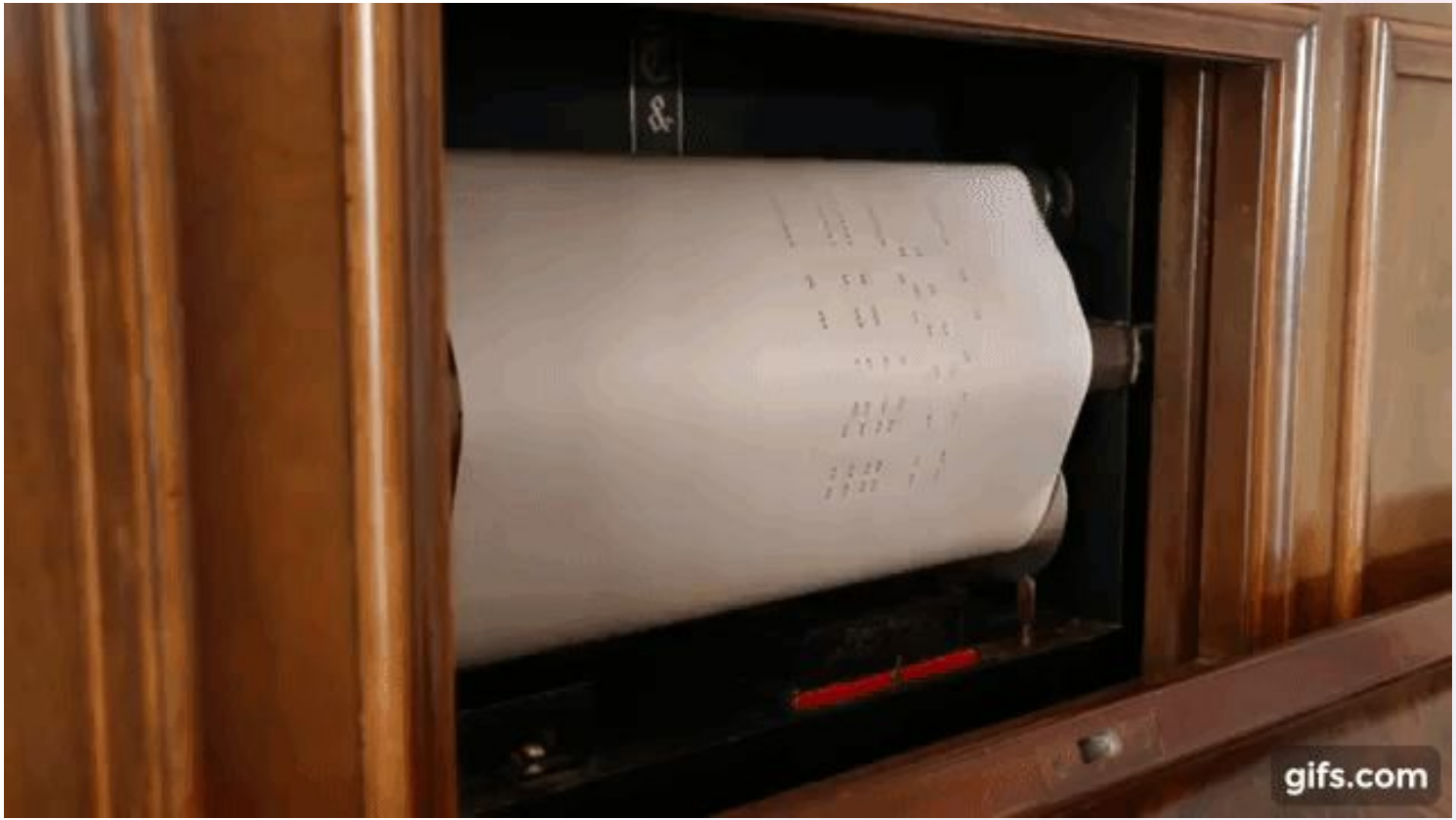
The machine cycle



From source code to machine code



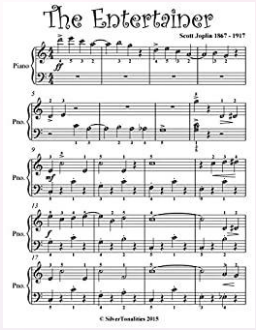
Java’s approach is slightly different

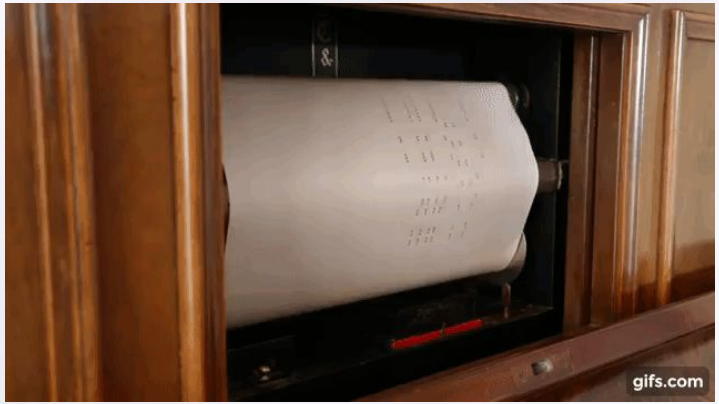

gifs.com

```java
public class Sourcemind {
    public static void main(String args[]) {
        System.out.println("Welcome to Java world!");
    }
}
```

