JAVA

CORE

CyberxNuke
DAY 1

JAVA
CONTENTS

- Basic Intro. To Java
- JDK, JRE, JVM
- Data types
- Variables
- Operators
- Loops (all types of loops)
- Coding Standards

JAVA

INTRODUCTION TO JAVA

- Java is an object oriented programming language (OOP).
- It is write once - use anywhere type of programming language.
- Object Oriented Programming Concepts:
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance

JDK, JRE, JVM

- **JDK**: Java Development Kit. It comprises of the development tools, the compiler and JRE.
- **JRE**: Java Runtime Environment. It comprises of the library classes and JVM.
- **JVM**: Java Virtual Machine. It is an interpreter and platform dependent. It converts the .class (bytecode) generated by the java compiler to machine language (binary).
- **JIT**: Just In Time Compiler. It compiles the frequently executed code (hot spots) during run time. This leads to substantial performance gains in execution.

VARIABLES & DATA TYPES

- Variable is a container to store data. Every variable is assigned memory according to it’s data type.
- Variable Types:
  - **Static**: A static variable can be accessed without creating the instance of a class. It is allocated memory only once.
  - **Instance**: An instance variable is accessible through an object/instance of a class. It is unique to that object.
  - **Local**: A local variable can be used inside the method where it is declared. It cannot be accessed outside its scope.

VARIABLES & DATA TYPES

- Variable is a container to store data. Every variable is assigned memory according to it’s data type.
- Primitive Data Types. They store the value:

int (4 Bytes)	**double (8 Bytes)**
short (2 Bytes)	**char (2 Bytes)**
long (8 Bytes)	**boolean (1 Byte)**
float (4 Bytes)	**Byte (1 Byte)**

VARIABLES & DATA TYPES

- Variable is a container to store data. Every variable is assigned memory according to it’s data type.
- Non-Primitive Data Types. They don’t store the value but they store the reference (address) to the value:

String
Arrays
Class
Interface

OPERATORS

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators

OPERATORS

ARITHMETIC OPERATORS

- + (**Addition**)
- - (**Subtraction**)
- * (**Multiplication**)
- / (**Division**)
- % (**Remainder**)
- ++ (**Increment**)
- - - (**Decrement**)

OPERATORS

RELATIONAL OPERATORS

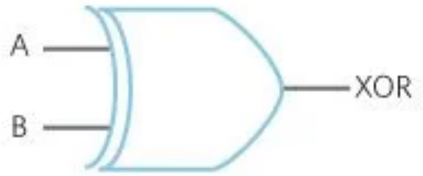
- < (**Less than**)
- > (**Greater Than**)
- <= (**Less than or equal to**)
- >= (**Greater than or equal to**)
- != (**Not equal to**)
- == (**equal to**)

CyberxNuke

OPERATORS
BITWISE OPERATORS

- & (**bitwise and**)
- | (**bitwise or**)
- ^ (**bitwise xor**)
- ~ (**bitwise compliment**)
- << (**Binary Left Shift**)
- >> (**Binary Right Shift**)
- >>> (**Shift right zero fill - unsigned**)
- >>> will always put a 0 in the left most bit, while >> will put a 1 or a 0 depending on what the sign of it is.

$$X = A \oplus B$$



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

OPERATORS

LOGICAL OPERATORS

- && (**Logical AND**)
- || (**Logical OR**)
- ! (**Logical NOT**)

OPERATORS

ASSIGNMENT OPERATORS

- = (**Assignment**)
- += (**Short Hand Addition**)
- -= (**Short Hand Subtraction**)
- *= (**Short Hand Multiplication**)
- /= (**Short Hand Division**)
- %= (**Short Hand Remainder**)
- &= (**Bitwise AND assignment**)
- |= (**Bitwise OR assignment**)
- ^= (**Bitwise XOR or exclusive OR assignment**)
- <<= (**Left Shift assignment**)
- >>= (**Right Shift assignment**)

OPERATORS

MISCELLANEOUS OPERATORS

- ? (**Conditional Operator** or **Ternary Operator**)
  - Used to evaluate boolean expressions.
  - **Example**
    - (3 > 2) ? True : False

LOOPS
ENTRY CONTROLLED

- An entry controlled loop checks the condition before executing the body of the loop.
- **Example**: for, while

```
for(;i<10; i++) {
  System.out.println(i);
}
```

```
int i = 0;
while(i < 10) {
    System.out.println(i);
    i++;
}
```

LOOPS
CONTINUE KEYWORD

- Continue keyword skips the loop and continues with next iteration in the loop.
- **Example**: continue

```
first:for(int x = 0; x < 10; x++) {

      for(int y = 0; y < 1; y++) {
        if ((x % 2) == 0) {
          continue first;
        }

        System.out.println("Numbers: " + x);

      }

    }
```

LOOPS
EXIT CONTROLLED

- An exit controlled loop checks the condition after executing the body of the loop. So, it is guaranteed to execute at least once.
- **Example**: do while.

```
int i = 11;
    do {
      System.out.println(i);
      i++;
    } while(i < 10);
```

- **Output**: 11

LOOPS
FOR EACH

- For-each loop uses a loop variable to iterate over a collection like array, ArrayList etc.

- **Example**: for-each.

```
int[] arr = {1,2,3,4,5};
    for(int elem: arr) {

    System.out.println(elem);
    }
```

IF-ELSE
CONDITION

- If-else condition is used to perform an action based on the condition. Conditional operators can be used in conjunction with operands as conditions.

- **Example**: if-else.

```
if (3>1) {
       System.out.println("True!");
    } else {
       System.out.println("Not true!");
    }
```

SWITCH CASE
CONDITION

- Switch case can be used to perform an action based on the given condition.
- **Example**: switch.

```
switch(1){
      case 1:
        System.out.print("TRUE");
        break;
      case 2:
        System.out.print("FALSE");
      default:
        break;
  }
```

CODING STANDARDS

- Class and interface names should be in Camel Case. Avoid acronyms/ abbreviations.
- Use meaningful variable names.
- Don’t declare or execute multiple statements in the same line.
- Use getters, setters (**getX()**, **setX()**) to assign values to the variables. Set the access modifier of the variables to private.

Reference https://medium.com/@rhamedy/a-short-summary-of-java-coding-best-practices-31283d0167d3