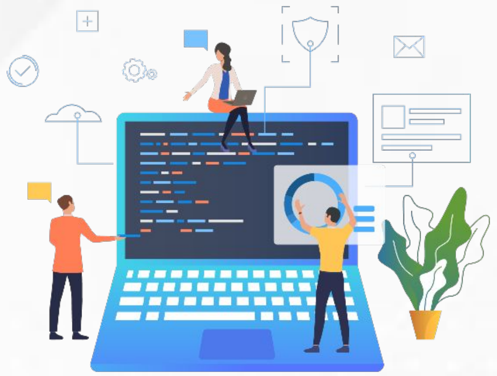
Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Object Oriented Programming-I



Prof. Swati R Sharma
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
swati.sharma@darshan.ac.in

Teaching and Examination Scheme

Teaching Scheme			Credits	Examination Marks				Total Marks
				Theory Marks		Practical Marks		
L	T	P	C	ESE (E)	PA (M)	ESE (V)	PA (I)	
4	0	2	5	70	30	30	20	150

Syllabus

Syllabus

Sr. No.	Content	Total Hrs
1	**Introduction to java and elementary programming:** Java language specification API, JDK and IDE, Creating, compiling and Executing a simple java program, Programming style, documentation and errors, Reading input from console, identifiers and variables, Assignment statements, Named constants and naming conventions, Data Types (Numeric, Boolean, Character, String) its Operations and Literals, Evaluating Expressions and operator Precedence, Types of Operators (Augmented assignment, Increment and Decrement, Logical), operator precedence and associativity, numeric type conversions.	04
2	**Selections , Mathematical functions and loops:** If statements, Two way, Nested if and multi-way if statements, Switch statements, Conditional Expressions, Common mathematical functions ,While , do-while and for loop, nested loops, Keyword break and continue.	04
3	**Methods and Arrays:** Defining and calling method, Passing argument by values, Overloading methods and scope of variables, Method abstraction and stepwise refinement, Single Dimensional arrays, copying arrays ,Passing and returning array from method, Searching and sorting arrays and the Array class, Two-Dimensional array and its processing, Passing Two-dimensional Array to methods, Multidimensional Arrays.	06

Syllabus (Cont.)

Sr. No.	Content	Total Hrs
4	**Objects and Classes:** Defining classes for objects, Constructors, accessing objects via reference variable, using classes from the java library, static variables, constants and methods, visibility modifiers and Data field encapsulation, passing objects to methods, array of objects, immutable objects and classes, scope of variable and the this reference.	04
5	**Object oriented thinking:** Class abstraction and Encapsulation, thinking in objects and class relationships, Primitive data type and wrapper class types, Big integer and Big decimal class, string class, String Builder and String Buffer class, super class and subclass, using super keyword, overriding and overloading methods, polymorphism and dynamic binding, casting objects and instanceof operator, The ArrayList class and its methods, The protected data and methods.	05
6	**Exception Handling, I/O, abstract classes and interfaces:** Exception types, finally clause, rethrowing Exceptions, chained exceptions, defining custom exception classes, file class and its input and output, Reading data from web, Abstract classes, interfaces, Comparable and Cloneabal interface.	04

Syllabus (Cont.)

Sr. No.	Content	Total Hrs
7	**JAVAFX basics and Event-driven programming and animations:** Basic structure of JAVAFX program, Panes, UI control and shapes, Property binding, the Color and the Font class, the Image and Image-View class, layout panes and shapes, Events and Events sources, Registering Handlers and Handling Events, Inner classes, anonymous inner class handlers, mouse and key events, listeners for observable objects, animation	05
8	**JAVAFX UI controls and multimedia:** Labeled and Label, button, Checkbox, RadioButton, Textfield, TextArea, Combo Box, ListView, Scrollbar, Slider, Video and Audio.	04
9	**Binary I/O ,Recursion and Generics:** Text I/O, binary I/O, Binary I/O classes, Object I/o, Random Access files, Problem solving using Recursion, Recursive Helper methods, Tail Recursion, Defining Generic classes and interfaces, Generic methods, Raw types and backward compatibility, wildcard Generic types, Erasure and Restrictions on Generics.	04
10	**List, Stacks, Queues and Priority Queues:** Collection, Iterators, Lists, The Comparator interface, static methods for list and collections, Vector and Stack classes, Queues and priority Queues.	04

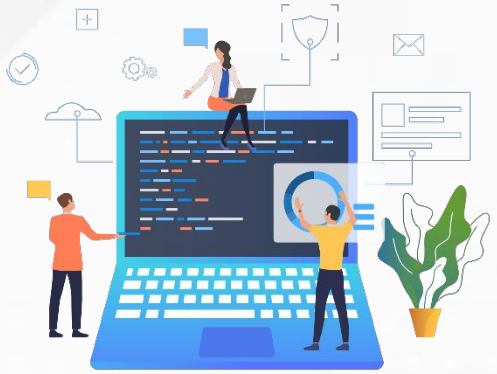
Syllabus (Cont.)

Sr. No.	Content	Total Hrs
11	**Sets and Maps:** Comparing the performance of Sets and Lists, singleton and unmodifiable collections and Maps.	02
12	**Concurrency:** Thread states and life cycle,Creating and Executing threads with the Executor Framework, Thread synchronization	02

Object-Oriented Programming
3140705

Darshan UNIVERSITY

Introduction to java and elementary programming



Prof. Swati R Sharma
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
swati.sharma@darshan.ac.in

