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

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 most significant new capability for writing Java for those wishing to learn the language

var – reduction of redundancy reduction

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

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

```
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>"""
```

switch – an expression & without a break

- 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;
};
```

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

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
Free 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 to work for us!");
        }
   }
}
```

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

```
loan = input("           loan: ")
interest = input("       interest: ")
term = input("           term: ")
tempInterest = float(interest) / 12;
result = float(loan)*(tempInterest / (1 - ((1 + tempInterest) ** -float(term))));
print("Monthly Payment: %.2f" % result)
```

```
import java.util.Scanner;

public class JavaCalculator01 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("           Loan: ");
        double loan = sc.nextDouble();
        System.out.print("       Interest: ");
        double interest = sc.nextDouble();
        System.out.print("           Term: ");
        double term = sc.nextDouble();
        double tempInterest = interest / 12.0;
        double result = loan *
                (tempInterest / (1.0 - Math.pow((1.0 + tempInterest), -term)));
        System.out.println("Monthly Payment: " + String.format("%.2f", result));
    }
}
```

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()
```

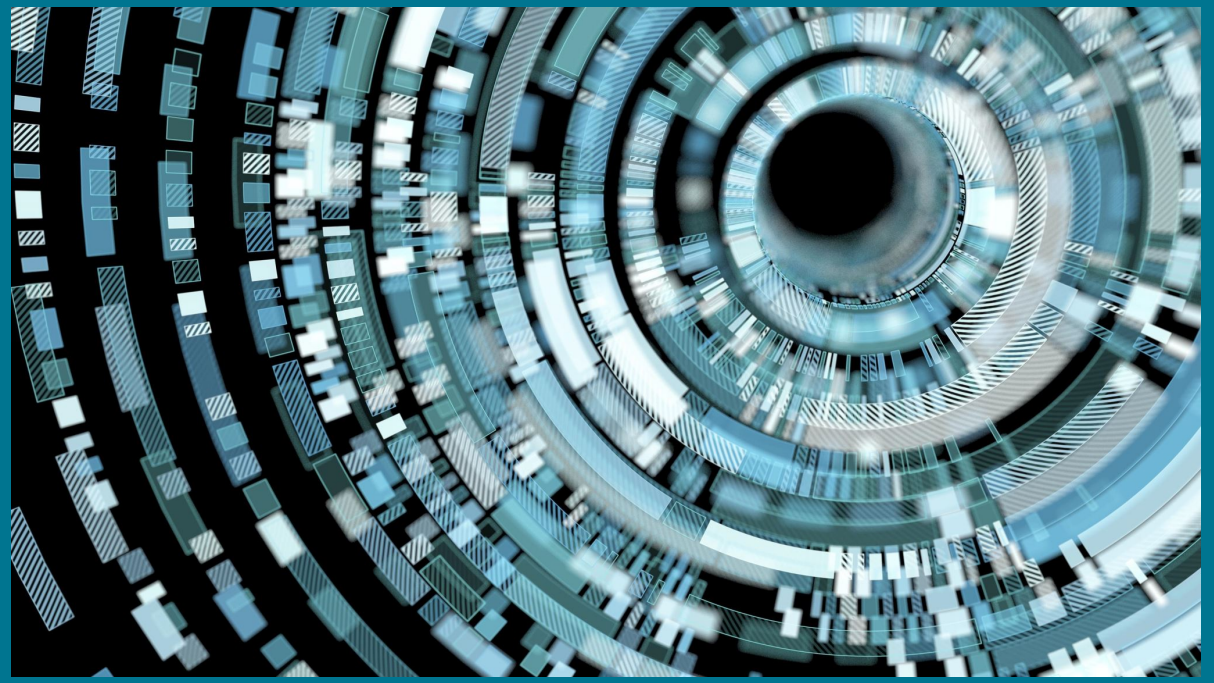
OOP Python

```
import java.util.Scanner;

public class JavaCalculator03 {
    private LoanRecord inputData() {
        Scanner sc = new Scanner(System.in);
        System.out.print("           Loan: ");
        double loan = sc.nextDouble();
        System.out.print("       Interest: ");
        double interest = sc.nextDouble();
        System.out.print("           Term: ");
        double term = sc.nextDouble();
        return new LoanRecord(loan, interest, term);
    }
    private double processData(LoanRecord loan) {
        double tempInterest = loan.interest() / 12.0;
        double result = loan.loan() * (tempInterest / (1.0 - Math.pow((1.0 + tempInterest), -loan.term())));
        return result;
    }
    private void outputResult(double result) {
        System.out.println("Monthly Payment: " + String.format("%.2f", result));
    }
    public void perform() {
        var loan = inputData();
        var result = processData(loan);
        outputResult(result);
    }
    public static void main(String[] args) {
        JavaCalculator03 calc = new JavaCalculator03();
        calc.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



