

What Is the Java™ Technology?

- Java technology is:
- A programming language
- An application environment
- It is similar in syntax to C++.
- It is used for developing both applets and applications.

Primary Goals of the Java Technology

- Provides an easy-to-use language by:
- Avoiding many pitfalls of other languages
- Being object-oriented
- Enabling users to create streamlined and clear code
- Provides an interpreted environment for:
- Improved speed of development
- Code portability
- Enables users to run more than one thread of activity
- Loads classes dynamically; that is, at the time they are actually needed
- Supports changing programs dynamically during runtime by loading classes from disparate sources
- Furnishes better security

The following features fulfil these goals:

The Java Virtual Machine (JVM™) 1

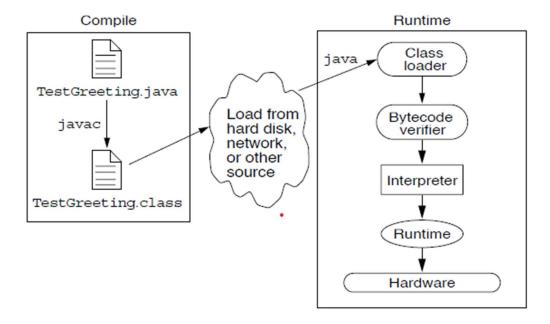
- Provides hardware platform specifications
- Reads compiled byte codes that are platform-independent
- Is implemented as software or hardware
- Is implemented in a Java technology development tool or a Web browser
- Instruction set (central processing unit [CPU])
- Register set
- Class file format
- Stack

- Garbage-collected heap
- Memory area
- Fatal error reporting
- High-precision timing support

Garbage collection

- Allocated memory that is no longer needed should be deallocated.
- In other languages, deallocation is the programmer's responsibility.
- The Java programming language provides a system-level thread to track memory allocation.
- Garbage collection has the following characteristics:
- Checks for and frees memory no longer needed
- Is done automatically
- Can vary dramatically across JVM implementations

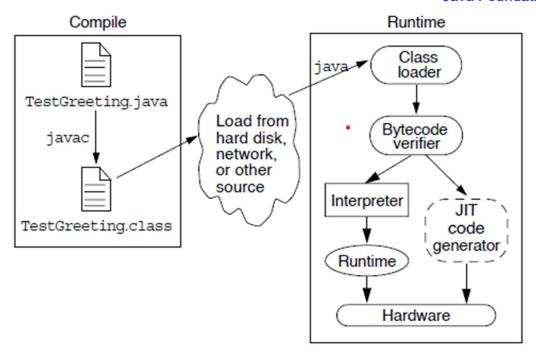
The Java Runtime Environment (JRE)



JVM tool interface

Operation of the JRE with a Just-In-Time (JIT) Compiler

Java Foundation



The Class Loader: Loads all classes necessary for the execution of a program Maintains classes of the local file system in separate namespaces Prevents spoofing

The Bytecode Verifier: The code adheres to the JVM specification. The code does not violate system integrity. The code causes no operand stack overflows or underflows. The parameter types for all operational code are correct. No illegal data conversions (the conversion of integers to pointers) have occurred.

FEATURES OF JAVA

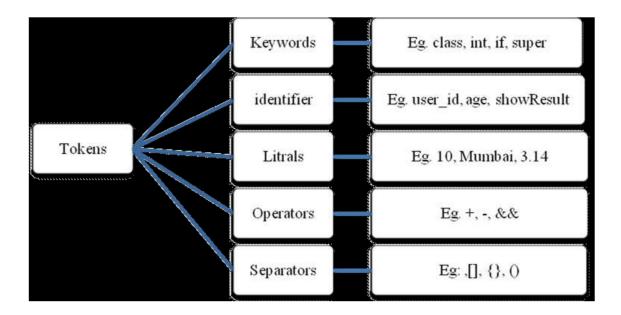
- Simple
- Object-oriented
- Secure, Portable and Robust
- Multithreaded
- Architecture-neutral
- Interpreted & High performance
- Distributed
- Dynamic