Introduction

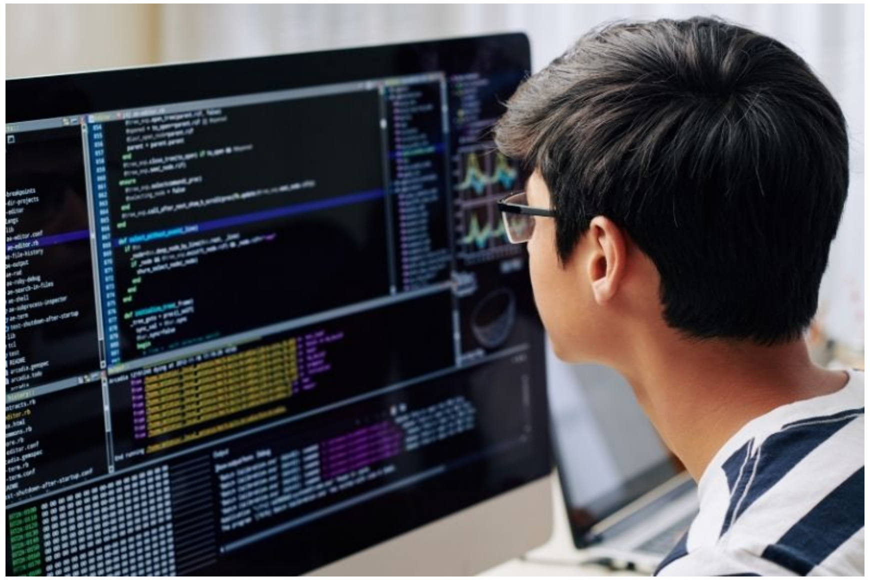
Welcome to the world of Programming

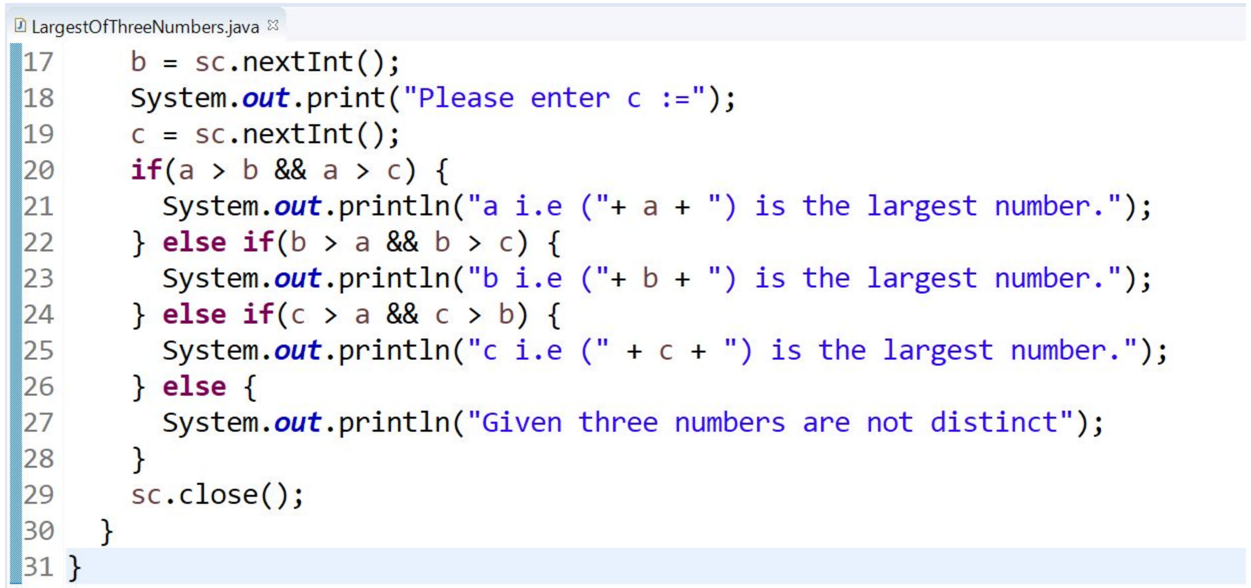
What is Language?

- A language is a system of communication used by a particular country or community.
- It is a structured system of communication used by humans, based on speech and gesture (spoken language), sign or often writing.
- Both person should understand each other’s language.

Programming Language

- A programming language is a computer language that is used by programmers (developers) to communicate with computers.
- It is a set of instructions written in a specific language to perform a specific task.
- It allow us to give instructions to a computer in a language the computer understands.
- 5000+ programming languages are there, notably used are approximate 250.
- E.g. C, C++, C#, Java, Python, PHP, Pascal, etc.



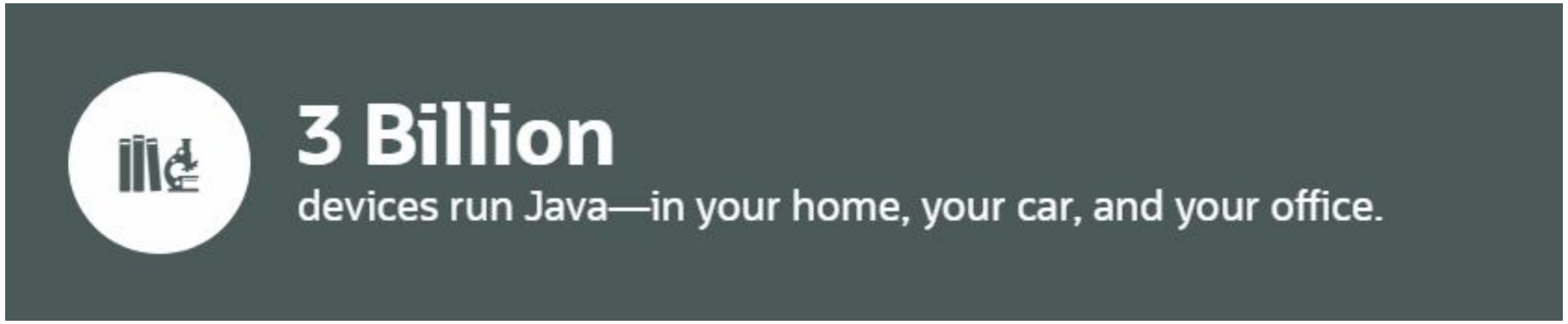


LargestOfThreeNumbers.java

```
    b = sc.nextInt();
    System.out.print("Please enter c :=");
    c = sc.nextInt();
    if(a > b && a > c) {
      System.out.println("a i.e ("+ a + ") is the largest number.");
    } else if(b > a && b > c) {
      System.out.println("b i.e ("+ b + ") is the largest number.");
    } else if(c > a && c > b) {
      System.out.println("c i.e (" + c + ") is the largest number.");
    } else {
      System.out.println("Given three numbers are not distinct");
    }
    sc.close();
  }
}
```

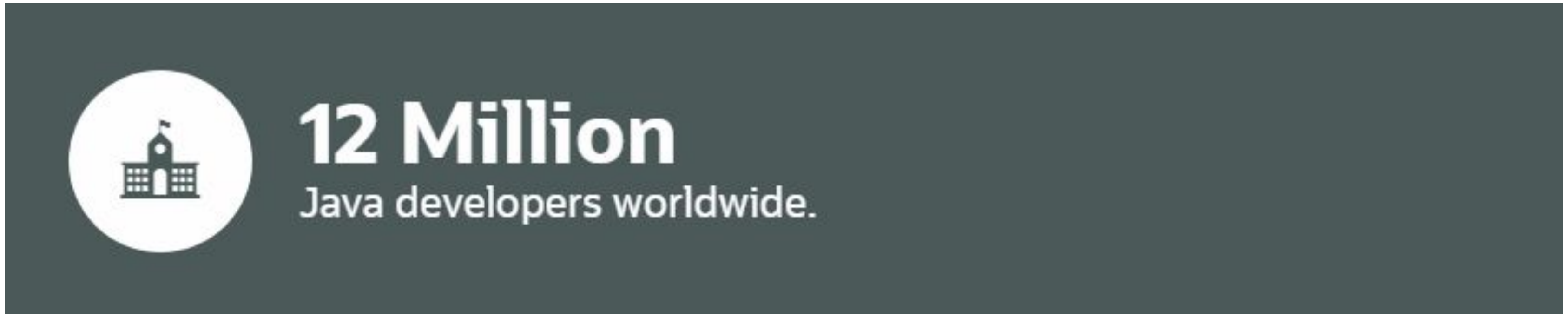
Types of Programming Languages

1. **Low-level programming language**
   - It is a machine-dependent (hardware specific) programming language.
   - It consists of a set of instructions that are either in the binary form (0 or 1) or in a symbolic and human-understandable mnemonics (ADD, MOV, SUB).
   - E.g. Machine level language, Assembly language, etc.
2. **High-level programming language**
   - It is closer to human languages than machine-level languages.
   - It is easy to read, write, and maintain as it is written in English like words.
   - It allows to write the programs which are independent of a particular type of machine (hardware).
   - A compiler is required to translate a high-level language into a low-level language.
   - E.g. Python, Java, JavaScript, PHP, C#, LISP, FORTRAN, etc.
3. **Middle-level programming language**
   - Middle-level programming language lies between the low-level and high-level programming language.
   - E.g. C, C++, etc.



