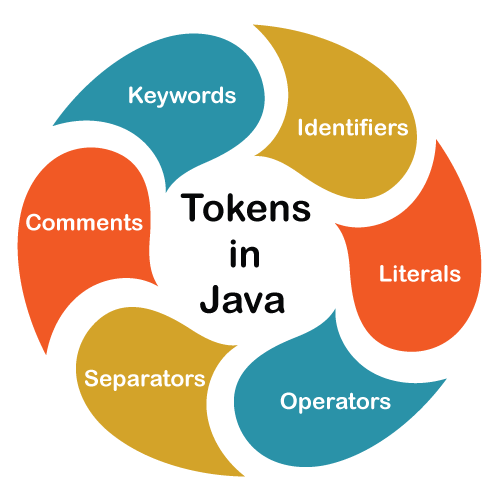
***Practical 2 : Questions***

Q1. What are the different tokens in Java programming?

Java Tokens

In Java, the program contains classes and methods. Further, the methods contain the expressions and statements required to perform a specific operation. These statements and expressions are made up of **tokens**. In other words, we can say that the expression and statement is a set of **tokens**. The tokens are the small building blocks of a Java program that are meaningful to the [Java](https://www.javatpoint.com/java-tutorial) compiler.

The Java compiler breaks the line of code into text (words) is called Java tokens. These are the smallest element of the Java program. The Java compiler identified these words as tokens. These tokens are separated by the delimiters. It is useful for compilers to detect errors. Remember that the delimiters are not part of the Java tokens.



REMEMBER : COSILK

C-> Comments

O-> Operators

S -> Separators

I -> Identifiers

L -> Literals

K -> Keywords

**Keywords:** These are the **pre-defined** reserved words of any programming language. Each [keyword](https://www.javatpoint.com/java-keywords) has a special meaning. It is always written in lower case. Java provides the following keywords

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01. abstract | 02. boolean | 03. byte | 04. break | 05. class |
| 06. case | 07. catch | 08. char | 09. continue | 10. default |
| 11. do | 12. double | 13. else | 14. extends | 15. final |
| 16. finally | 17. float | 18. for | 19. if | 20. implements |
| 21. import | 22. instanceof | 23. int | 24. interface | 25. long |
| 26. native | 27. new | 28. package | 29. private | 30. protected |
| 31. public | 32. return | 33. short | 34. static | 35. super |
| 36. switch | 37. synchronized | 38. this | 39. thro | 40. throws |
| 41. transient | 42. try | 43. void | 44. volatile | 45. while |
| 46. assert | 47. const | 48. enum | 49. goto | 50. strictfp |

**Identifier:** Identifiers are used to name a variable, constant, function, class, and array. It usually defined by the user. It uses letters, underscores, or a dollar sign as the first character. The label is also known as a special kind of identifier that is used in the goto statement. Remember that the identifier name must be different from the reserved keywords. There are some rules to declare identifiers are:

* The first letter of an identifier must be a letter, underscore or a dollar sign. It cannot start with digits but may contain digits.
* The whitespace cannot be included in the identifier.
* Identifiers are case sensitive.

**Literals:** In programming literal is a notation that represents a fixed value (constant) in the source code. It can be categorized as an integer literal, string literal, Boolean literal, etc. It is defined by the programmer. Once it has been defined cannot be changed. Java provides five types of literals are as follows:

* Integer
* Floating Point
* Character
* String
* Boolean

|  |  |
| --- | --- |
| **Literal** | **Type** |
| 23 | int |
| 9.86 | double |
| false, true | boolean |
| 'K', '7', '-' | char |
| "javatpoint" | String |
| null | any reference type |

**Operators:** In programming, operators are the special symbol that tells the compiler to perform a special operation. Java provides different types of operators that can be classified according to the functionality they provide. There are eight types of [operators in Java](https://www.javatpoint.com/operators-in-java), are as follows:

|  |  |
| --- | --- |
| **Operator** | **Symbols** |
| **Arithmetic** | + , - , / , \* , % |
| **Unary** | ++ , - - , ! |
| **Assignment** | = , += , -= , \*= , /= , %= , ^= |
| **Relational** | ==, != , < , >, <= , >= |
| **Logical** | && , || |
| **Ternary** | (Condition) ? (Statement1) : (Statement2); |
| **Bitwise** | & , | , ^ , ~ |
| **Shift** | << , >> , >>> |

**Separators:** The separators in Java is also known as punctuators. There are nine separators in Java, are as follows:

separator <= ; , . ( ) { } [ ]

* **Square Brackets []:** It is used to define array elements. A pair of square brackets represents the single-dimensional array, two pairs of square brackets represent the two-dimensional array.
* **Parentheses ():** It is used to call the functions and parsing the parameters.
* **Curly Braces {}:** The curly braces denote the starting and ending of a code block.
* **Comma (,):** It is used to separate two values, statements, and parameters.
* **Assignment Operator (=):** It is used to assign a variable and constant.
* **Semicolon (;):** It is the symbol that can be found at end of the statements. It separates the two statements.
* **Period (.):** It separates the package name form the sub-packages and class. It also separates a variable or method from a reference variable.

