Java in Education

<Insert your name here if using>

JUG Presentation

For Junior Developers and Students

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

Myths & Benefits of Learning Java

- Java can be a fairly steep learning curve for a beginner
  - *Only if the instructor themselves had a steep curve in learning the language*
- Java is not suitable for lightweight, quick tasks
- Better suited for larger and more complex applications.
  - *Have you seen Single-File Source-Code and under Linux have you tried shebang execution?*
  - *This Just In!* ***Unnamed Classes and Instance Main Methods***
- 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*
- More Java programmers than any other type of programmer in the world
  - *easy to find people who can help you out and mentor you*
- Java derives its syntax from C
  - *learn Java, then learning a language like JavaScript, C#, C++ & even Python is much easier*
- *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.

There is even a comparison of a program in Java and Python.

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 330 - Launch Single-File Source-Code Programs JDK 11

- Addresses the overhead of running code
  - Traditional Style
    - Two-step to execution
      - javac
      - java -jar
  - Single-File Source-Code Style
    - One-step to execution
      - java
        - If the file has a public class with a main it compiles and executes
        - Works with preview features as well as established features
        - Single file may contain multiple classes
- This may be the second* most significant new capability for writing Java for those wishing to learn the language
- No need to master an IDE to learn Java.

* The first will be shown on slide 18.

Java 21 - Unnamed Classes and Instance Main Methods Preview

- The most common complaint about Java is its unsuitability, as compared usually to Python, for beginners
- JEP 330 - Launch Single-File Source-Code was a first step in simplification
- JEP 445 - Unnamed Classes and Instance Main Methods is the next step
  - no need for any class declaration
  - import statements permitted
  - package statement not allowed
  - constructors not allowed

Java 21 - Unnamed Classes and Instance Main Methods Preview

- Here is a complete Java program that can be compiled and executed.

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

- Expected in JDK 22 is the removal of `System.out`

The New and Improved main! Useable anywhere!

1. `static void main(String[]args)`
2. `static void main()`
3. `void main(String[] args)`

And our favourite:

4. `void main()`

- Single File Execution:
- `java --enable-preview --source 21 file.java`
- Compile & Execute
- `javac --enable-preview --source 21 file.java`
- `java --enable-preview file`
- You can use any or all the new **`main`** method formats.
- They will execute in the order shown here, 1, then 2, then 3, then 4

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
JDK 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()));
```

And still more . . . String template JDK 21 preview

```
String email = "person@mail.com";
String page = STR."""
  <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>\{email}</span>
     </body>
  </html>""";
System.out.println(page);
```

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?

```java
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;
}
```

```java
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 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
- Numerous online programming environments

Python

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

Why is Python Gaining Popularity In Education?

The most feared design pattern:

- Stream of Consciousness

Programs flow as tasks come to mine

Appeals to individuals who need to code but who don't necessarily want to learn to code professionally.

Let’s Compare Python to Java Discuss them as you review them.

On the next slides is the same program in Python and Java

These programs request three floating point values

- Amount of money borrowed called the loan
- The annual percentage rate (APR) for interest on the borrowed money
- The length of the load expressed in months called the term

From these values the program calculates the monthly repayment and displays it

Basic Python

```
loan = 1000.0
interest 0.05
term = 5
tempInterest = float(interest) / 12
result = float(loan) * \
   (tempInterest / (1.0 - ((1.0 + tempInterest) ** -float(term))))
print("Monthly Payment: %.2f" % result)
```

Classic Java

```
public class JavaCalculator01 {

    public static void main(String[] args) {

        double loan = 1000.0;
        double interest = 0.05;
        double term = 5;

        var tempInterest = interest / 12.0;
        var result = loan * (tempInterest / (1.0 - Math.pow((1.0 + tempInterest), -term)));

        System.out.printf("Monthly Payment: %.2f%n", result);
    }
}

// Single Source File Code example
// runs with java JavaCalculator01.java
```

Nameless Class Java

```
void main() {

    var loan = 1000.0;
    var interest = 0.05;
    var term = 5;

    var tempInterest = interest / 12.0;
    var result = loan * (tempInterest / (1.0 - Math.pow((1.0 + tempInterest), -term)));

    System.out.printf("Monthly Payment: %.2f%n", result);
}

// Single Source File Code example
// runs with java --enable-preview --source 21 JavaCalculator01N.java
```