Java Foundation

1 pa	ıcl	kage gross_calculator;	//Package Section		
2 im	ıρ	ort java.util.Scanner;	// Import Section		
3 pı	ıb	lic class GrossPayCalculator {	//Class Section or definition		
4	pu	blic static void main(String[] args) {	//Main Section		
5		//Get the number of hours worked	//Comments		
6	i	nt hours = 0;	//Variables		
7		System.out.println("Enter Hours you worked : ");	//Output		
8		Scanner sc = new Scanner (System.in);	//Input		
9		hours = sc.nextInt();			
10		//Get the hourly pay rate			
11		double payRate = 0;			
12	12 System.out.println("Enter Pay Rate per hours : ");				
13	payRate = sc.nextDouble();				
14	4 //Multiply hours and pay rate				
15	5 double grossPay = hours *				
16	6 payRate; Display result				
17	System.out.println("Gross Pay : "+ grossPay);				
18	18 }				
19	19 }				
Line	b	y Line explanation of program			
Line No. Explanation					
	1 : Package Its Container or directory contains classes 2 : Its import library Syntax: import Pkg name> _*				
	 2 : Its import library. Syntax: import<pkg_name><sub_pkg_name>*.</sub_pkg_name></pkg_name> 3 : It's the source-code blueprint for a run-time object Class entry here 				
	4 : Its entry point of program every class have main function				
	5 ,10,14,16 : Comment its documentation its single line // comments here				
Multi-line					
	/*				
		Comments			
		Here */			

- 6 : int is integer data and hours is Variables its location in memory. Variables are data type
- 7 ,12,17 : Its print output of program the System "S" is capital and ("Enter Hours you worked: ") In Parentheses are Massage or Variables System.out.println("Gross Pay : "+ grossPay); the + connect Variable with massage
- 8 : Take input from keyboard Scanner "S" is capital "sc" is user defined object
- 9 : Take input using Scanner and stored in hours variable

Java Tokens - Tokens are the basic building blocks of the java programming language that are used in constructing expressions, statements and blocks. The different types of tokens in java are:



Keywords - These words are already been defined by the language and have a predefined meaning and use. Key words cannot be used as a variable, method, class or interface etc.

Java Programming Language Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Java Foundation

Identifiers - Identifiers are used to name a variable, method, block, class or interface etc. Java is case-sensitive. Identifier may be any sequence of uppercase and lowercase letters, numbers, or the underscore characters.

Literals - Literals are the value assigned to a variable; Example: 10, "Mumbai", 3.14, 'Y', '\n' etc.

Operators - Operators are the symbols that performs operations. Java contains different types of operators like.

Arithmetic Operator (/, *, -, +, %,)

Assignment Operators (+=, -=, *=, /=, %=)

Comparison Operator (<, >, <=,>=, ==, !=)

Logical Operator (&&, ||, !)

Unary Operator (++, --)

Bitwise Operators (\sim - Complement & - AND \wedge - XOR | - OR)

Shift Operators (Right-Shift Operators >> and >>> Left-shift (<<))

OPERATOR PRECEDENCE

Operators	Associative
++ + unary - unary ~ ! (<data_type>)</data_type>	R to L
* / %	L to R
+ -	L to R
<< >> >>>	L to R
< > <= >= instanceof	L to R
== !=	L to R
&	L to R
^	L to R
	L to R
&&	L to R
11	L to R
<pre><boolean_expr> ? <expr1> : <expr2></expr2></expr1></boolean_expr></pre>	R to L
= *= /= %= += -= <<= >>= &= ^= =	R to L