2. Java Virtual Machine (JVM)

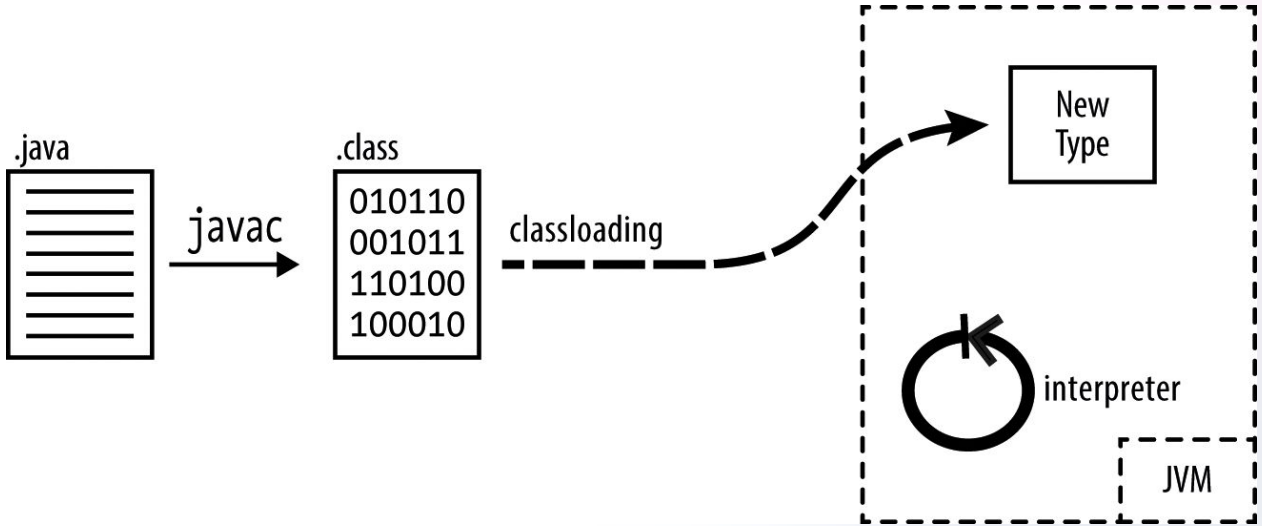
- JVM is a program that runs your Java program
- Your Java program is OS-independent
- JVM implementation is OS-dependant
- Java source files are converted to ***bytecodes***
- Bytecodes are instructions that JVM understands
- JVM executes bytecodes
- The OS executes JVM as an isolated process