OOP Python

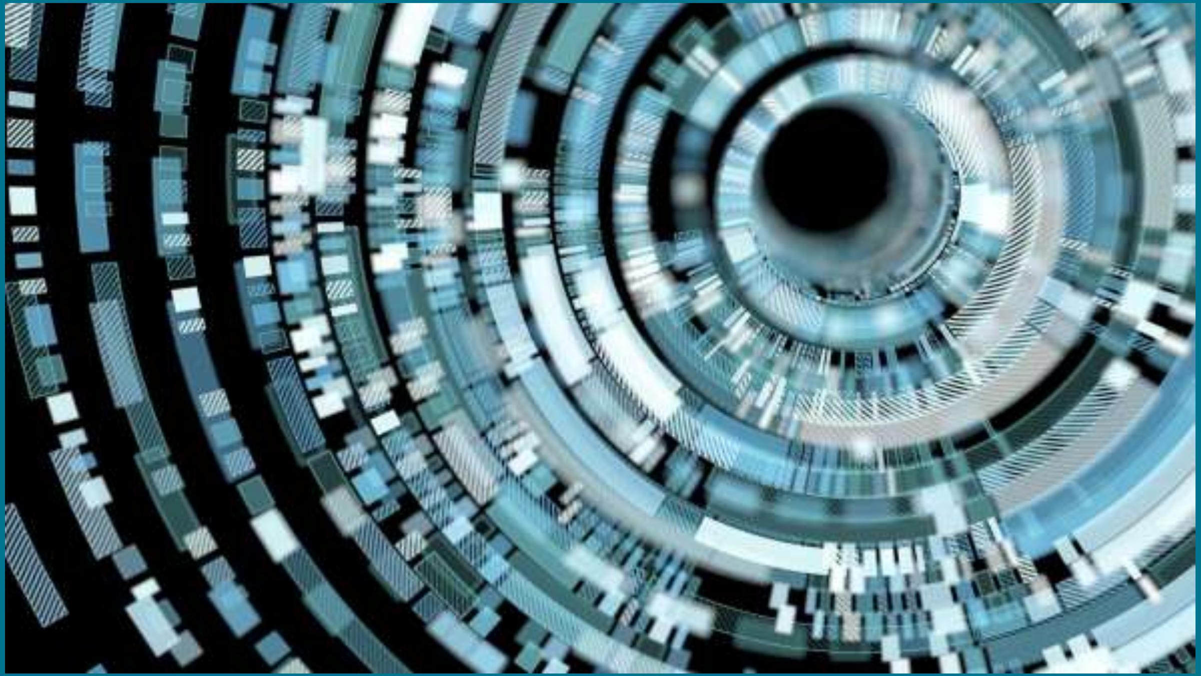
```
class PythonCalculator03:

    def func_input(self):
        loan = float(input("           loan: "))
        interest = float(input("       interest: "))
        term = float(input("           term: "))
        return loan, interest, term

    def func_process(self, input_data):
        (loan, interest, term) = input_data
        temp_interest = float(interest) / 12.0;
        return loan * (temp_interest / (1.0 - ((1.0 + temp_interest) ** -term)));

    def func_output(self, result):
        print('Monthly Payment: %.2f' % result)

    def func_work(self):
        input_data = self.func_input()
        result = self.func_process(input_data)
        self.func_output(result)


worker = PythonCalculator03()
worker.func_work()
```

OOP Java

```
public class JavaCalculator03 {

    private LoanRecord inputData() {
        var sc = new Scanner(System.in);
        System.out.printf("           Loan: ");
        var loan = sc.nextDouble();
        System.out.printf("       Interest: ");
        var interest = sc.nextDouble();
        System.out.printf("           Term: ");
        var term = sc.nextDouble();
        return new LoanRecord(loan, interest, term);
    }

    private double processData(LoanRecord loan) {
        var tempInterest = loan.interest() / 12.0;
        var result = loan.loan() * (tempInterest / (1.0 - Math.pow((1.0 + tempInterest), -loan.term())));
        return result;
    }

    private void outputResult(double result) {
        System.out.printf("Monthly Payment: %.2f%n", result);
    }

    public void perform() {
        var loan = inputData();
        var result = processData(loan);
        outputResult(result);
    }

    public static void main(String[] args) {
        new JavaCalculator03().perform();
    }
}

record LoanRecord(double loan, double interest, double term) {}
```

