Concepts of Programming in Java

- Akshita Chanchlani

Trainer Introduction

- **Name**: Mrs.Akshita S.Chanchlani
- **Designation** : Technical Trainer
- **Education** :
  - PhD (Pursuing) : Computer Science and Engineering
  - Masters of Engineering (ME) in Information Technology
  - B.Tech in Computer Engineering, From VJTI Mumbai
- **Training Experience**
  - PreCAT Batches at Sunbeam
  - C, C++ , core java, python
- **Professional Experience**
  - **11+ years**
- **Email** : akshita.chanchlani@sunbeaminfo.com

Documentation

- **JDK download:**
  - https://adoptopenjdk.net/
- **Spring Tool Suite 3 download:**
  - https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3
- **Oracle Tutorial:**
  - https://docs.oracle.com/javase/tutorial/
- **Java 8 API Documentation:**
  - https://docs.oracle.com/javase/8/docs/api/
- **MySQL Connector:**
  - https://downloads.mysql.com/archives/c-j/
- **Core Java Tutorials:**
  1. http://tutorials.jenkov.com/java/index.html
  2. https://www.baeldung.com/java-tutorial
  3. https://www.journaldev.com/7153/core-java-tutorial

Java History

- **James Gosling, Mike Sheridan** and **Patrick Naughton** initiated the Java language project in June 1991.
- **Java** was originally **developed by James Gosling** at **Sun Microsystems** and **released in 1995.**
- The language was initially called **Oak** after an oak tree that stood outside Gosling's office.
- Later the project went by the name ***Green*** and was finally renamed ***Java***, from Java coffee, a type of coffee from Indonesia.
- Gosling and his team did a brainstorm session and after the session, they came up with several names such as **JAVA, DNA, SILK, RUBY, etc.**
- Sun Microsystems released the first public implementation as Java 1.0 in 1996.

Java History

- The Java programming language is a general-purpose, concurrent, class-based, object-oriented language.
- The Java programming language is a high-level language.
- The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included.
- **It is intended to be a production language, not a research language.**
- The Java programming language is statically typed.
- It promised **Write Once, Run Anywhere** (**WORA**) functionality.
- Standardizing Java Language Specification(JLS) is a job of Java Community Process(**JCP**).

Version History

Version	Date
JDK Beta	1995
JDK1.0	January 23, 1996[39]
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11, 2006
Java SE 7	July 28, 2011
Java SE 8	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11	September 25, 2018[40]
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020
Java SE 15	September 15, 2020[41]
Java SE 16	March 16, 2021

- The first version was released on January 23, 1996.

- The acquisition of Sun Microsystems by Oracle Corporation was completed on January 27, 2010

- As of September 2020, Java 8 and 11 are supported as Long Term Support (LTS) versions

- In September 2017, Mark Reinhold, chief Architect of the Java Platform, proposed to change the release train to "one feature release every six months".

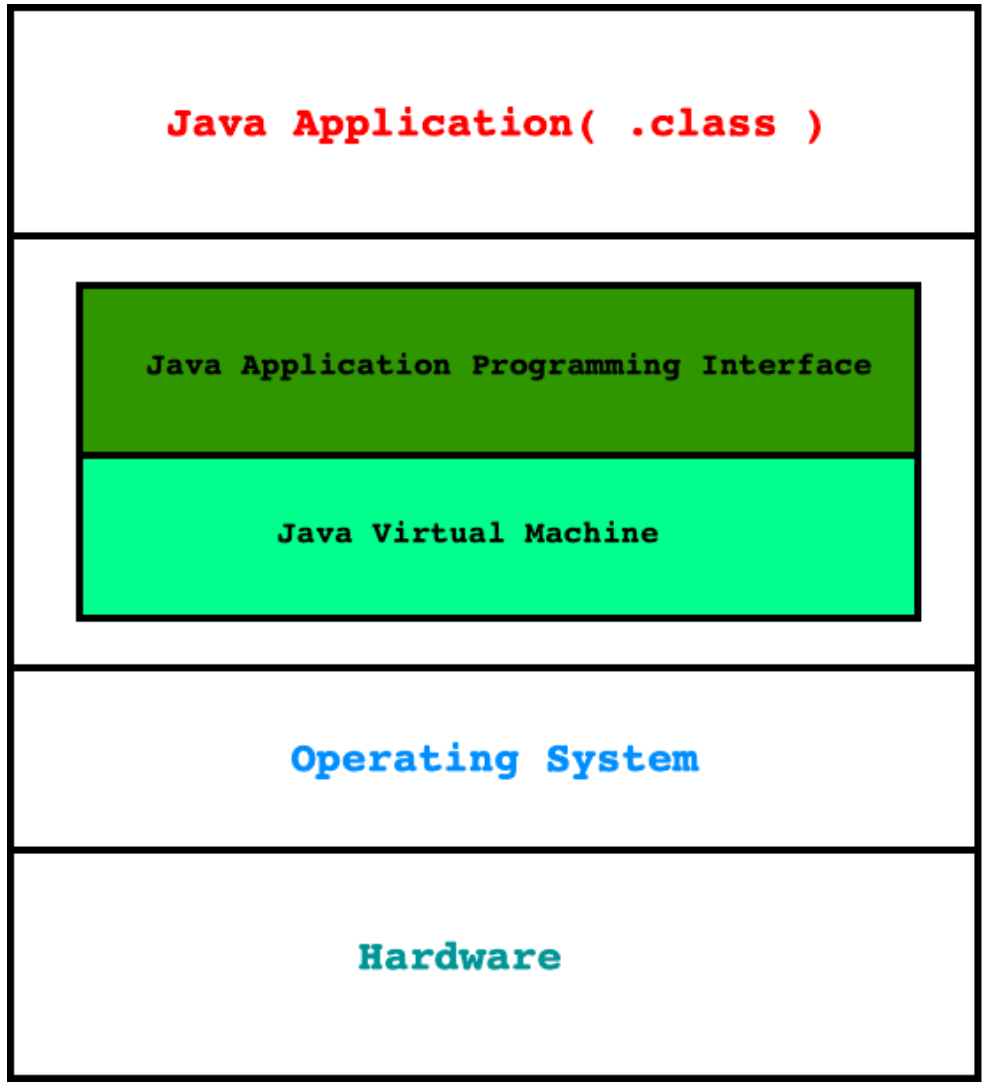
- OpenJDK (Open Java Development Kit) is a free and open source implementation of the (Java SE). It is the result of an effort Sun Microsystems began in 2006.