sourcemind

JVM executes Java programs



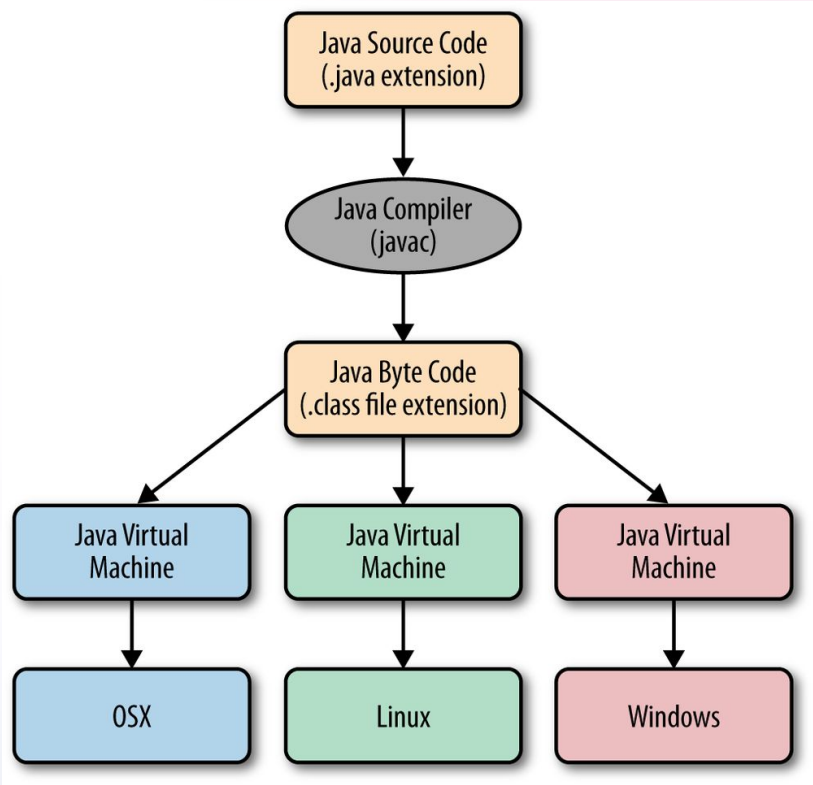
Sourcemind.java

```java
public class Sourcemind {
    public static void main(String args[]) {
        System.out.println("Welcome to Java world!");
    }
}
```

```
% javac Sourcemind.java
% ls
Sourcemind.class     Sourcemind.java
```

Sourcemind.class

```
?????7

<init>()VCodeLineNumberTablemain([Ljava/lang/String;)V
SourceFileSourcemind.java

                        Welcome to Java world!

ourcemindjava/lang/Objectjava/lang/SystemoutLjava/io/PrintStream;java/io/PrintStream
println(Ljava/lang/String;)V!       *??


        %       ???
```



sourcemind

How JVM sees our code? Bytecodes!

```
cafe babe 0000 003d 001d 0a00 0200 0307
0004 0c00 0500 0601 0010 6a61 7661 2f6c
616e 672f 4f62 6a65 6374 0100 063c 696e
6974 3e01 0003 2829 5609 0008 0009 0700
0a0c 000b 000c 0100 106a 6176 612f 6c61
6e67 2f53 7973 7465 6d01 0003 6f75 7401
0015 4c6a 6176 612f 696f 2f50 7269 6e74
5374 7265 616d 3b08 000e 0100 1657 656c
636f 6d65 2074 6f20 4a61 7661 2077 6f72
6c64 210a 0010 0011 0700 120c 0013 0014
0100 136a 6176 612f 696f 2f50 7269 6e74
5374 7265 616d 0100 0770 7269 6e74 6c6e
0100 1528 4c6a 6176 612f 6c61 6e67 2f53
7472 696e 673b 2956 0700 1601 000a 536f
7572 6365 6d69 6e64 0100 0443 6f64 6501
000f 4c69 6e65 4e75 6d62 6572 5461 626c
6501 0004 6d61 696e 0100 1628 5b4c 6a61
7661 2f6c 616e 672f 5374 7269 6e67 3b29
5601 000a 536f 7572 6365 4669 6c65 0100
0f53 6f75 7263 656d 696e 642e 6a61 7661
0021 0015 0002 0000 0000 0002 0001 0005
0006 0001 0017 0000 001d 0001 0001 0000
0005 2ab7 0001 b100 0000 0100 1800 0000
0600 0100 0000 0100 0900 1900 1a00 0100
1700 0000 2500 0200 0100 0000 09b2 0007
120d b600 0fb1 0000 0001 0018 0000 000a
0002 0000 0003 0008 0004 0001 001b 0000
0002 001c
```

javap -v

javap -c (simple)

```
Classfile /Users/vahep/courses/5.-java/java-1/sessions/session-1/examples/Sourcemind.class
  Last modified Jul 16, 2022; size 436 bytes
  SHA-256 checksum fccde5fca503690f0bd9b5476962045e91dfbf444951a6ba0d47e02d446a825e
  Compiled from "Sourcemind.java"
public class Sourcemind
  minor version: 0
  major version: 61
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
  this_class: #21                         // Sourcemind
  super_class: #2                         // java/lang/Object
  interfaces: 0, fields: 0, methods: 2, attributes: 1
Constant pool:
   #1 = Methodref          #2.#3          // java/lang/Object."<init>":()V
   #2 = Class              #4             // java/lang/Object
   #3 = NameAndType        #5:#6          // "<init>":()V
   #4 = Utf8               java/lang/Object
   #5 = Utf8               <init>
   #6 = Utf8               ()V
   #7 = Fieldref           #8.#9          // java/lang/System.out:Ljava/io/PrintStream;
   #8 = Class              #10            // java/lang/System
   #9 = NameAndType        #11:#12        // out:Ljava/io/PrintStream;
  #10 = Utf8               java/lang/System
  #11 = Utf8               out
  #12 = Utf8               Ljava/io/PrintStream;
  #13 = String             #14            // Welcome to Java world!
  #14 = Utf8               Welcome to Java world!
  #15 = Methodref          #16.#17        // java/io/PrintStream.println:(Ljava/lang/String;
  #16 = Class              #18            // java/io/PrintStream
  #17 = NameAndType        #19:#20        // println:(Ljava/lang/String;)V
  #18 = Utf8               java/io/PrintStream
  #19 = Utf8               println
  #20 = Utf8               (Ljava/lang/String;)V
  #21 = Class              #22            // Sourcemind
  #22 = Utf8               Sourcemind
  #23 = Utf8               Code
  #24 = Utf8               LineNumberTable
```

[Class File Format]

[Instruction Set]

sourcemind

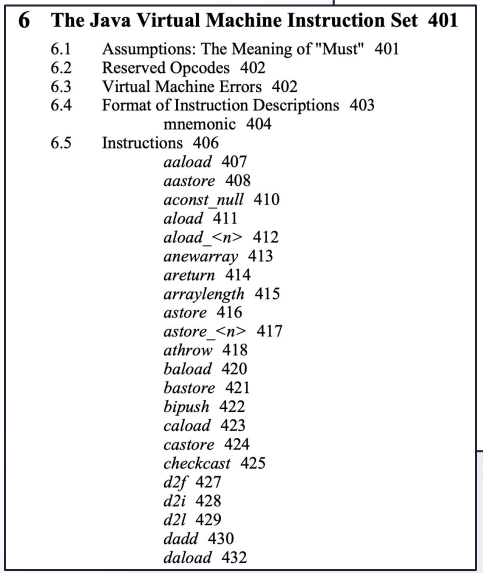
Why?

JVM Design Goals

- Comprise a container for application code to run inside
- Provide a secure and reliable execution environment as compared to C/C++
- Take memory management out of the hands of developers
- Provide a cross-platform execution environment (write once, run anywhere)

JVM is a *specification*

Implement



6 The Java Virtual Machine Instruction Set 401

The Java® Virtual Machine Specification

Java SE 16 Edition

Tim Lindholm
Frank Yellin
Gilad Bracha
Alex Buckley
Daniel Smith

2021-02-12

https://docs.oracle.com/javase/specs/index.html

We will work with OpenJDK 17

Exercise #1

- Download JDK 17
- Install and configure Java on your machine
- Run in command line **`java -version`**

  `echo $JAVA_HOME`
- Write a simple Java program
- Compile and run the program

Exercise #1

- **`javac`**
  Compile source code to bytecodes

- **`java`**
  Execute bytecodes in JVM

- **`jps`, `jmap`**
  Inspect running Java processes

```
% which java
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java

% cd /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin

% ls -l
-rwxr-xr-x  1 root  wheel   54752 Dec  8 20:42 jaotc
-rwxr-xr-x  1 root  wheel   50544 Dec  8 20:42 jar
-rwxr-xr-x  1 root  wheel   50576 Dec  8 20:42 jarsigner
-rwxr-xr-x  1 root  wheel   50608 Dec  8 20:42 java
-rwxr-xr-x  1 root  wheel   50560 Dec  8 20:42 javac
```

Java interactive shell

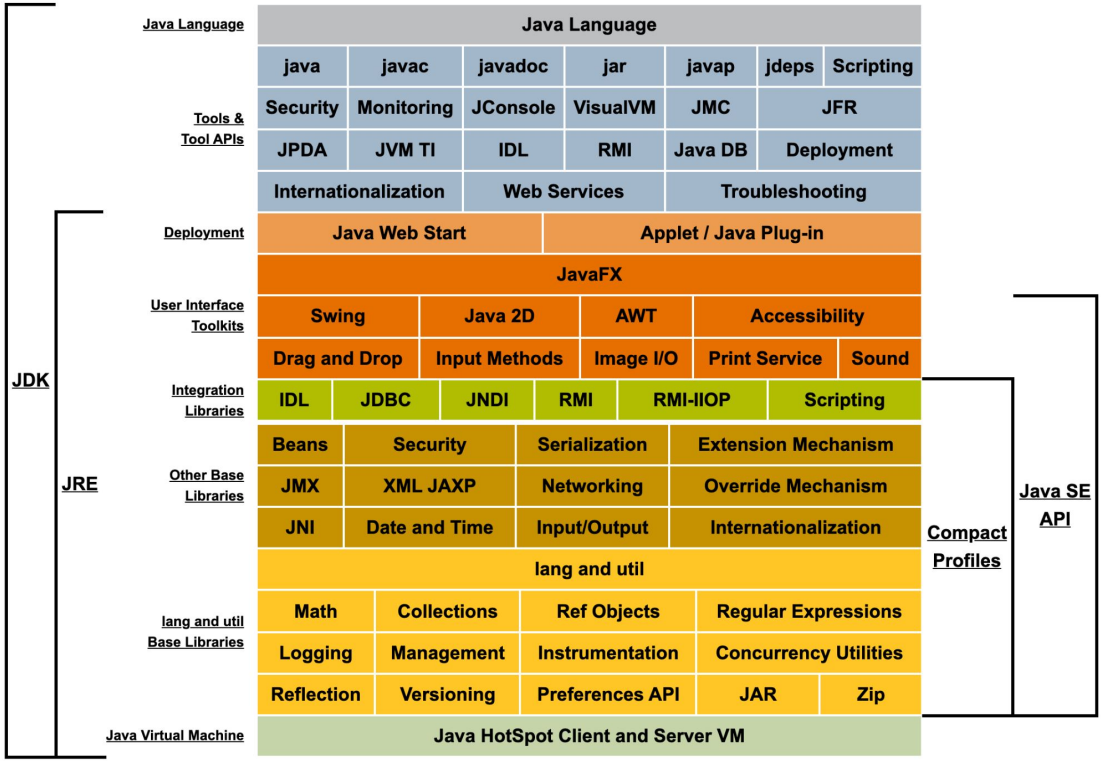
- `jshell`
- Useful for trying code snippets
- Not for development!

```
% jshell

|  Welcome to JShell -- Version 17.0.1
|  For an introduction type: /help intro

jshell> int a = 1
a ==> 1

jshell> String s = (a + 2) + ""
s ==> "3"

jshell> System.out.println(s)
3

jshell> /exit
|  Goodbye
```

