

Java in Education

<Insert your name here if using>
JUG Presentation
For CS Instructors and Students

Prepared by Ken Fogel & the JCP Executive
Committee (EC) Java in Education Working Group

Version 2.3



Myths & Benefits of Learning Java

- Java can be a steep learning curve for a beginner
 - *Only if the instructor themselves had a steep curve in learning the language*

Myths & Benefits of Learning Java

- Java is not suitable for lightweight, quick tasks
 - Better suited for larger and more complex applications.
 - *Have you seen Multi-File Source-Code and under Linux have you tried shebang execution?*
 - *This Just In! – **Paving the on-ramp***

Myths & Benefits of Learning Java

- Oracle Java Development Kit (JDK), is not open source
 - *OpenJDK is a completely open source implementation of the JDK*
 - *Continuing development of Java is done in the OpenJDK project by Oracle Java developers*

Myths & Benefits of Learning Java

- Java is an “old” language (Java 1996 & Python 1991)
 - *also means it's established, widely used and well-documented*

Myths & Benefits of Learning Java

- More Java programmers than any other type of programmer in the world
 - *easy to find people who can help you out and mentor you*

Myths & Benefits of Learning Java

- *Java (and its JVM variant Kotlin) are the basis of Android development*

Java Language Enhancements

This presentation looks at enhancements to the Java language

These enhancements help dispel some of the myths surrounding Java.

It is about why Java should be the language taught at all levels in schools today.

JShell - Read- Evaluate- Print Loop (REPL) JDK 9

A tool for simplifying instruction.

Execution as you enter code and press return.

Immediate response line by line.

You can also write entire methods first and then execute them.

Ideal in teaching Java one line at a time.

JEP 458 - Launch
Multi-File Source-
Code Programs
JDK 22

- Addresses the overhead of running code
 - **Traditional Style**
 - Two-step to execution
 - javac
 - java -jar

JEP 458 - Launch
Multi-File Source-
Code Programs
JDK 22

- **Multi-File Source-Code Style**
 - One-step to execution
 - `java`
 - If the file has a public class with a main it compiles and executes
 - Now you can have multiple class files in the same folder or in a subfolder
 - You can even include jar files
- No need to master an IDE

Preview Features

- New features in the Java language are not immediately available
- They are designated Preview features and a switch on the command line or in your IDE must be set.

```
javac --source 23 --enable-preview Main.java
```

```
java --enable-preview Main
```

- Or using the source code launcher

```
java --enable-preview Main.java
```



Too many decorations!

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Step 1: Java 22 - Implicitly Declared Classes and Instance Main Methods


- The most common complaint about Java is its unsuitability, as compared to other languages, for beginners
- The ultimate simplification
 - no need for any class declaration
 - no need for access control declarations



Step 1: Java 22 - Implicitly Declared Classes and Instance Main Methods

- **main** can be expressed as an instance method.
- Can be used in any Java program, not just implicit classes.
- Here is a complete Java program that can be compiled and executed.

```
void main() {  
    System.out.println("Hello World");  
}
```



Step 2:
Java 23 -
Implicitly
Declared
Classes and
Instance
Main
Methods

Coming soon in Java 23!

To be available in every implicit class

```
public static void println(Object obj);
```

```
public static void print(Object obj);
```

```
public static String readln(String prompt);
```



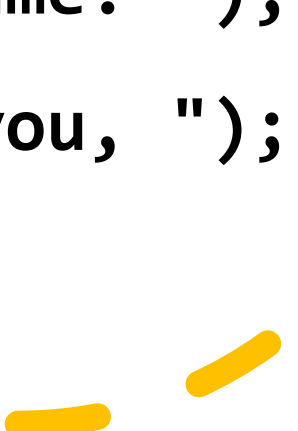
Step 2:
Java 23 -
Implicitly
Declared
Classes and
Instance
Main
Methods

Here are complete Java programs:

```
void main() {  
    println("Hello, World!");  
}
```

How about some input:

```
void main() {  
    String name = readln("Name: ");  
    print("Pleased to meet you, ");  
    println(name);  
}
```



var – reduction of redundancy reduction JDK 10

No more:

- `MyClass m = new MyClass();`

It now becomes:

- `var m = new MyClass();`

Encourages only
creating objects with
initialization

- Will reduce the occurrence of the dreaded `NullPointerException`

text blocks (15)

Finally, what you enter into your source code is what you get



Especially useful for Strings that contain HTML, XML and JSON



Who doesn't like writing three quotation marks in a row?

|||||

|||||

Old School Concatenation

```
String htmlStr = "<html><head><link rel='stylesheet' "
    + "href='styles/main.css' "
    + "type='text/css'/><title>The Learning Servlet</title></head>"
    + "<body><h1>GET method</h1>"
    + "<form id='form:index' action='index.html'>"
    + "<br/><input type='submit' value='Return to Home page'/></form>"
    + "</body></html>";
```

New School Text Block JDK 15

```
String htmlStr = ""
<html>
  <head>
    <link rel='stylesheet'href='styles/main.css' type='text/css'/>
    <title>The Learning Servlet</title>
  </head>
  <body>
    <h1>GET method</h1>
    <form id='form:index' action = 'index.html'>
      <br/>
      <input type= 'submit' value='Return to Home page' />
    </form>
  </body>
</html>"";
```

But wait, there is more . . . String formatted JDK 15

```
out.println("""
<html>
  <head>
    <title>Just Servlet Output</title>
    <link rel='stylesheet' href='styles/main.css' type= 'text/css'/>
  </head>
  <body>
    <h1>Thanks for joining our email list</h1>
    <p>Here is the information that you entered:</p>
    <label>Email:</label>
    <span>%s</span>
  </body>
</html>""").formatted(user.getEmailAddress());
```

switch – an expression & without a break JDK 14

A switch that can be explained sensibly

Reduction in duplication of code when used to set a value

Switch expressions or switch rules

The end of break, all cases terminate!

Which would you prefer to learn or teach?

```
double value = 0;
switch (point) {
    case NORTH:
        value = 12.12;
        break;
    case SOUTH:
        value = 14.14;
        break;
    case EAST:
        value = 16.16;
        break;
    case WEST:
        value = 18.18;
        break;
}
```

```
double value = switch (point) {
    case NORTH -> 12.12;
    case SOUTH -> 14.14;
    case EAST -> 16.16;
    case WEST -> 18.18;
    default -> 0.0;
};
```


Java 21 The pattern matching switch.

```
Object x = "4";
String designation = switch (x) {
    // case Integer i when i > 4 && i < 12 -> "child";
    case Integer i when i < 12 -> "child";
    case Integer i when i < 18 -> "teenager";
    case Integer i when i < 25 -> "young adult";
    case Integer i when i < 65 -> "adult";
    case Integer i when i >= 65 -> "senior";
    default -> "Not an Integer";
};
System.out.printf("Designation is %s\n", designation);
```

records – boilerplate reduction with immutable flavouring and a dash of compact constructor JDK 16

Data objects are known for boilerplate code:

- Initializing constructors, setters, getters, equals, hashCode, and toString

To the rescue is the immutable record

More than just a simplification of a bean

It's the path to objects defaulting to immutability

And then there is the compact constructor

- Validating initial values without a separate constructor

No setters, just simple getters.

Implied equals, hashCode and toString.

And what a lovely compact constructor for validation.

```
public record Person(String firstName,  
                    String lastName,  
                    int age,  
                    String postion,  
                    LocalDate birthday) {  
    public Person{  
        if (age < 18) {  
            throw new IllegalArgumentException("Too young");  
        }  
    }  
}
```

Virtuous Virtual Threads

Java threads, not OS threads

JDK 21.

```
public class VirtualThreadClass extends Thread { . . }
```

```
public void perform() {  
    for (int i = 0; i < 5; ++i) {  
        Thread.ofVirtual().name("Thread # " + i).  
            start(new VirtualThreadClass());  
    }
```

What's Pushing Java Aside?