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

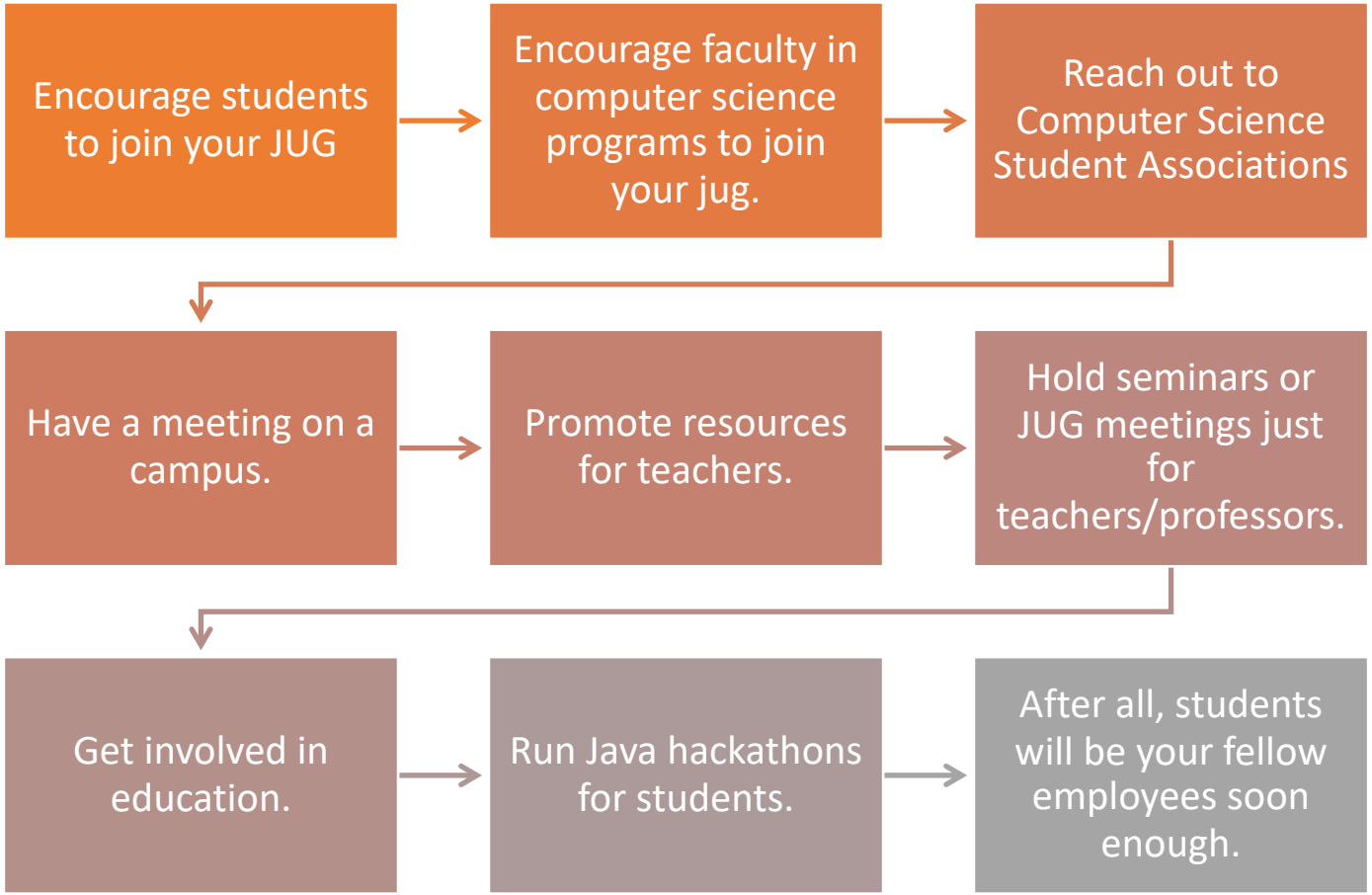
Twitter, LinkedIn, Amazon and others use Java

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



*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