3. The Java ecosystem

- Third-party libraries (reusable and hopefully free components from other developers...)
- Frameworks (e.g. Spring, JSF, JUnit, ...)
- Application servers (e.g. Tomcat, JBoss, Glassfish, ...)
- Build tools (e.g. Maven, Gradle, Ant, ...)
- Datasource connectors (e.g. JDBC)
- Messaging systems (e.g. Kafka)
- ...

Oracle Java Products



To develop Java programs, you need JDK

To execute Java programs, you only need JRE

Source:

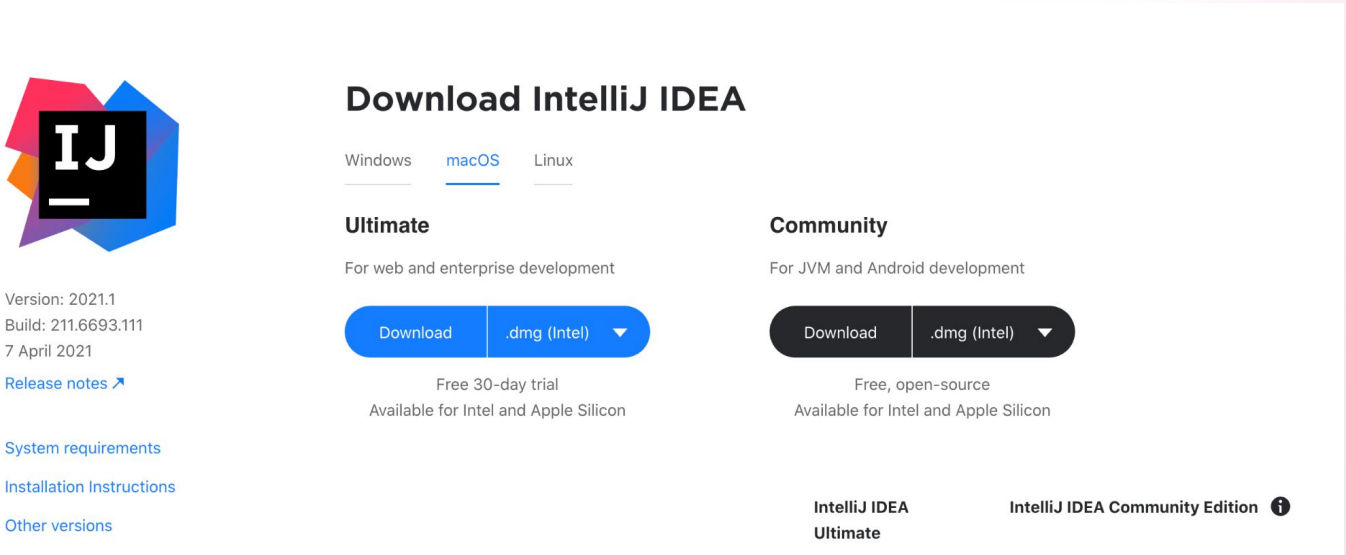
https://docs.oracle.com/javase/8/docs/

sourcemind

Java overview

- Object oriented
- Statically typed
- Multi-threaded
- Portable, platform-independent
- Automatic memory management (GC: garbage collection)
- No pointers or direct access to machine memory
- Mix of interpreted and compiled execution (JIT: Just-in-time compilation)
- Mainly open source
- Widely used, constantly improved
- *Conservative in adopting changes*

Exercise #2: Setting up the IDE



- **IntelliJ IDEA**
- Eclipse
- NetBeans
- BlueJ
- vim
- Sublime
- ...

sourcemind

A Java Program Example

Exercise #3

- Write the provided Java program in your IDE
- Use the command line **javac** tool to compile and **java** to run the program
- Use the IDE to compile and run the program
- Investigate the output

```java
/*
    Our first example
    Receive inputs from OS and print them
*/
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        // Make sure there is at least one argument
        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1); // Report error to OS
        }

       // Print to the OS standard output
        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

```java
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1);
        }

        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

Class declaration

- Every program has a public class
- Filename must match public class name

```java
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1);
        }

        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

Method declaration

- Every program has a ***public static void main*** method
- args = external arguments

```java
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1);
        }

        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

Value assignment

- Declared type (int)
- args.length is also int

```
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1);
        }

        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

```java
public class Example {
    public static void main(String args[]) {
        int len = args.length;

        if (len == 0) {
            System.err.println("No arguments are provided");
            System.exit(1);
        }

        for (int i = 0; i < len; i++) {
            System.out.println("Arg " + i + ": " + args[i]);
        }
    }
}
```

```
% javac Example.java
```

Compile

```
% java Example
No arguments provided

echo $?
1
```

```
% javac Example.java
```

```
% java Example
No arguments provided
```

Execute

```
echo $?
1
```

```
% javac Example.java

% java Example
No arguments provided
```

```
echo $?
1
```

$? gives exit status of last command (1 = error)

```
% javac Example.java

% java Example Hi this is an example
Arg 0: Hi
Arg 1: this
Arg 2: is
Arg 3: an
Arg 4: example

echo $?
0

% java Example "Hi this is an example"
```

Execute and provide arguments

```
% javac Example.java

% java Example Hi this is an example
Arg 0: Hi
Arg 1: this
Arg 2: is
Arg 3: an
Arg 4: example

echo $?
0

% java Example "Hi this is an example"
```

```
% javac Example.java

% java Example Hi this is an example
Arg 0: Hi
Arg 1: this
Arg 2: is
Arg 3: an
Arg 4: example

echo $?
0

% java Example "Hi this is an example"
Arg 0: Hi this is an example
```

Exercise #4

- ExampleAdd is not a **`public class`**.
- **`+`** operator is used for integers and strings.
- `print()` vs. `println()`
- Create another class, `ExampleMult` that multiplies two numbers.
- Can you use a `Program.java` file to write the program? Why?
- Try using `public` for `ExampleAdd` and `ExampleMult` classes.

```java
// Exercise 4: Add two numbers
class ExampleAdd {
    public static void main(String args[]) {
        int a = 3;
        int b = 2;

        // Print initial values
        System.out.print("a: " + a);
        System.out.println(", b: " + b);

        int sum = a + b;

       // Print the addition result
        System.out.println("a + b = " + sum);
    }
}
```

Summary

- The name of the class file (.java) should match the name of the public class.
- Each file can contain only one public class.
- The file may contain multiple non-public classes.
- The name of the non-public classes do not matter, however, it is a good practice to always use the same name as the class for the containing file.
- The class where the program begins must have a

  `public static void main(String[] args) { … }`

  method.
- The `main()` method is invoked by the JVM.
- Input from OS: `String[] args`
- Output to OS: Standard output, Standard error, System.exit(exitCode)

Homework 1

Write the following programs:

1. A program that accepts two integer inputs from user and prints the larger number.
2. A program that accepts a real number and prints its ceiling value. (e.g. 3.2 → 4)
3. A program that accepts arbitrary number of integer values, sorts them, and prints the sorted numbers as the output.

End of session 1