- Java Language and Virtual Machine Specifications: https://docs.oracle.com/javase/specs/

Java Platforms

1. **Java SE**
   - Java Platform Standard Edition.
   - It is also called as Core Java.
   - For general purpose use on Desktop PC's, servers and similar devices.
2. **Java EE**
   - Java Platform Enterprise Edition.
   - It is also called as advanced Java / enterprise java / web java.
   - Java SE plus various API's which are useful client-server applications.
3. **Java ME**
   - Java Platform Micro Edition.
   - Specifies several different sets of libraries for devices with limited storage, display, and power capacities.
   - It is often used to develop applications for mobile devices, PDAs, TV set-top boxes and printers.
4. **Java Card**
   - A technology that allows small Java-based applications (applets) to be run securely on smart cards and similar small-memory devices.

Components of the Java platform



Java Platform
[Software Platform]

[Hardware Based Platform]

SDK, JDK, JRE, JVM

- **SDK** = Development Tools + Documentation + Libraries + Runtime Environment.

- **JDK** = Java Development Tools + Java Docs + **rt.jar** + JVM.
  - JDK : Java Development Kit.
  - It is a software, that need to be install on developers machine.
  - We can download it from oracle official website.

- **JDK** = Java Development Tools + Java Docs + **JRE[rt.jar + JVM**].
  - JRE : Java Runtime Environment.
  - rt.jar and JVM are integrated part of JRE.
  - JRE is a software which comes with JDK. We can also download it separately.
  - To deploy application, we should install it on client's machine.

- rt.jar file contains core Java API in **compiled form**.

- **JVM** : An engine, which manages execution of Java application. (also called as Execution Engine)

C++ versus Java terminology

C++ Terminology	Java Terminology
Class	Class
Data Member	Field
Member Function	Method
Object	Instance
Pointer	Reference
Access Specifier	Access Modifier
Base Class	Super Class
Derived Class	Sub Class
Derived From	Extended From
Runtime Polymorphism	Dynamic Method Dispatch

Simple Hello Application

```
//File Name : Program.java
class Program{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

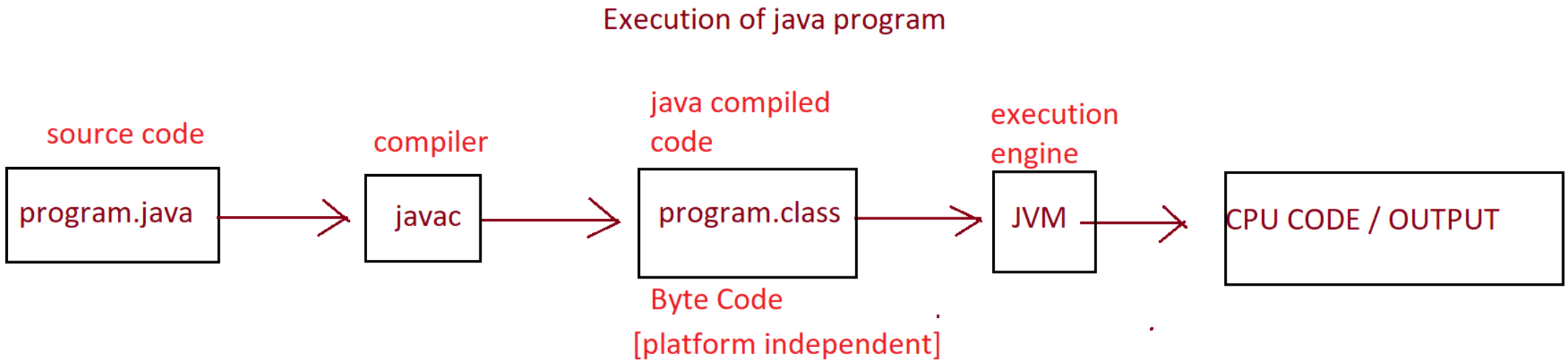
```
Compilation=> javac Program.java  //Output : Program.class
Execution=>   java Program
```