Java Foundation

O Function call Array subscript Dot (Member of structure) Arrow (Member of structure) I Logical NOT One's-complement Unary minus (Negation) H++ Increment Decrement Address-of Indirection (cast Sizeof Multiplication Y Modulus (Remainder) Addition Left-to-Right Addition Subtraction Left-to-Right Subtraction Left-t	C) Function call Array subscript Dot (Member of structure)		Java Foundation		
Comma separates to defining the precedence in expressions, containing expressions in control statements. Every statement end with semicolon Space Used to declare array values. Size of statement Space Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separates consecutive identifiers in a variable declarations. Also used to	CI Array subscript Dot (Member of structure) -> Arrow (Member of structure) ! Logical NOT - One's complement Unary minus (Negation) ++ Indirection - Decrement				
- Dot (Member of structure) Arrow (Member of structure) Logical NOT Che's-complement Unary minus (Negation) He Increment Decrement Address-of Indirection Cast Sizeof Sizeof Sizeof Sizeof Multiplication Left-to-Right 12 Division Modulus (Remainder) He Addition Left-to-Right 11 Subtraction Cast Sizeof Sizeof Left-to-Right 10 Subtraction Left-to-Right 10 Subtraction Left-to-Right 10 Subtraction Left-to-Right 8 Less than or equal to Creater than or equal to Creater than or equal to Left-to-Right 8 Site Sizeof Sizeo	- Dot (Member of structure) - Arrow (Member of structure) ! Logical NOT - One's-complement - Unary minus (Negation) Heavier of the complement of the compl	()			
-> Arrow (Member of structure) ! Logical NOT - One's-complement Unary minus (Negation) ++ Decrement Address of Indirection (type) sizeof Sizeof / Division Modulus (Remainder) + Addition	-> Arrow (Member of structure) ! Logical NOT - One's-complement - Unary minus (Negation) ++ Increment - Decrement - Address-of		Array subscript		
Logical NOT One*-complement Unary minus (Negation) Increment Decrement Address-of Indirection Cast Sizeof Sizeof Sizeof Sizeof Sizeof Hultiplication Division Modulus (Remainder)	! Logical NOT One's-complement Unary minus (Negation) ++	-	Dot (Member of structure)		
- One's-complement Unary minus (Negation) Increment Decrement Address-of Indirection Cast Sizeof * Multiplication Division Modulus (Remainder) + Addition Subtraction - Subtraction - Subtraction - Subtraction - Left-to-Right - Right shift - Less than - Less than - Greater than - Greater than or equal to - Greater than - Greater than or equal to - Hitwise AND - Bitwise OR - Left-to-Right - Bitwise OR - Left-to-Right - Right - Subtraction - Subtraction - Left-to-Right - Conditional - Left-to-Right - Right-shift - Right-shift - Conditional - Right-shift - Ri	- One's-complement Unary minus (Negation) ++ Increment Decrement Address-of Indirection Cast Sizeof Multiplication Jivision Modulus (Remainder) + Addition - Subtraction - Left-to-Right - Right-shift - Less than - Less than - Left-to-Right - Less than or equal to - Greater than or equal to - Greater than - Greater than - Greater than - Right-to-Right - Bitwise AND - Left-to-Right - Bitwise AND - Left-to-Right - Right-to-Right - Right-to-Left - Right				
- Unary minus (Negation) + + - Decrement Address-of Indirection Cast Sizeof * Multiplication / Division / Modulus (Remainder) + Addition Subtraction - Subtraction - Left-to-Right - Subtraction - Left-shift - Left-to-Right - Right-shift - Less than - Creater than Greater than or equal to - Greater than or equal to - Hitwise AND - Left-to-Right - Hitwise AND - Right-shift - Conditional - Right-to-Left - Right - Hitwise AND - Hitwise AND - Right-to-Right - Right-to-Left - Right - Hitwise AND - Left-to-Right - Right - Hitwise AND - Left-to-Right - Right - Ri	- Unary minus (Negation) Increment Address of Indirection Cast Sizeof Multiplication Joivision Modulus (Remainder) + Addition Subtraction < Left-to-Right Left-to-Right 10	ļ.		Right-to-Left	13
++ Increment Address of Indirection Cast Sizeof Sizeof Sizeof Multiplication Left-to-Right 12 Division Modulus (Remainder) Head of the process of	++ Increment Decrement Address of Indirection Cast Sizeof Sizeof Sizeof Sizeof Multiplication Left-to-Right 12 Division Modulus (Remainder) + Addition Left-to-Right 11 Subtraction Cast Left-shift Left-to-Right 10 Subtraction Cast Left-shift Left-to-Right 10 Cast Less than Left-to-Right 8 Cast	-	One's-complement	13243	