3 Billion

devices run Java—in your home, your car, and your office.



12 Million

Java developers worldwide.

Introduction t

JAVA

- Java is a general-purpose computer-programming language that is open source, platform independent, object-oriented and specifically designed to have as few implementation dependencies as possible.
- Java was originally developed by **James Gosling** at Sun Microsystems and released in 1995.
- Java was initially named as Oak language and renamed to JAVA in 1995.

Current Version	**Java SE 17.0. 2** (as of September, 14th 2021)
Version we will use	Java SE 11 (LTS)
Setup size	149 MB (Linux), 152 MB (Windows x64)
Download Link	https://www.oracle.com/in/java/technologies/javase-jdk11-downloads.html
Official Website	https://java.com
Integrated **D**evelopment **E**nvironment (IDE)	1. Eclipse NetBeans 2. IntelliJ IDEA Community Edition 3. BlueJ

Features of JAVA

Simple

Simple: Java inherits C/C++ syntax and many object-oriented features of C++.

Object-oriented

Object Oriented: "Everything is an object" paradigm, which possess some state, behavior and all the operations are performed using these objects.

Robust

Robust: Java has a strong memory management system. It helps in eliminating error as it checks the code during compile and runtime.

Multithreaded

Multithreaded: Java supports multiple threads of execution, including a set of synchronization primitives. This makes programming with threads much easier.

Features of JAVA (Cont.)

Portable

Architectural Neutral: Java is platform independent which means that any application written on one platform can be easily ported to another platform.

Interpreted

Interpreted: Java is compiled to bytecodes, which are interpreted by a Java run-time environment.

High Performance

High Performance: Java achieves high performance through the use of bytecode which can be easily translated into native machine code. With the use of JIT (Just-In-Time) compilers, Java enables high performance.

Features of JAVA (Cont.)

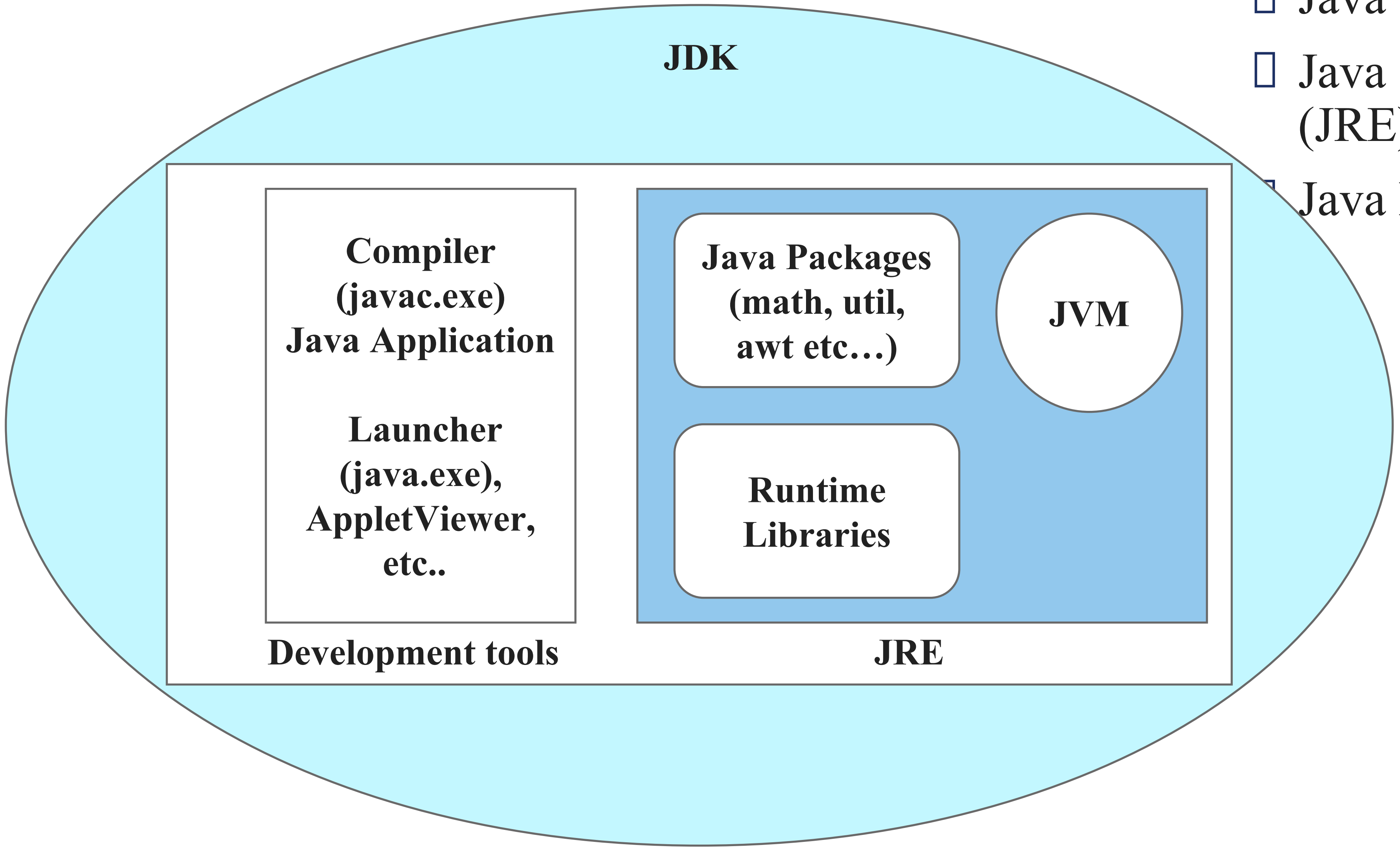
Distributed

Distributed: Java provides a feature which helps to create distributed applications. Using Remote Method Invocation (RMI), a program can invoke a method of another program across a network and get the output. You can access files by calling the methods from any machine on the internet.

Dynamic

Dynamic: Java has ability to adapt to an evolving environment which supports dynamic memory allocation due to which memory wastage is reduced and performance of the application is increased.