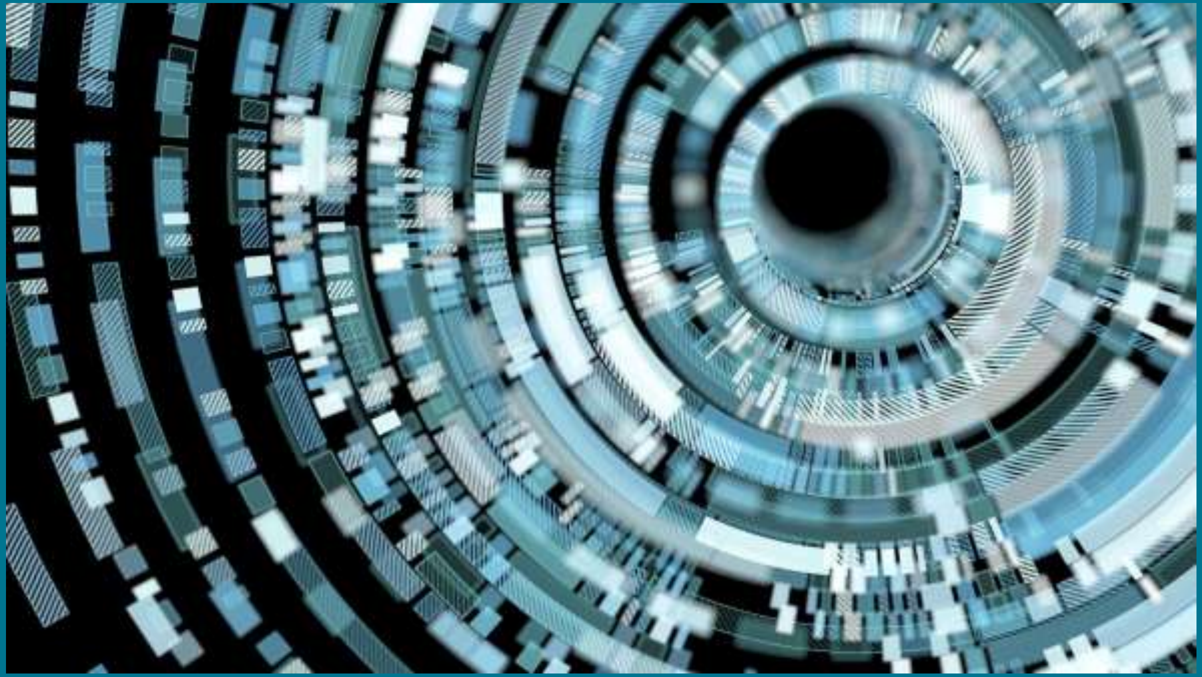
Machine Learning and Big Data
VisiRec JSR 381

- Java is doing machine learning now!
- Amazon's Deep Java Library (DJL) 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?

- The language Python is written in is C
- Most AI/ML libraries are written in C
- This simplifies using these libraries in Python
- With Java 21 we will have a Foreign Linker API & Foreign Memory Access API that will simplify accessing C libraries