JavaScript

- Little to download
- Available in the browsers on every school PC

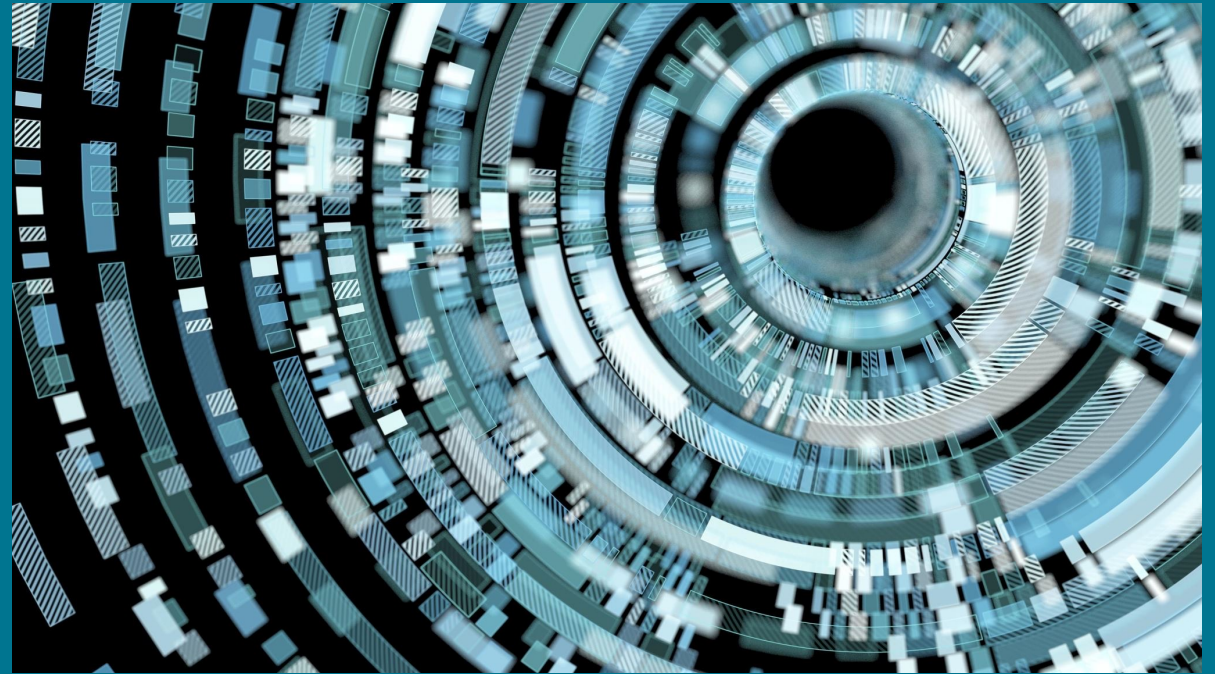
Python

- Associated with the two big trends:
 - Big Data
 - AI/ML
- Online Jupyter notepad is popular

Machine Learning and Big Data

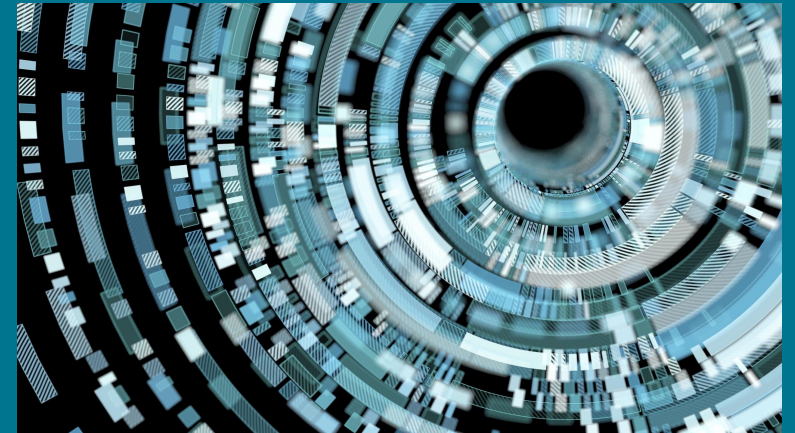
VisiRec JSR 381

- Java is doing machine learning now!
- Amazon's Deep Java Library (DJI) is one of several implementations of this new JSR
- The depth and breadth of Java tooling make it the best platform for ML



Why is Python widely used for AI/ML?

- Python is written in C such that it can easily interact with C libraries
- As many AI/ML libraries are written in C, Python can more easily interact with them
- A weakness of Java has been interacting with other languages
- With Java 22 we have a Foreign Function & Memory API that will greatly simplify accessing C libraries



The Java Virtual Machine – Home to More Than Java

- Kotlin, Scala, Groovy, Clojure and more
- There is even a Python called Jython that runs on the JVM and supports interoperability between Java and Python

Why teach Java to students?