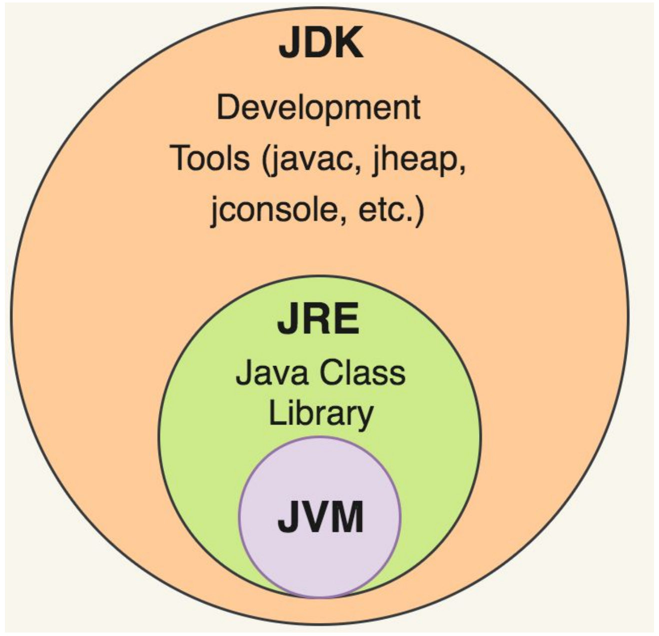
Components of Java



- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

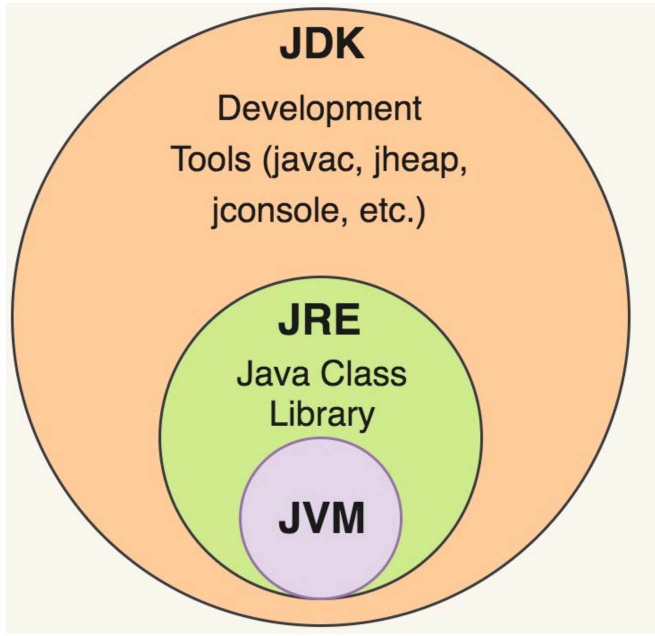
Java Development Kit (JDK)

- JDK contains tools needed ,
  - To develop the Java programs and
  - JRE to run the programs.
- The tools include
  - compiler (javac.exe),
  - Java application launcher (java.exe),
  - Appletviewer, etc…
- Java application launcher (java.exe) opens a JRE, loads the class, and invokes its main method.



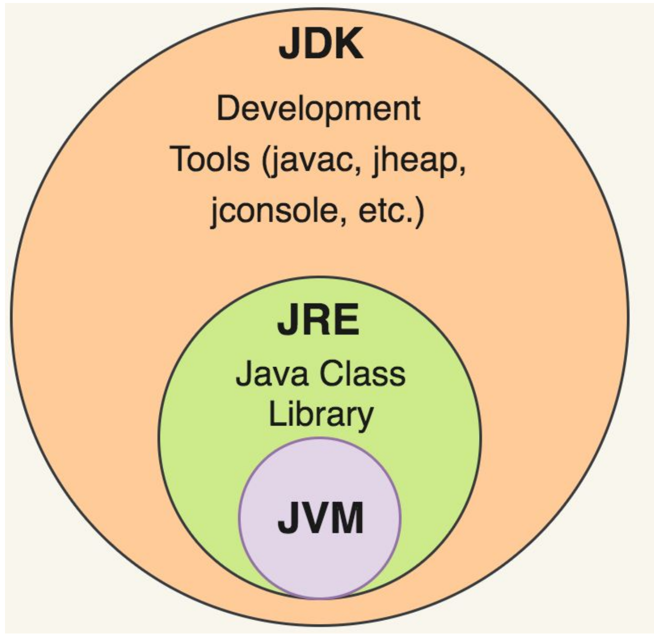
Java Runtime Environment (JRE)

- The JRE is required to run java applications.
- It combines the Java Virtual Machine (JVM), platform core classes and supporting libraries.
- JRE is part of the Java Development Kit (JDK), but can be downloaded separately.
- It does not contain any development tools such as compiler, debugger, etc.

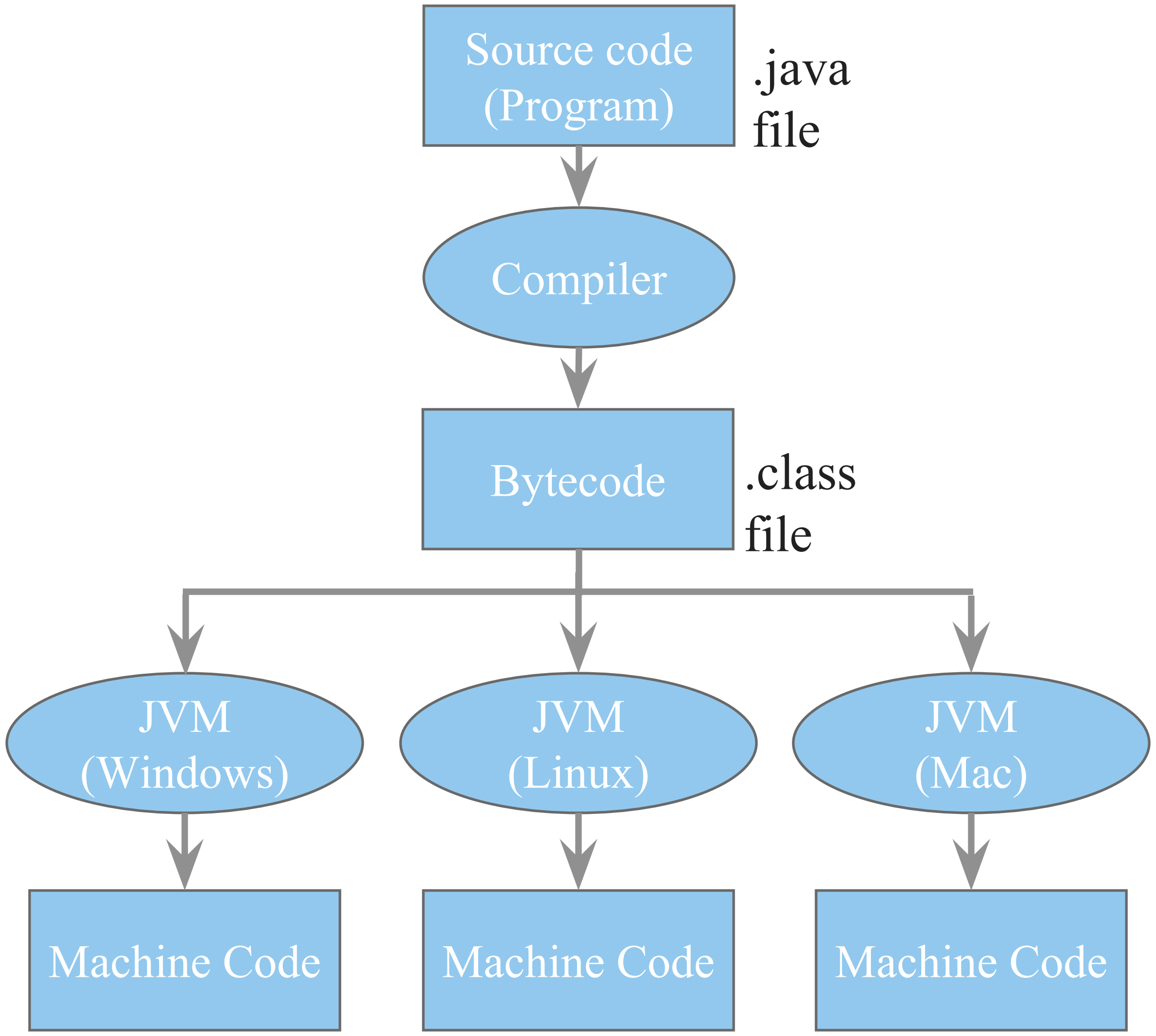


Java Virtual Machine (JVM)