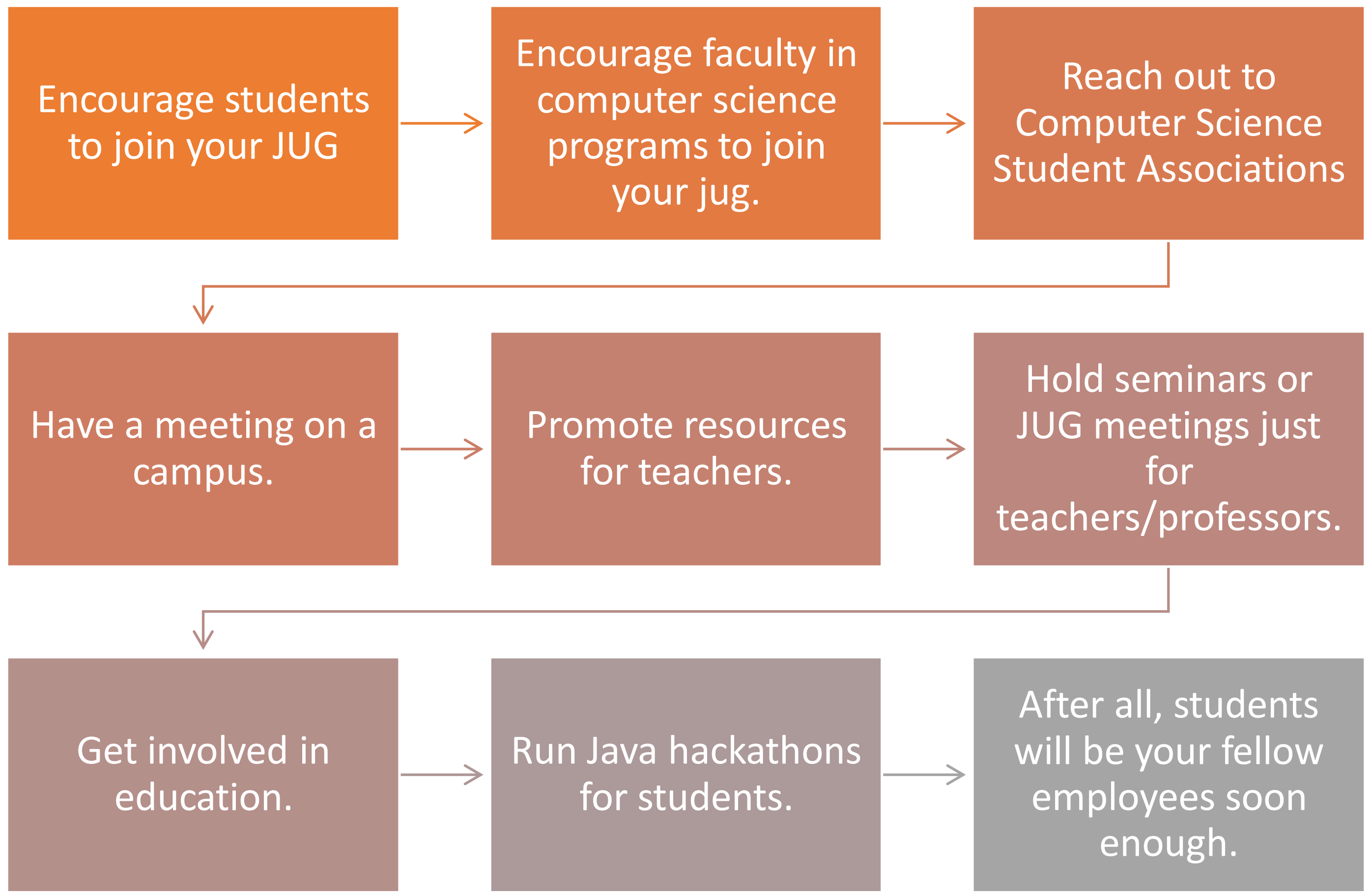
The Java Virtual Machine – Home to More Than Java

- 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

What are your job prospects if you learn Java?

- 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
- Learning Java is the best language to learn to prepare you to work with any language during your career.
- It's the best language to teach to give students a clear understanding of what it means to program.

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


Encourage students to join your JUG
Encourage faculty in computer science programs to join your jug.
Reach out to Computer Science Student Associations
Have a meeting on a campus.
Promote resources for teachers.
Hold seminars or JUG meetings just for teachers/professors.
Get involved in education.
Run Java hackathons for students.
After all, students will be your fellow employees soon enough.

*Discounts for all User Group members from Oracle University here : https://education.oracle.com/usergroupchampions
Currently 25% discount through 12/21/2020.

Sample code can be found at:

https://github.com/omniprof/JCP_EC_Education_WG_Presentation