		_	Unary minus (Negation)		
* Address-of Indirection Cast Sizeof Sizeof * Multiplication / Division Modulus (Remainder) + Addition	Address-of Indirection (type) Sizeof Multiplication / Division / Modulus (Remainder) + Addition Subtraction <	++	Increment		
* Indirection Sizeof Sizeof Sizeof Multiplication / Division / Modulus (Remainder) + Addition Subtraction - Subtraction - Subtraction - Left-to-Right - Subtraction - Left-shift - Leas than - Lest than or equal to - Greater than - Greater than or equal to - Equal to - Equal to - Bitwise AND - Bitwise AND - Bitwise AND - Bitwise AND - Left-to-Right - Bitwise AND - Left-to-Right - Bitwise AND - Left-to-Right - Right-shift - Conditional - Right-to-Left - Right - Right-to-Right - Right-t	* Indirection Cast Sizeof * Multiplication Division * Modulus (Remainder) + Addition Subtraction < Left-shift >> Right-shift < Less than Careater than or equal to Seater than Or eater than Or equal to Seater than Or equal to Seater than Or e				
Cast Sizeof Sizeof	(type) sizeof sizeof Multiplication Division Modulus (Remainder) + Addition Subtraction < Left-to-Right 11 - Subtraction < Left-shift Left-to-Right 10 Right-shift < Less than or equal to Greater than Segretar than or equal to Segretar than or eq				
Cast Sizeof Sizeof	(type) sizeof sizeof Multiplication Division Modulus (Remainder) + Addition Subtraction < Left-to-Right 11 - Subtraction < Left-shift Left-to-Right 10 Right-shift < Less than or equal to Greater than Segretar than or equal to Segretar than or eq				
# Multiplication	# Multiplication Left-to-Right 12 / Division Modulus (Remainder) + Addition Left-to-Right 11 - Subtraction Left-to-Right 10 < Left-shift Left-to-Right 10 > Right-shift Left-to-Right 8 < Less than or equal to Greater than or equal to = Equal to Left-to-Right 8 ! Not equal to \$\frac{1}{2}\$ Bitwise AND Left-to-Right 6 I Bitwise OR Left-to-Right 5 \$\frac{1}{2}\$ Logical AND Left-to-Right 3 ? : Conditional Right-to-Left 2 - + Assignment operators Right-to-Left 1 * = , etc. Comma Left-to-Right Lowest Symbol Name Description Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast ty Braces Used to contains the values of automatically initialized arrays.				
Multiplication Left-to-Right 12	# Multiplication Left-to-Right 12				
/ Division Modulus (Remainder) + Addition Subtraction <	/ Division Modulus (Remainder) + Addition Left-to-Right 11 - Subtraction < Left-shift Left-to-Right 10 Right-shift Left-to-Right 8 - Less than or equal to Greater than Greater than or equal to = Equal to Left-to-Right 8 ! Not equal to \$\frac{\pi}{2}\$ Bitwise AND Left-to-Right 6 Bitwise AND Left-to-Right 6 Bitwise OR Left-to-Right 5 \$\frac{\pi}{2}\$ Logical AND Left-to-Right 4 Logical OR Left-to-Right 3 ? : Conditional Right-to-Left 2 =, += Assignment operators Right-to-Left 1 *=, etc. Symbol Name	*		Left-to-Right	12
# Addition Left-to-Right 11 - Subtraction Left-to-Right 10 - Subtraction Left-to-Right 10 - Right-shift Left-to-Right 10 - Right-shift Left-to-Right 8 - Less than or equal to Left-to-Right 8 - Greater than Left-to-Right 8 - Greater than Left-to-Right 8 - Right-shift Left-to-Right 8 - Right-shift Reft-to-Right 8 - Ritwise AND Left-to-Right 7 - Ritwise AND Left-to-Right 6 - Ritwise OR Left-to-Right 5 - Ritwise OR Left-to-Right 5 - Ritwise OR Left-to-Right 4 - Right Logical AND Left-to-Right 3 - Conditional Right-to-Left 2 - Reft Right-to-Left 2 - Reft Right-to-Left 1 - Ritwise OR Right-to-Left 1 - Right-to-Right 3 - Right-to-Righ	# Addition Left-to-Right 11 - Subtraction Left-to-Right 11 < Left-shift Left-to-Right 10 > Right-shift Left-to-Right 8 < Less than Left-to-Right 8 < Less than or equal to Cereater than Cereater	7		Lete to reight	12
+ Addition - Subtraction < Left-shift >> Right-shift	+ Addition Subtraction <				
Subtraction <pre></pre>	- Subtraction <			Left to Dight	11
<pre></pre>	<pre></pre>	T		Lett-to-rught	11
Right-shift Less than Left-to-Right 8	Not equal to Conditional			T C + D' L	10