- JVM is a virtual machine that enables a computer to run Java programs as well as programs written in other languages and compiled to Java Bytecode.
- Bytecode is a highly optimized set of instructions designed to be executed by the Java Virtual Machine(JVM).
- Byte code is intermediate representation of java source code.
- Java compiler provides byte code by compiling Java Source Code.
- Extension for java class file or byte code is '.class', which is platform independent.
- JVM is virtual because , It provides a machine interface that does not depend on the operating system and machine hardware architecture.
- JVM interprets the byte code into the machine code.
- **JVM** itself is **platform dependent**, but **Java** is **Not**.



How Java become Platform Independent?



Java Interview Question

1. Difference between JRE and JVM?
2. Difference between interpreter and JIT compiler?
3. Why Java is platform independent?
4. What are Java bytecodes?
5. JVM vs. JRE vs. JDK

Hello World Java Program

```
public class HelloWorld
{
  public static void main(String[] args)
  {
   System.out.println("Hello World");
  }
}
```

File must be saved as HelloWorld.java

Main method from where execution will start

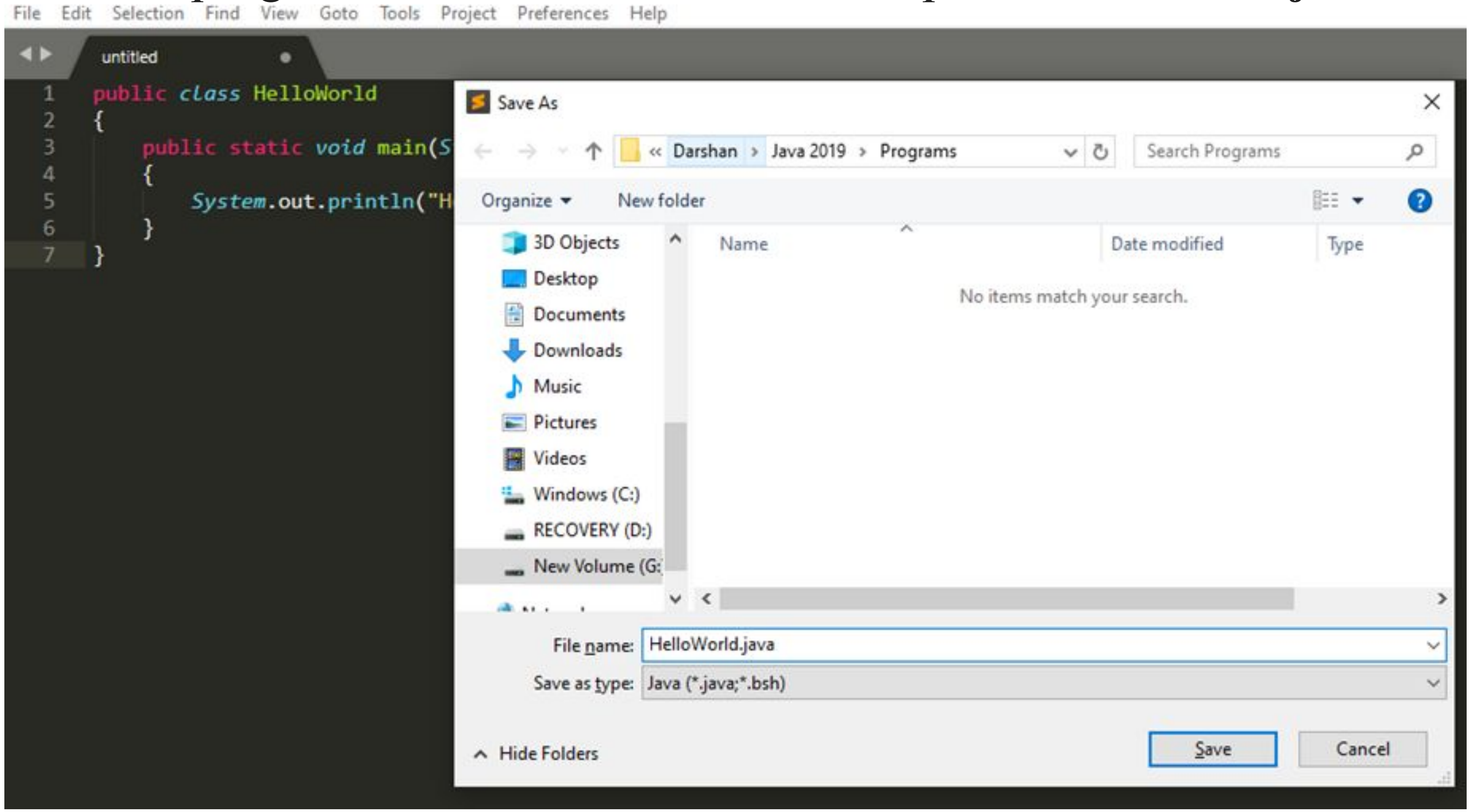
String must start with capital letter

System must start with capital letter

- We have to save this in HelloWorld.java file as it has public class named HelloWorld.
- String and System are inbuilt Java Classes.
- Classes in java are always written in Camel case.

How to execute Java Program?

1. Save the program with the same name as the public class with **.java** extension.



```
public class HelloWorld
{
    public static void main(S
    {
        System.out.println("H
    }
}
```

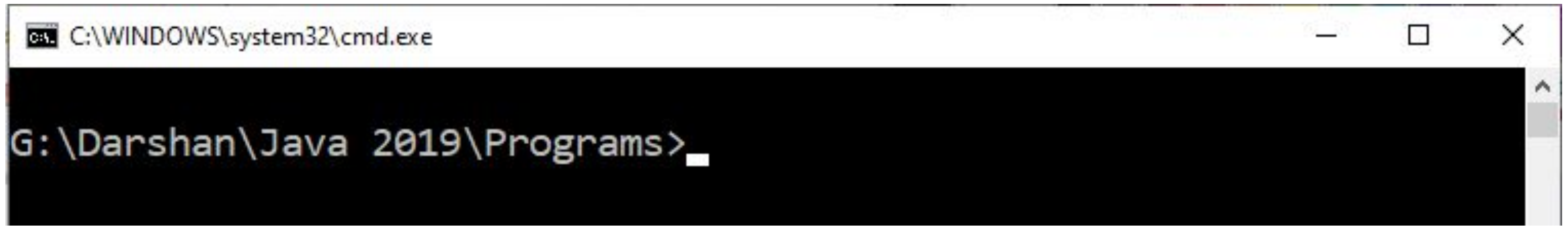
No items match your search.