Many financial institutions depend on Java to run their backend



Twitter, LinkedIn, Amazon and others use Java



Your prospects are a function of how well you code

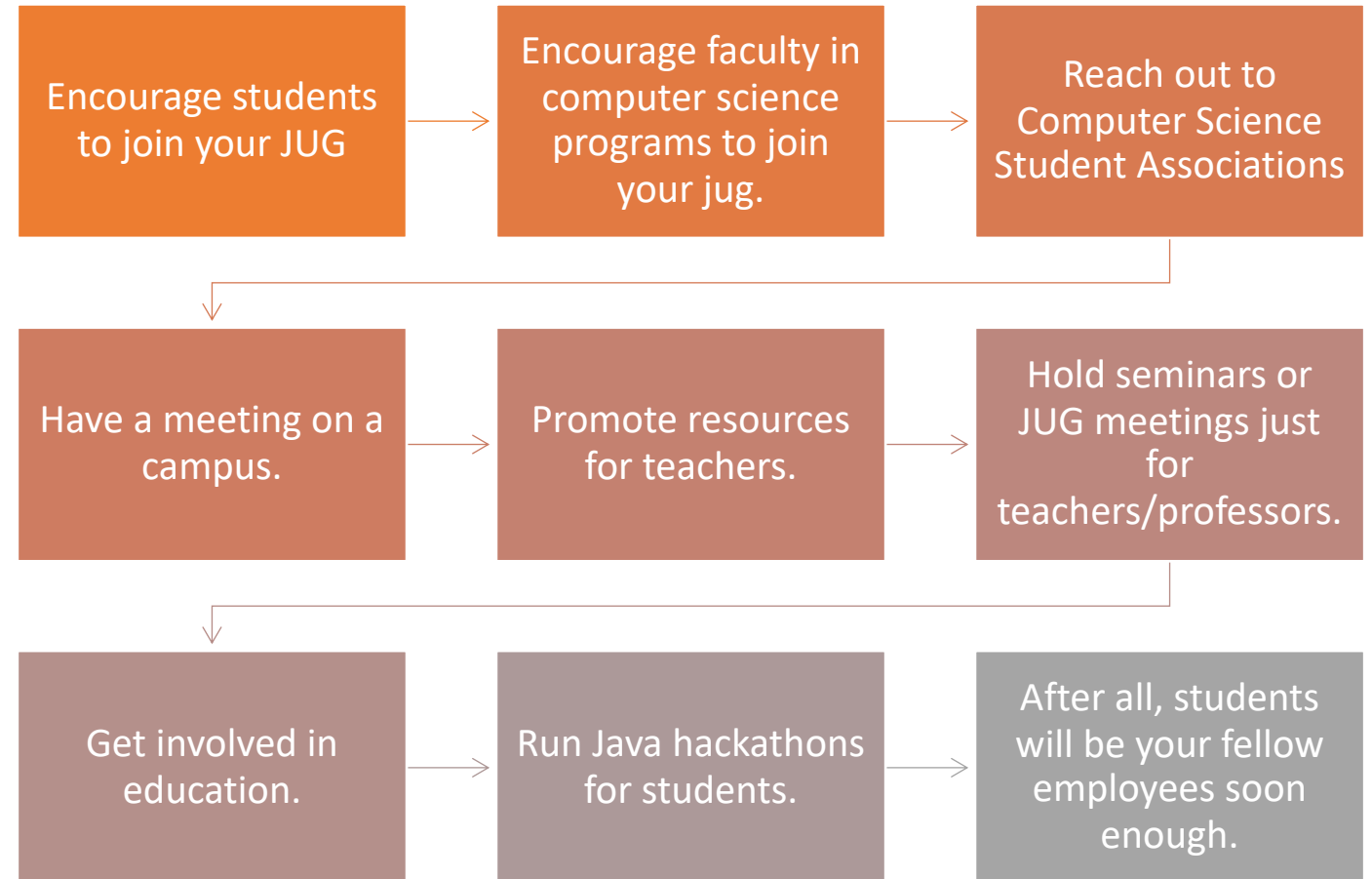


The best language to learn to prepare you to work with any language during your career.



The best language for giving students a clear understanding of what it means to program.

Conclusion – Reach Out To Schools and Teachers/Professors at All Levels



Sample code can be found at:

[https://github.com/omniprof/JCP_EC](https://github.com/omniprof/JCP_EC_Education_WG_Presentation)
Education WG Presentation