 Less than Less than or equal to Greater than Greater than Host equal to Not equal to Left-to-Right Not equal to Not equal to Left-to-Right Ritwise AND Left-to-Right Bitwise OR Left-to-Right Left-to-Right Conditional Right-to-Right Right-to-Left Conditional Right-to-Left Right-to-Left Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement 	<pre></pre>			Lett-to-Right	10
Less than or equal to Symbol Name Description	<pre></pre>			T 6: T	
> Greater than or equal to == Equal to != Not equal to Equal to Left-to-Right 8	Semicolon Used to separate the statements. Every statement end with semicolon			Left-to-Right	8
Symbol Name Semicolon Used to separate the statements. Every statement end with semicolon Space Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separate consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement Comma separate consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Semicolon Used to separate the statements. Every statement end with semicolon				
Equal to Not equal to	## Equal to Not equal to ## Bitwise AND Left-to-Right 7 Bitwise XOR Left-to-Right 6 Bitwise OR Left-to-Right 5 Left-to-Right 5 Left-to-Right 4 I Logical AND Left-to-Right 4 I Logical OR Left-to-Right 3 Conditional Right-to-Left 2 Assignment operators Right-to-Left 1 Eq. etc. Comma. Left-to-Right Lowest Symbol Name Semicolon				
! = Not equal to & Bitwise AND	! = Not equal to & Bitwise AND				
Bitwise AND Bitwise XOR Bitwise OR Left-to-Right 6 Bitwise OR Left-to-Right 5 Logical AND Left-to-Right 4 Logical OR Left-to-Right 3 Picket Logical OR Left-to-Right 3 Picket Logical OR Left-to-Right 3 Picket Logical OR Right-to-Left 2 Left-to-Right 1 Assignment operators Right-to-Left 1 Fight Lowest 0 Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types Braces Used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Bitwise AND Bitwise XOR Bitwise OR Left-to-Right Bitwise OR Left-to-Right Logical AND Left-to-Right Logical OR Conditional Assignment operators E, etc. Comma Comma Left-to-Right Right-to-Left Right-to-Left Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast ty Braces Used to contains the values of automatically initialized arrays.	== Equal to Left-to-Right			8
Bitwise OR	Bitwise XOR Bitwise OR Left-to-Right 5 Bitwise OR Left-to-Right 5 Logical AND Left-to-Right 4 Logical OR Left-to-Right 3 Conditional Right-to-Left 2 Assignment operators Right-to-Left 1 Eq. etc. Right Lowest	! = Not equal to			
Bitwise OR	Bitwise OR Left-to-Right 5	& Bitwise AND		Left-to-Right	7
Logical AND Left-to-Right 4 Logical OR Left-to-Right 3 Left-to-Right 3 Conditional Right-to-Left 2 Assignment operators Right-to-Left 1 Comma Left-to-Right 1 Assignment operators Right-to-Left 1 Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Left-to-Right 4 I Logical OR Left-to-Right 3 Conditional Right-to-Left 2 Assignment operators Right-to-Left 1 Eq. + = Assignment operators Right-to-Left 1 Symbol Name Description	-	Bitwise XOR	Left-to-Right	6
Logical AND I Logical OR Conditional Right-to-Right Right-to-Left Right-to-Left Right-to-Left Right-to-Left Right-to-Left Right-to-Left Right-to-Left Lowest 0 Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Logical AND Left-to-Right 4 Logical OR Conditional Right-to-Left 2 Assignment operators Right-to-Left 1 Comma Left-to-Right 1 Right-to-Left 1 Lowest Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast ty Braces Used to contains the values of automatically initialized arrays.	1	Bitwise OR	Left-to-Right	5
Logical OR	?: Conditional Right-to-Left 2 =, += Assignment operators Right 1 **Example 1	**	Logical AND		4
?: Conditional Right-to-Left 2 =, +=, etc. Right Lowest 0 Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. () Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types are used to define a block of code, for classes, methods, and local scopes. [] Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	? : Conditional Right-to-Left 2 =, += Assignment operators Right-to-Left 1 * =, etc.	11			3
# = , +=	# = , # = Assignment operators Comma				
Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	 t = , etc. Comma. Left-to-Right Lowest Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. 				
Symbol Name Description	Symbol Name Description			Tugin to Lexi	
Symbol Name Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	 Symbol Name Description Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. 	, ,			
; Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. () Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. [] Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	 Semicolon Used to separate the statements. Every statement end with semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. 	0 1 101			
Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.	Symbol Name	nbol Name Description		
Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.				
Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	semicolon Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.				
Space Used to separate tokens. () Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. [] Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.	:	Semicolon Used to separate the stater	nents. Every statem	ient end with
Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Space Used to separate tokens. Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.	•			
Parentheses Used to contain the lists of parameters in method definition and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Parentheses Used to contain the lists of parameters in method definit and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.				
and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.		space used to separate tokens.		
and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.	Describered Head to contain the lists of management and its additional in the list of manageme		thad dafinition	
and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast types. Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	and invocation. Also used for defining the precedence in expressions, containing expressions in control statements, and surrounding cast tyles. Braces Used to contains the values of automatically initialized arrays.			•	
Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	containing expressions in control statements, and surrounding cast tyles are seen as the same of automatically initialized arrays.	and invocation. Also used for defining the precedence		the precedence in e	expressions,
Braces Used to contains the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Braces Used to contains the values of automatically initialized arrays.	containing expressions in control statements, and surroundin			•
Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement					
Also used to define a block of code, for classes, methods, and local scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Also used to define a block of code for classes, methods, and less!	{}		•	•
scopes. Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	AISO USEU LO UETITIE A DIOCK OF COUE, TOT CIASSES, MELHOUS, AND IOCAL	and local			
Brackets Used to declare array types. Also used when dereferencing array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement				,,	
array values. Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement					
Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	Brackets Used to declare array types. Also used when dereferencing	[]	Brackets Used to declare array types. /	Also used when der	eterencing
Comma separates consecutive identifiers in a variable declarations. Also used to chain statements together inside a for statement	array values.		array values.		
used to chain statements together inside a for statement	41.47 14.40.				
used to chain statements together inside a for statement					
used to chain statements together inside a for statement	Comma separates consecutive identifiers in a variable declarations. Al	Comma separates consecutive identifiers in a variable declarat		clarations. Also	
	· ·				
Deviced Head to separate markets are from sub-markets and places	used to chain statements together inside a for statement				
Devied Head to consust analysis and from sub-neglected and places					
I DOUND LICENTAIN TO CONTRACT DATE AND CHANGE THAT CHE AND CHACE AND CHACE	Period Used to senarate nackages names from sub nackages and class	sec and classes			
, , , , , , , , , , , , , , , , , , , ,		•	Period Used to separate packages names from sub packages and classes.		
I Also used to separate a variable or method from a reference variable.	Also used to separate a variable or method from a reference variable.		Also used to separate a variable or method from a reference variable.		