File name: HelloWorld.java

Save as type: Java (*.java;*.bsh)

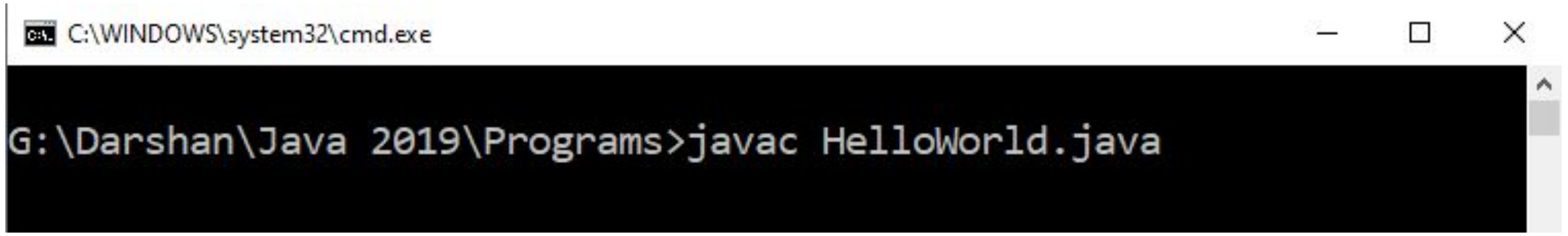
How to execute Java Program?

2. Open command prompt (cmd) / terminal & navigate to desired directory / folder.



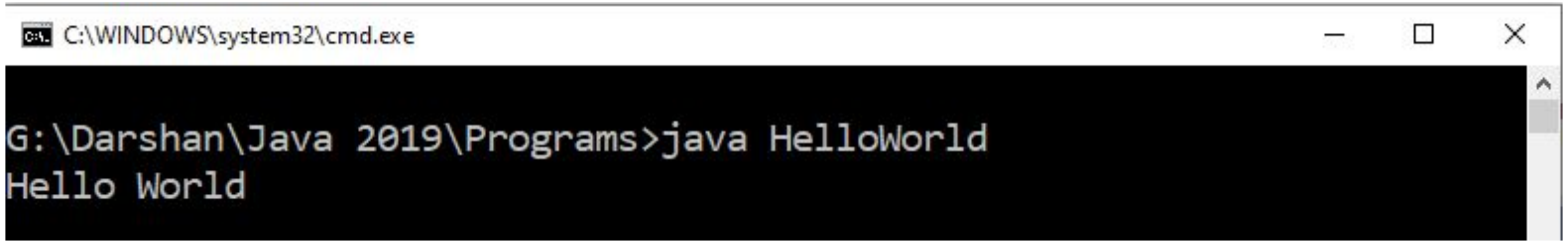
```
G:\Darshan\Java 2019\Programs>_
```

3. Compile the ".java" file with **javac** command.



```
G:\Darshan\Java 2019\Programs>javac HelloWorld.java
```

4. Execute the ".class" file with **java** command without extension.



```
G:\Darshan\Java 2019\Programs>java HelloWorld
Hello World
```

Tokens

Tokens

- The smallest individual unit of a language / program is known as a **token**.
- Tokens are basic building blocks of any language.

"Jane bakes tasty cookies."

- Jane is noun
- bakes is verb
- tasty is adjective
- cookies is noun
- '.' is special character to end the sentence.

- Each and every word and punctuation is a token.
- We divide sentence into tokens to understand the meaning of a sentence.
- Similarly, the compilers of programming language breaks a program into the tokens and proceeds to the various stages of the compilation.
- However, collection of tokens in appropriate sequence makes a meaningful sentence.

Classification of Tokens

Sr.	Token	Description	Examples
1	**Keywords**	Predefined reserved words	void, int, float, for, if
2	**Identifiers**	User-defined combination of alphanumeric characters. Name of a variable, function, class, etc.	a, i, sum, number, pi
3	**Constants**	Fixed values that do not change	17, -25.50, 82, 0
4	**Strings**	A sequence of characters	“Darshan”, “Hi!”
5	**Special Symbols**	Symbols that have special meaning	#, $, @, %, =, :, ;
6	**Operators**	A symbol that performs operation on a value or a variable	+, -, *, /

Darshan UNIVERSITY

Identifiers

Identifiers

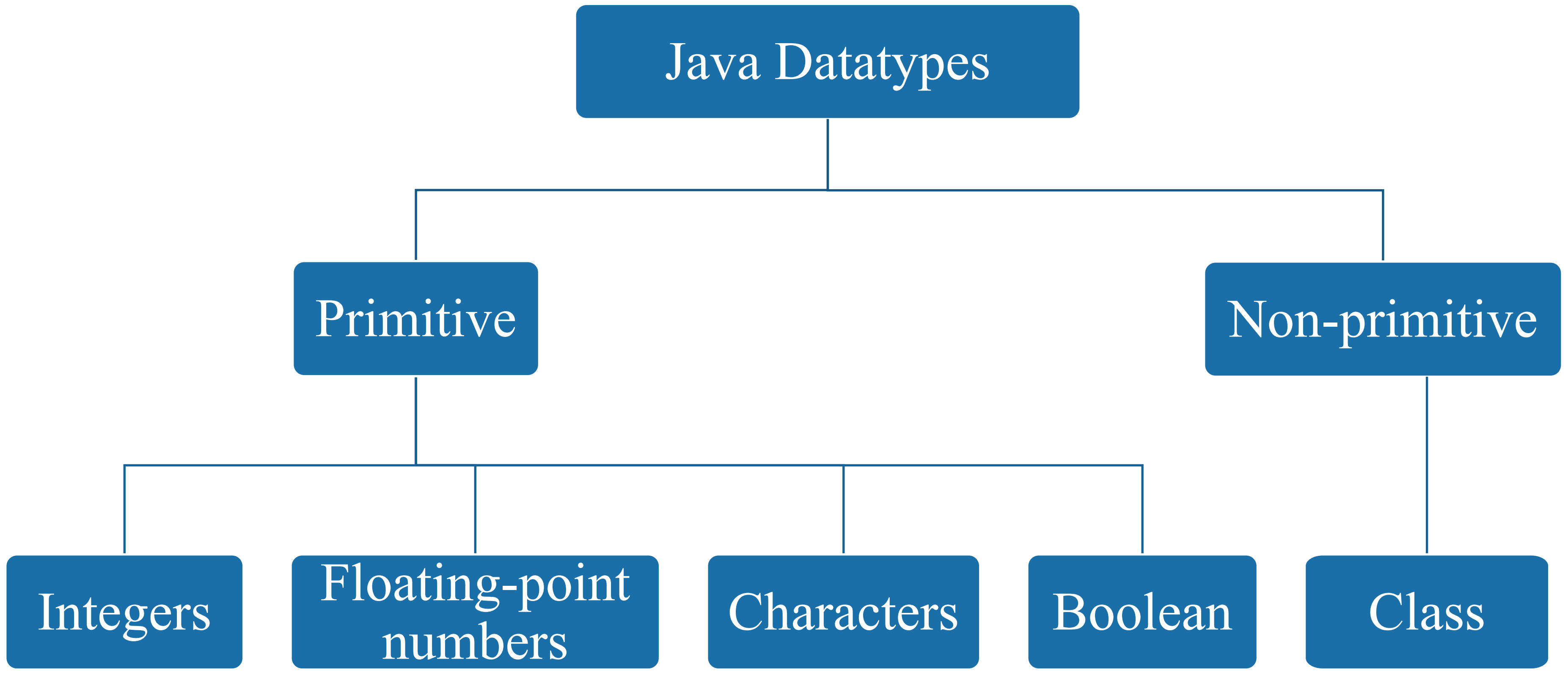
- They are used for class names, method names and variable names.
- An identifier may be any descriptive sequence of
  - uppercase(A…Z) and lowercase(a..z) letters
  - Numbers(0..9)
  - Underscore(_) and dollar-sign($) characters
- Examples for valid Identifiers,
  - AvgTemp
  - count
  - a4
  - $test
  - this_is_ok
- Examples for invalid Identifiers,
  - 2count (Identifiers can not start with digit)
  - High-temp (Identifiers can not contain dash)
  - Ok/NotOK (Identifiers can not contains slash)

Darshan UNIVERSITY

Identifier Name Valid or Invalid?

🟥 for	🟩 Rajkot	🟩 _Name	🟩 If	🟥 Roll Number	🟥 Rs.
🟩 __________a	🟥 int	🟩 student	🟩 _7	🟩 Identifier	🟩 Valid
🟩 C	🟥 C++	🟩 Java	🟥 C#	🟩 Compiler	🟩 ____8___a___8
🟥 7Student	🟥 Who?	🟥 v.a.l.u.	🟩 A_B_C_D_E	🟩 i	🟥 if
🟩 Int	🟩 A2020	🟥 2021	🟩 sum	🟥 a,b,c	🟥 i;

Data Types


Java Datatypes
Primitive
Non-primitive
Integers
Floating-point numbers
Characters
Boolean
Class

Primitive Data Types

Data Type	Size	Range	Example
byte	1 Byte	-128 to 127	byte a = 10;
short	2 Bytes	-32,768 to 32,767	short a = 200;
int	4 Bytes	-2,147,483,648 to 2,147,483,647	int a = 50000;
long	8 Bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	long a = 20;
float	4 Bytes	1.4e-045 to 3.4e+038	float a = 10.2f;
double	8 Bytes	4.9e-324 to 1.8e+308	double a = 10.2;
char	2 Bytes	0 to 65536 (Stores ASCII of character)	char a = ‘a’;
boolean	Not defined	true or false	boolean a = true;

Escape Sequences

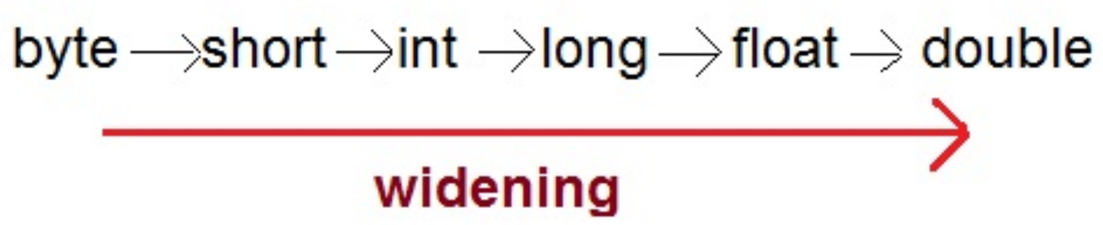
- Escape sequences in general are used to signal an alternative interpretation of a series of characters.
- For example, if you want to put quotes within quotes you must use the escape sequence, \", on the interior quotes.

```
System.out.println("Good Morning \"World\"");
```

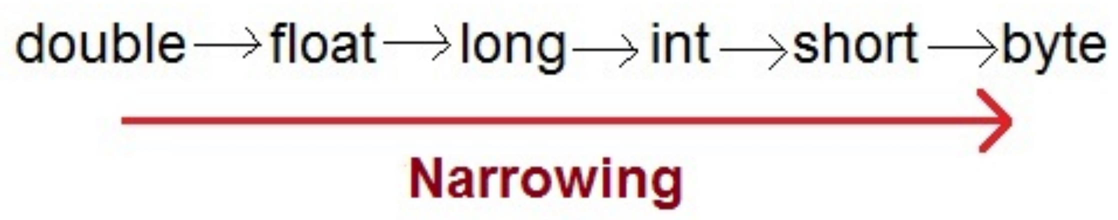
Escape Sequence	Description
\’	Single quote
\”	Double quote
\\	Backslash
\r	Carriage return
\n	New Line
\t	Tab

Type Casting

- Assigning a value of one type to a variable of another type is known as Type Casting.
- In Java, type casting is classified into two types,
  - Widening/Automatic Type Casting (Implicit)



  - Narrowing Type Casting(Explicitly done)



Automatic Type Casting

- When one type of data is assigned to other type of variable , an *automatic type conversion* will take place if the following two conditions are satisfied:
  - The two types are compatible
  - The destination type is larger than the source type
- Such type of casting is called "*widening conversion*".
- Example:

  int can always hold values of **byte** and **short**

```
public static void main(String[] args) {
    byte b = 5;
    // √ this is correct
    int a = b;
}
```

Casting Incompatible Types

- To create a conversion between two incompatible types, you must use a *cast*
- A ***cast*** is an explicit type conversion.
- Such type is called "*narrowing conversion*".
- Syntax:

  (target-type) value

- Example:

```
public static void main(String[] args) {
    int a = 5;
    // × this is not correct
    byte b = a;
    // √ this is correct
    byte b = (byte)a ;
}
```

Operator

Perform definite operation

Operators

1. Arithmetic Operators
2. Relational Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Conditional / Ternary Operator
7. Instance of Operator

Operators

- An operator is a symbol to perform specific mathematical or logical functions.
- We use operators in maths to perform certain operations, e.g. +, -, *, /, etc.
- Unary operators (++, --) take one operand, Binary operators (+, -, *, /) take two operands.
- Programming languages are rich in operators which can be divided in following categories,

Sr.	Operator	Examples
1	Arithmetic Operators	+, -, *, /, %
2	Relational Operators	<, <=, >, >=, ==, !=
3	Logical Operators	&&, ||, !
4	Assignment Operators	=, +=, -=, *=, /=
5	Increment and Decrement Operators	++, --
6	Conditional Operator	?:
7	Bitwise Operators	&, |, ^, <<, >>

Arithmetic Operators

- An arithmetic operator performs basic mathematical calculations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning	Example	Description
+	Addition	a + b	Addition of a and b
-	Subtraction	a – b	Subtraction of b from a
*	Multiplication	a * b	Multiplication of a and b
/	Division	a / b	Division of a by b
%	Modulo division- remainder	a % b	Modulo of a by b

Relational Operators

- A relational operators are used to compare two values.
- They check the relationship between two operands, if the relation is true, it returns 1; if the relation is false, it returns value 0.
- Relational expressions are used in decision statements such as if, for, while, etc…

Operator	Meaning	Example	Description
<	is less than	a < b	a is less than b
<=	is less than or equal to	a <= b	a is less than or equal to b
>	is greater than	a > b	a is greater than b
>=	is greater than or equal to	a >= b	a is greater than or equal to b
==	is equal to	a == b	a is equal to b
!=	is not equal to	a != b	a is not equal to b

Relational Operators

Note : A = 10 & B = 20

Operator	Description	Example
==	Equals	(A == B) is not true.
!=	Not Equals	(A != B) is true.
>	Greater than	(A > B) is not true.
<	Less than	(A < B) is true.
>=	Greater than equals	(A >= B) is not true.
<=	Less than equals	(A <= B) is true.