```
System    :   Final class declared in java.lang package
out       :   public static final field of System class. Type of out is PrintStream
println   :   Non static method of java.io.PrintStream class
```

To view class File :

javap -c Program.class

Java application flow of execution



Comments

- Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program.
- Types of Comments:
  - **Implementation Comment**
    - Implementation comments are those found in C++, which are delimited by /*...*/, and //.
      1. Single-Line Comment
      2. Block Comment(also called as multiline comment)
  - **Documentation Comment**
    - Documentation comments (known as "doc comments") are Java-only, and are delimited by **/**...*/**.
    - Doc comments can be extracted to HTML files using the javadoc tool.

Entry point method

- Syntax:
  1. **public static void main(String[] args)**
  2. **public static void main(String... args)**

- Java compiler do not check/invoke main method. JVM invoke main method.

- When we start execution of Java application then JVM starts execution of two threads:
  1. Main thread : responsible for invoking main method.
  2. Garbage Collector : responsible for deallocating memory of unused object.

- We can overload main method in Java.

- We can define main method per class. But only one main method can be considered as entry point method.

Data Types

- Data type of any variable decide following things:
  1. **Memory:** How much memory is required to store the data.
  2. **Nature:** Which kind of data is allowed to store inside memory.
  3. **Operation:** Which operations are allowed to perform on the data stored in memory.
  4. **Range:** Set of values that we can store inside memory.

- The Java programming language is a statically typed language, which means that every variable and every expression has a type that is known at compile time.
  - Types of data type:
    1. **Primitive type(also called as value type)**
       - **boolean** type
       - Numeric type
         1. Integral types(**byte, char, short, int, long**)
         2. Floating point types(**float, double**
    2. **Non primitive type(also called as reference type)**
       - **Interface, Class, Type variable, Array**

Data Types

Sr.No.	Primitive Type	Size[In Bytes]	Default Value[For Field]	Wrapper Class
1	boolean	Isn't specified	FALSE	Boolean
2	byte	1	0	Byte
3	char	2	\u0000	Character
4	short	2	0	Short
5	int	4	0	Integer
6	float	4	0.0f	Float
7	double	8	0.0d	Double
8	long	8	0L	Long

Note : Char datatype supports UNICODE character set, so 2 bytes .

Variables

- The name of some **location of memory** used to hold a data value
- **Different types** of data require **different amounts** of memory. The compiler's job is to reserve sufficient memory
- Variables need to be declared once
- Variables are **assigned** values, and these values may be changed later
- Each variable has a type, and **operations can only be performed between compatible types**

Print Ascii value of any character

```
class Test
{
    public static void main(String args[])
    {
        char ch='A';
        System.out.println('a');
        System.out.println((int)'a');

    }
}
```

Formatting the output in java

```
class test
{
  public static void main(String args[])
  {
    int x = 100;
    System.out.printf("Printing simple integer: x = %d\n", x);

    // this will print it upto 2 decimal places
    System.out.printf("Formatted with precision: PI = %.2f\n", Math.PI);
    float n = 5.2f;

    // automatically appends zero to the rightmost part of decimal
    System.out.printf("Formatted to specific width: n = %.4f\n", n);
    n = 2324435.3f;
    // here number is formatted from right margin and occupies a
    // width of 20 characters
    System.out.printf("Formatted to right margin: n = %20.4f\n", n);
  }
}
```

Operators

- Java provides a rich set of operators to manipulate variables.
- We can divide all the Java operators into the following groups.

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

Arithmetic Operators

Operator	Description
+	Addition - Adds values on either side of the operator
-	Subtraction - Subtracts right hand operand from left hand operand
*	Multiplication - Multiplies values on either side of the operator
/	Division - Divides left hand operand by right hand operand
%	Modulus - Divides left hand operand by right hand operand and returns remainder
++	Increment - Increases the value of operand by 1
--	Decrement - Decreases the value of operand by 1

Relational Operator

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Bit Wise Operators

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; now in binary format they will be `a = 0011 1100` and `b = 0000 1101`.

```
a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a  = 1100 0011
```

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

Assignment Operator

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

Other Operators

- **Conditional Operator (? :) / Ternary Operator**

```
variable x = (expression) ? value if true : value if false
```

- **instanceOf Operator**

The operator checks whether the object is of a particular type(class type or interface type).
The **instanceOf** operator is wriiten as:

```
( Object reference variable ) instanceOf  (class/interface type)
```

Example :

```
String name = = 'Akshita';
boolean result = name instanceOf String;
// This will return true since name is type of String
```

Control Statements

Loops

Java has very flexible three looping mechanisms. You can use one of the following three loops:

- while Loop
- do...while Loop
- for Loop

Other Keywords

- Break
- continue

Decision Making

- If
- If..else
- Nested if else
- switch

Thank you.

akshita.Chanchlani@sunbeaminfo.com