DATA TYPES:-

Primitive Data: Predefined or built in data types &

Non Primitive Data: User defined data type. They are String, Array, Classes and Interface ...etc.

Primitive Data:

Refer https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Data Type	Size	Example
Integer (int)	By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -231 and a maximum value of 231-1. In Java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of 232-1. Use the Integer class to use int data type as an unsigned integer. See the section The Number Classes for more information. Static methods like compareUnsigned, divideUnsigned etc have been added to the Integer class to support the arithmetic operations for unsigned integers.	Int grossPay = 2000;
long	The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -263 and a maximum value of 263-1. In Java SE 8 and later, you can use the long data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of 264-1. Use this data type when you need a range of values wider than those provided by int. The Long class also contains methods like compareUnsigned, divideUnsigned etc to support arithmetic operations for unsigned long.	Long int grossPay = 200000;
Double	The double data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the Floating-Point Types, Formats, and Values section of the Java Language Specification. For decimal values, this data type is generally the default choice. As mentioned above, this data type	double d1 = 35.25d;

Java Foundation

		• • • • • • • • • • • • • • • • • • • •
	should never be used for precise values,	
Short	such as currency. The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with byte, the same guidelines apply: you can use a short to save memory in large arrays, in situations where the memory savings actually matters.	Stores whole numbers from -32,768 to 32,767
Byte	The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters. They can also be used in place of int where their limits help to clarify your code; the fact that a variable's range is limited can serve as a form of documentation.	byte x = 10;
Float	The float data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the Floating-Point Types, Formats, and Values section of the Java Language Specification. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the java.math.BigDecimal class instead. Numbers and Strings covers BigDecimal and other useful classes provided by the Java platform.	Float grossPay = 2000.52;
Char	The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).	char choice = 'A';
Boolean	The Boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.	a=10; b=20; bool a==b;

Java Foundation

Print format		Example
%d	Int	System.out.print("%d"+ grossPay);
%f	Float	System.out.print("%f"+ grossPay);
%с	Char	System.out.print("%c"+ choice);
%S	String	System.out.print("%S"+ name);

NAMING CONVENTION FOR VARIABLES:-

- 1. Don't start variables name start with digit Example: int 1sale
- 2. Variables name are case sensitive Example: int sale & int Sale are different
- 3. Should not be a keyword
- 4. White space not allowed
- 5. Can contain alphabets, \$ and _underscore and digit
- 6. Use Pascal Convention for classes Example: public class GreenTree;
- 7. Use camel Case Convention for functions and variables Example: getInput(), getPrint()
- 8. Package name should lower-case

Examples:

- 1. Write a java program to add three numbers
- 2. Write java program to calculate percentage of given student in CBSE board exam. His marks from 5 subjects must be taken input from the keyboard.
- 3. Write a java program which asks the user to enter his/her name and greeting them with "Hello <name>, have a good day" text.
- 4. Write a java program to converts kilometres to miles
- 5. Write a java program to detect whether a number entered by the user is integer or not
- 6. How will you write following expression in java
 - x-y/2, $b^2-4ac/2a$,
 - V²-u², a *b-d
- 7. What is the value of following
 - Int y=7;Int x= ++y*8
- 8. What will be the result of following expression a = 7/4*9/2
- 9. User comparison operator to find out whether a given number is greater than the user enter number or not.
- 10. Write the following expression in a java program $v^2-u^2/2a5$
- 11. Write a java program to convert a string to lower case.