Bitwise Operators

Note : A = 60 & B = 13

Operator	Description	Example
&	Binary AND Operator	A & B = 12 which is 0000 1100
|	Binary OR Operator	A | B = 61 which is 0011 1101
^	Binary XOR Operator	A ^ B = 49 which is 0011 0001
~	Binary Ones Complement Operator	~A = -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator	A << 2 = 240 which is 1111 0000
>>	Binary Right Shift Operator.	A >> 2 = 15 which is 1111
>>>	Shift right zero fill operator.	A >>>2 = 15 which is 0000 1111

Logical Operators

- Logical operators are decision making operators.
- They are used to combine two expressions and make decisions.
- An expression containing logical operator returns either 0 or 1 depending upon whether expression results false or true.

Operato r	Meaning	Example (Let’s assume c=5 and d=2)
&&	Logical AND. True only if all operands are true	expression ((c==5) && (d>5)) equals to 0.
||	Logical OR. True only if either one operand is true	expression ((c==5) || (d>5)) equals to 1.
!	Logical NOT. True c	n !(c==5) equals to 0.

a	b	a && b	a || b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Logical Operators

Note : A = true & B = false

Operator	Description	Example
&&	Logical AND operator	(A && B) is false.
||	Called Logical OR Operator	(A || B) is true.
!	Called Logical NOT Operator	!(A && B) is true.

Assignment Operators

- Assignment operators are used to assign a new value to the variable.
- The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value or a result of an expression.
- Meaning of = in Maths and Programming is different.
  - Value of LHS & RHS is always same in Math.
  - In programming, value of RHS is assigned to the LHS

Operator	Meaning	
=	Assigns value of right side to left side. Suppose a=5 and b=10. a=b means now value of a is 10.	
+=	a += 1 is same as a = a + 1	Shorthand Assignment Operators
-=	a -= 5 is same as a = a – 5	
*=	a *= b is same as a = a * b	
/=	a /= c is same as a = a / c	
%=	a %= 10 is same as a = a % 10	

Assignment Operators

Operator	Description	Example
=	Simple assignment operator	C = A + B will assign value of A + B into C
+=	Add AND assignment operator	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator	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	C |= 2 is same as C = C | 2

Darshan UNIVERSITY

Increment / Decrement Operators

- Increment and decrement operators are unary operators that add or subtract one, to or from their operand.
- the increment operator ++ increases the value of a variable by 1, e.g. a++ means a=a+1
- the decrement operator -- decreases the value of a variable by 1. e.g. a— means a=a–1
- If ++ operator is used as a prefix (++a) then the value of a is incremented by 1 first then it returns the value.
- If ++ operator is used as a postfix (a++) then the value of a is returned first then it increments value of a by 1.

Expression	Evaluation (Let's say a=10, c=15)
b = a++	Value of b would be 10 and value of a would be 11.
b = ++a	Value of b & a would be 11.
b = a--	Value of b would be 10 and value of a would be 9.
b = --a	Value of b & a would be 9.

Expression	Evaluation (Let's say a=10, c=15)
b = --a + c++	b = 24
b = a++ + ++c	b = 26
b = ++a - ++c	b = –5

Conditional Operator (Ternary)

- Conditional Operator (? :)
  - Syntax:

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

```
b = (a == 1) ? 20 : 30;
```

Operator Precedence & Associativity

- How does java evaluate `1 + 10 * 9 ?`
  - `(1 + 10 ) * 9 = 99` **OR** `1 + (10 * 9) = 91`
- To get the correct answer for the given problem Java came up with Operator precedence. (multiplication have higher precedence than addition so correct answer will be **91** in this case)
- For Operator, associativity means that when the same operator appears in a row, then to which direction the expression will be evaluated.
- How does java evaluate `1 * 2 + 3 * 4 / 5 ???`

```
        2    +    12     / 5
        2  +  2.4
          4.4
```

Precedence of Java Operators

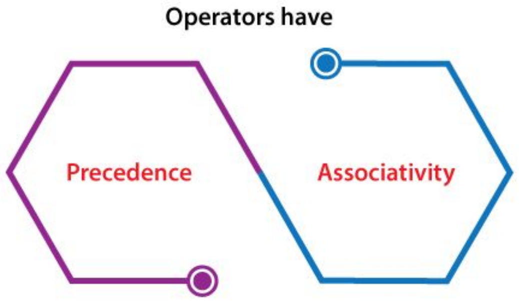
Category	Operator	Associativity
Postfix	() [] . (dot operator)	Left to right
Unary	++ - - ! ~	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	>> >>> <<	Left to right
Relational	> >= < <=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Operators Precedence & Associativity

Priority matters!

Operators Precedence & Associativity

- Operators Precedence and Associativity are two characteristics of operators that determine the evaluation order of sub-expressions in absence of brackets.
- Operator precedence determines which operation is performed first in an expression with more than one operators with different precedence.
  - a=10 + 20 * 30 is calculated as 10 + (20 * 30) and not as (10 + 20) * 30 so answer is 610.
- Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either Left to Right (L to R) or Right to Left (R to L).
  - a=100 / 10 * 10
    - If Left to Right means (100 / 10) * 10 then answer is 100
    - If Right to Left means 100 / (10 * 10) then answer is 1
    - Division (/) & Multiplication (*) are Left to Right associative so the answer is **100.**



Operators Precedence and Associativity

1) Associativity is only used when there are two or more operators of same precedence.

2) All operators with the same precedence have same associativity

Priority	Operator	Description	Associativity
1	( ) [ ] . -> a++ a—	Parentheses (function call) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement	left-to-right
2	++a —a + – ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) De-reference Address (of operand) Determine size in bytes on this implementation	**right-to-left**
3	* / %	Multiplication/division/modulus	left-to-right
4	+ –	Addition/subtraction	left-to-right
5	<< >>	Bitwise shift left, Bitwise shift right	left-to-right

Priority	Operator	Description	Associativity
7	== !=	is equal to/is not equal to	left-to-right
8	&	Bitwise AND	left-to-right
9	^	Bitwise exclusive OR	left-to-right
10	|	Bitwise OR	left-to-right
11	&&	Logical AND	left-to-right
12	| |	Logical OR	left-to-right
13	? :	Ternary conditional	**right-to-left**
14	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Shorthand Assignments Bitwise exclusive/inclusive assignment Bitwise shift left/right assignment	**right-to-left**
15	,	Comma (separate expressions)	left-to-right

Exercise

Sr.	Exercise	Answer

1.	int i=1;
	i=2+2*i++;
		4

2.	int a=2,b=7,c=10;
	c=a==b;
		0

3.	int a=2,b=7,c=10;
	c=a!=b;
		1

4.	int a=100 + 200 / 10 - 3 * 10	90

5.	int a = 2, b = 6, c = 12, d;
	d = a * b + c / b;
	printf("The value of d = %d ", d);
		14

Thank You