**Comments:** Comments allow us to specify information about the program inside our Java code. Java compiler recognizes these comments as tokens but excludes it form further processing. The Java compiler treats comments as whitespaces. Java provides the following two types of comments:

* **Line Oriented:** It begins with a pair of forwarding slashes (**//**).
* **Block-Oriented:** It begins with /\* and continues until it founds **\*/**.

Q2. What are the different data types in Java and explain size and range of each data types? AND

Q6. What are the default values of different data types?

Primitive data types

|  |  |
| --- | --- |
| byte | Value ranges from -128 to 127  Takes 1 byte  Default value is 0 |
| short | Value ranges from -(2)^16/2 to ((2^16)/2) -1  Takes 2 bytes  Default value is zero |
| int | Value ranges from –(2)^32/2 to ((2^32)/2)-1  Takes 4 bytes  Default value is zero |
| float | Takes 4 bytes  Default value is 0.0f |
| long | Ranges from –(2)^64/2 to ((2^64)/2)-1  Takes 8 bytes  Default value is zero |
| double | Takes 8 bytes  Default value is 0.0d |
| char | Value ranges from 0 to (2^16)-1  Takes 2 bytes (it supports Unicode)  Default value is \U0000 |
| bool | Value can be true or false  Default value is false  Size depends on JVM |

Q3. How to get in variable values from the user for different data types?

Make use of Scanner class from util package

Syntax : import java.util.Scanner;

….

….

Scanner Sc = new Scanner (System.in);

byte x = sc.nextByte();

short x = sc.nextShort();

int x = sc.nextInt();

float x = sc.nextFloat();

long x = sc.nextLong();

double x = sc.nextDouble();

char x = sc.nextCharAt(0);

boolean x = sc.nextBoolean();

String s = sc.next();

Q4. How to check user variable values are correct or not?

Scanner sc = new Scanner (System.in);

System.out.println(sc.hasNextInt());

Q5. What are the different types of variables in JAVA programming?

* + Local variable
  + Instance variable
  + Static or Class variable

### Local Variables

These variables are declared inside the body of a method. These can be used within the same method where it is being initialized.

**Some of the properties of a Local Variable include:**

1. Local variables are declared inside a method, constructor, or block.
2. No access modifiers for local variables.
3. These can be used only within the same block, method, or constructor where it is initialized.
4. No default value after you have declared your local variable. You need to initialize your declared local variable.
5. It can’t be defined by a static keyword.

### Instance Variables

Instance variables are those variables that are declared inside a class. Unlike Local variables, these variables cannot be declared within a block, method, or constructor.

**Enlisted below are the properties of the Instance variable:**

1. They are declared within a class but outside a block, method or constructor.
2. It cannot be defined by a static keyword.
3. Unlike Local variables, these variables have a default value.
4. The integer type has a default value ‘0’ and the boolean type has the default value ‘false’.
5. Unlike Local variables, we have access modifiers for Instance variables.

### Static Or Class Variable

Unlike the Local and Instance variable (where we can not use static), we have another variable type which is declared as static and is known as “Static or Class variable”.

**Given below are some of the properties of the Static or Class variable:**

1. These variables cannot be local.
2. Static variables are shared among all the instances of a class.
3. The default values of Static/Class variables are the same as the Instance variables.
4. Static variables can be used within a program by calling the className.variableName
5. The memory allocated to store Static variables is Static memory.

Q7. What are the different backslash constants in JAVA?

The following table denotes the backslash character constants used in Java.

|  |  |
| --- | --- |
| \b | Backspace |
| \f | From feed |
| \n | New line |
| \r | Carriage return |
| \t | Horizontal tab |
| \" | Double quote |
| \' | Single quote |
| \\ | Backslash |
| \v | Vertical tab |
| \a | Alert |
| \? | Question mark |
| \N | Octal constant |
| \xN | Hexadecimal constant |

Q8. What are the different types of typecasting in Java? AND

Q9. How to perform widening and narrow typecasting. Explain with example?

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

* **Widening Casting** (automatically) - converting a smaller type to a larger type size  
  byte -> short -> char -> int -> long -> float -> double
* **Narrowing Casting** (manually) - converting a larger type to a smaller size type  
  double -> float -> long -> int -> char -> short -> byte

## **Widening Casting**

Widening casting is done automatically when passing a smaller size type to a larger size type:

### **Example**

public class Main {

public static void main(String[] args) {

int myInt = 9;

double myDouble = myInt; // Automatic casting: int to double

System.out.println(myInt); // Outputs 9

System.out.println(myDouble); // Outputs 9.0

}

}

## **Narrowing Casting**

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

### **Example**

public class Main {

public static void main(String[] args) {

double myDouble = 9.78d;

int myInt = (int) myDouble; // Manual casting: double to int

System.out.println(myDouble); // Outputs 9.78

System.out.println(myInt); // Outputs 9

}

}

Q10. What are the different types of operators in JAVA ?

|  |  |  |
| --- | --- | --- |
| Unary | postfix | *expr*++ *expr*-- |
| prefix | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| Arithmetic | multiplicative | \* / % |
| additive | + - |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |