**22W\_CST8116\_451 Intro to Computer Programming**

Assignment #1 Java Programming Environment

**Carol Ann Wilson**

Student ID: 040672794

Instructor: Piyush Jangam

Date: January 11, 2022

# Assignment #1 Java Programming Environment

## Outline key points in the history of Java Programming Language.

The Java programming language was started by James Gosling and Patrick Naughton of Sun Microsystems in 1991. Originally code named “Oak” the language was meant to be a simple, secure, and OS/Processor agnostic, meaning that any system that could run a Java VM would be able to run the written Java program. A simple programming language that borrows syntax from C and C++ while making its usage simpler and more uniform. The use of Java as a programming language has grown significantly since its inception. It is easier to use than C++ and allows for complex usage because of the large number of libraries available that allow Java to interact with various components such as GPUs, networking components, etc.

Originally, Java was developed for use in consumer devices, such as the setup boxes for a television. However, no customer was ever found for the technology. Of course, this has long since changed. Java, in its various forms (Java EE, Java ME, etc.) now appears in cell phones, on websites, in your television, and many other uses.

Java was a proprietary standard owned by Sun Microsystems. In the years since its development Java now includes additional releases such as Amazon Corretto or OpenJDK, expanding its usability for enterprise or personal use.

## Explain the three types of Java Programming, using examples.

There are three types of Java Programming:

* Applications
* Applets
* Servlets

### Applications

Applications are programs that are applied to specific tasks. The Microsoft Office suite is a prime example of an application. Word is used for word-processing, Excel for spreadsheets, etc.

### Applets

Applets are smaller programs that can run within a web page and usually contain some level of security. Applets are delivered to users in bytecode and executed on the client side. The applet might run within the web page you’re on, in a new window, in a new frame, or even a pop-out widget of some sort. It is separate from the web page itself, even though it runs from within the page however, applets are NOT standalone programs. Applets MUST run on a Java compatible browser, or they will not work. They make a website more interesting for users.[[1]](#footnote-1)

One example of an applet would be a web page that includes an embedded Fahrenheit to Celsius converter.

### Servlets

Servlets are Java programs that perform their tasks on the server-side of a web server or application server extending the capabilities of a server. Servlets can handle complex requests through the webserver that can include things such as serving up web pages, displaying forms, acting on the information submitted through that form, and so on.

“Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.”[[2]](#footnote-2)

Signing up for a web account, such as the Kobo book reader account, could be done using a Servlet.

## What is Object Oriented Programming language?

Object Oriented Programming languages uses the concept of classes and objects that allows you to create code that is structured, reusable, and simple. One of the best non-programming examples is in the Big Java textbook of constructing a house. A homebuilder builds the house, but uses common elements such as windows, water heaters, etc., that they do not build themselves. These would be reusable objects in a Java program.

Object Oriented languages use objects that are self-contained and reusable for more than one problem and tend to focus on the information than it does the process.

### How does it differ from Procedure Oriented Programming?

Procedure Oriented Programming languages are more sequential, following a step-by-step approach to accomplishing your task. It’s very linear and takes each task that needs to be done and puts the steps in the program in order. The biggest downside is that procedural code is usually not reusable. It tends to focus more on the process than it does on the information, which can be a serious detriment. I feel that this method probably has less flexibility in the long term.

One of the best pieces of information I found was on the hackr.io blog talking about procedural and object-oriented programming and comparing the two. The table I’ve copied below provides a great comparison of the differences between the two methods:[[3]](#footnote-3)

|  |  |
| --- | --- |
| **Procedural Programming** | **Object-Oriented Programming** |
| Uses immutable data | Uses mutable data |
| Follows the declarative programming model | Follows the imperative programming model |
| Extends support to parallel programming | Not suitable for parallel programming |
| The execution order of statements is not the primary focus | The execution order of statements is very important |
| Flow control is performed using function calls | Flow control is performed through conditional statements and loops |
| Uses recursion concept to iterate collective data | Uses loop concept to iterate collection data |
| No such side-effects of its functions | The method can have certain side-effects |
| The focus in Procedural Programming is on **‘What You are Doing’** | The focus in Object-Oriented Programming is on **‘How You are Doing It’** |

## Write a simple Java Program

**import** java.util.Scanner;

**public** **class** assignment1 {

**public** **static** **void** main(String[] args) {

Scanner userInput = **new** Scanner(System.***in***);

//This allows me to intake the username I am requesting in the next line

System.***out***.println("Welcome to Kat Industries, please enter your username.");

//Tells the user to enter their username

String userName = userInput.nextLine();

/\*Reads the user input, the nextLine statement uses the return

to indicate that the user is done entering\*/

userInput.close();

/\*closes the scanner, while cleanup will take care of memory resources

it’s always good practice to close something you open when you’re done\*/

System.***out***.println("Thank you " + userName + ". Welcome to the system.");

}

}

### Explain the structure of Java Language with reference to that program.

1. Documentation: Optional but recommended. Would usually include the author of the program, creation date, maybe the version of the software, etc. The only documentation included in my program at this point are the comments that appear, for example:

//This allows me to intake the username I am requesting in the next line

1. Import Statements: In my program above, I’m using an import statement because I wanted to use the Scanner functionality that allowed me to input a piece of information to the program:

**import** java.util.Scanner;

1. Class Definition: You cannot leave out a class definition. Every piece of Java software has a class definition. In my case I’ve named my class assignment1, so my class statement is:

**public** **class** assignment1

1. The next stage would be any variables or constants that you need to declare. My program doesn’t really declare any variables or constants, and there are different schools of thought on where these should appear. I work with two developers, one of whom believes they should be declared at the beginning and the other who believes they should be declared just before the location they are being used in. My understanding is that either way is acceptable.
2. Main Method: Again, like the class definition, you must always have a main method defined. This is where the actual execution of the program starts and invariably starts with public static void main() {:

**public** **static** **void** main(String[] args) {

1. Inside your main method you have a variety of statements that tells the software what to do. My program statements:

Scanner userInput = **new** Scanner(System.***in***);

System.***out***.println("Welcome to Kat Industries, please enter your username.");

String userName = userInput.nextLine();

userInput.close();

System.***out***.println("Thank you " + userName + ". Welcome to the system.");

And of course, never forget your opening and closing braces or your semi-colons!

## Explain each phase of Java Program Development Life Cycle using appropriate diagrams and examples.

There are many different viewpoints on the phases of the Java Program Development Life Cycle. In my work environment our developers work with an Agile Software Development Life Cycle. In our Brightspace content you see one type of lifecycle, in our Big Java textbook you see a slightly different lifecycle.

|  |  |
| --- | --- |
| **Brightspace:** | **Big Java:** |
| 1. Understand the problem 2. Plan the logic 3. Code the program 4. Use software (a compiler or interpreter) to translate the program into machine language 5. Test the program 6. Put the program into production 7. Maintain the program | 1. Understand the problem 2. Develop and describe an algorithm 3. Test the algorithm with simple inputs 4. Translate the algorithm into Java 5. Compile and test your program |

Almost all development lifecycles, however, are circular. They always go back to the beginning point of understanding the problem that you’re trying to solve.

Text

Description automatically generated

My preferred development lifecycle is shown in the Big Java eText. The key differences are Steps 2, 3, and 4 above. Instead of “plan the logic, code the program, compile the program”, you develop and test the algorithm before you even start writing code. Obviously, the list from Big Java is missing the parts about putting the software into production and maintaining the software, which are also important parts of the development life cycle.

1. Understand the problem
   1. Ask yourself, what are you actually trying to solve with your software?
2. Develop and describe an algorithm
   1. Put into words the steps that you need to pass through to solve your problem. When you write it out in simple language does it make sense?
3. Test the algorithm with simple inputs
   1. Using mock data or information, make sure the algorithm you’ve created to solve the problem really does solve the problem. Make sure you aren’t missing any steps or leaving out any information. Do you need to reorder the content to achieve a better result? Can you simplify and still achieve the same result?
4. Translate the algorithm into Java
   1. Write the code that takes your text-based algorithm and turns it into the program that will resolve your problem.
5. Compile and test your program
   1. Make sure it compiles, fix any errors. TEST your program. Does it do what you want it to do, does it provide feedback if the wrong information is provided, is there any missing functionality?
6. Productize and maintain your code
   1. One of the reasons I find the Big Java software development process more appealing is because Steps 6 and 7 from Brightspace are often outside the purview of a developer to implement. In the real world of software development productizing and maintaining software is controlled outside of the developers’ hands, usually by middle or upper management levels that must weigh the cost-benefit ratio for productizing and maintenance.

## References

### Software

Eclipse Foundation. (n.d.). Eclipse IDE for Java Developers, Version (2021-12(4.22.0). *Eclipse.org*. Retrieved January 20, 2022, from <https://www.eclipse.org/> .

Amazon. (n.d.). Amazon Corretto Java Development Kit, Version (11.0.13.8.1). *Amazon Corretto*. Amazon. Retrieved January 20, 2022, from <https://aws.amazon.com/corretto/>

### Books

HORSTMANN, C. A. Y. (2020). Big java: Early objects. JOHN WILEY.

### Websites

Intro to Computer Programming CST8116. <https://brightspace.algonquincollege.com/d2l/le/content/409412/Home>

Free Java Guide & Tutorials. (n.d.). History of Java programming language. Retrieved January 20, 2022, from <https://freejavaguide.com/history.html>

*A computer science portal for geeks*. GeeksforGeeks. (n.d.). Retrieved January 20, 2022, from https://www.geeksforgeeks.org/

Bathia, S. (2021, January 8). *What is procedural programming? [definition] - key features*. Hackr.io. Retrieved January 20, 2022, from <https://hackr.io/blog/procedural-programming>

1. <https://www.geeksforgeeks.org/java-applet-basics/> [↑](#footnote-ref-1)
2. <https://www.geeksforgeeks.org/introduction-java-servlets/> [↑](#footnote-ref-2)
3. <https://hackr.io/blog/procedural-programming> [↑](#footnote-ref-3)