**Template: Study Material**

|  |  |  |
| --- | --- | --- |
| *Insert the details within the < quotes >*  <CO1>: <22412>: <Java Programming>: <Basic Syntactical Constructs in Java>: <LO2>: <Study Material> | | |
| <Yogita Jore> | <01/02/2021> | <Vijay Patil> |
| Key words  **Nesting of Methods and This Keyword** | Learning Objective:  **student should understand characteristics of the given Java Token.** | Diagram/ Picture  **Smallest individual units in a program are known as tokens**.  Rules to define identifier:  1) All identifiers must start with either a letter (a to z or A to Z) or currency character ($) or an underscore.  2) They must not begin with a digit  3) After the first character, an identifier can have any combination of characters.  4) A Java keyword cannot be used as an identifier.  5) Identifiers in Java are case sensitive, foo and Foo are two different identifiers.  6) They can be any length   | Constant | Type of Value Stored | | --- | --- | | Integer Literals | Literals which stores integer value | | Floating Literals | Literals which stores float value | | Character Literals | Literals which stores character value | | String Literals | Literals which stores string value | | Boolean Literals | Literals which stores true or false | |
| Key Questions  **Identify the identifier from following program.**    Ans: | Concept Map    **Explanation of Concept**  **Java Token**  A **token** is the smallest element of a program that is meaningful to the compiler. Simply we can say that java program is also collection of tokens, comments and whitespaces.  A class is defined by a set of declaration statements and methods containing executable statements. Most statements contain expressions, which describe the actions carried out on data. The word formed from the character set is building block of java and are known as **token**. These tokens represent the individual entity of language. |
| | **Token** | **Meaning** | **Example** | | --- | --- | --- | | **Keyword** | A variable is a meaningful name of data storage location in computer memory. When using a variable you refer to memory address of computer. | **int** distance;  **continue**  **break**  **public**  **return**  **do**  **while**  **for** | | **Constant** | Constants are expressions with a fixed value. They are also called as literals  **Syntax:** final data\_type variable\_name; | final int a=20; | | **Identifier** | Identifiers are the names of variables, methods, classes, packages and interfaces. | **MYVARIABLE**  **x**  **i**  **\_myvariable**  **$myvariable**  **\_9pins**  **andros**  **??????** | | **String** | Sequence of characters. | char[] ch={'a','t','n','y','l','a'};  String s=new String(ch); | | **Special Symbol** | Symbols other than the Alphabets and Digits and white-spaces. | [] () {}, ; \* = | | **Operators** | A symbol that represents a specific mathematical or non-mathematical action. | / + == 1 ? |   **Data Types:**  Data types specify the different sizes and values that can be stored in the variable.    Java has two categories of data:  **Primitive Data Type:** such as boolean, char, int, short, byte, long, float, and double  **Non-Primitive Data Type or Object Data type:** such as String, Array, etc.   |  |  |  | | --- | --- | --- | | **Data Type** | **Default Value** | **Default size** | | boolean | False/true | 1 bit | | char | '\u0000' | 2 byte | | byte | 0 | 1 byte | | short | 0 | 2 byte | | int | 0 | 4 byte | | long | 0L | 8 byte | | float | 0.0f | 4 byte | | double | 0.0d | 8 byte | | Key Definitions/ Formulas  Constructors are used to assign initial values to instance variables of the class.  There are many types of operators in java which are given below:   * Unary Operator, * Arithmetic Operator, * shift Operator, * Relational Operator, * Bitwise Operator, * Logical Operator, * Ternary Operator and * Assignment Operator   Data types specify the different sizes and values that can be stored in the variable. |
|  |
| **Variables:**  A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.  Variable is a name of memory location. There are three types of variables in java: local, instance and static.  **Rules for naming variables**   * The name of a variable needs to be meaningful, short and without any embedded space or symbol like ? ! @ # % ^ & \* ( ) [ ] { } . , ; : “ ‘ / and \. However underscore can be used wherever a space is required for example basic\_salary. * Variable name must be unique. For example to store four different numbers, four unique variable names need to be used. * A variable name must begin with a letter, a dollar symbol (‘$’) or an underscore (‘\_’) , which may be followed by a sequence of letters or digits (0-9), ‘$’ or ‘\_’. * Keywords cannot be used for variable names. For example, you cannot declare a variable called *switch.* * Variable names must be meaningful. The names must reflect the data that the variables contain. Example to store the age of an employee, the variable name could be *employeeage*. * Variable names are nouns and begin with a lowercase letter. * If a variable name contains two or more words, join the words and begin each word with an uppercase letter. The first word, however, starts with a lowercase letter. * Valid variable names: address1, studentname, total\_salary * Invalid variable names: $salary, 1stname   **How to declare variables?**  Syntax: datatype var1, var2, var3, ………., varn;  The declaration should be terminated by semicolon (;) whereas all the variable names of same datatype are separated by commas;    **Example:**  float simpleInterest; //Declaring float variable  int time = 10, speed = 20; //Declaring and Initializing integer variable  char var = 'h'; // Declaring and Initializing character variable  **Types of variables/scope of variable:**  There are three types of variables in Java:   * Local Variables * Instance Variables * Static Variables   **Local Variables**: A variable defined within a block or method or constructor is called local variable.   * These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function. * The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block. * Initialization of Local Variable is Mandatory.   **Instance Variables**: Instance variables are non-static variables and are declared in a class outside any method, constructor or block.   * As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed. * Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier, then the default access specifier will be used. * Initialization of Instance Variable is not Mandatory. Its default value is 0 * Instance Variable can be accessed only by creating objects.   **Static Variables**: Static variables are also known as **Class variables.**   * These variables are declared similarly as instance variables; the difference is that static variables are declared using the static keyword within a class outside any method constructor or block. * Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create. * Static variables are created at the start of program execution and destroyed automatically when execution ends. * Initialization of Static Variable is not Mandatory. Its default value is 0 * If we access the static variable like Instance variable (through an object), the compiler will show the warning message and it won’t halt the program. The compiler will replace the object name to class name automatically. * If we access the static variable without the class name, Compiler will automatically append the class name.   **Instance variable Vs Static variable**   * Each object will have its **own copy** of instance variable whereas We can only have **one copy** of a static variable per class irrespective of how many objects we create. * Changes made in an instance variable using one object will **not be reflected** in other objects as each object has its own copy of instance variable. In case of static, changes**will be reflected** in other objects as static variables are common to all object of a class. * We can access instance variables **through object references** and Static Variables can be accessed **directly using class name.**   **Dynamic initialization:**  Initialization is the process of providing value to a variable at declaration time. A variable is initialized once in its life time. Any attempt of setting a variable's value after its declaration is called assignment. To use a local variable, you have to either initialize or assign it before the variable is first used. But for class members, the compulsion is not so strict. If you don't initialize them then compiler takes care of the initialization process and set class members to default values. Java allows its programmers to initialize a variable at run time also. Initializing a variable at run time is called dynamic initialization. Java allows variables to be initialized dynamically, using any expression valid at the time the variable is declared.  **Example:**  public class MainClass  { public static void main(String args[])  { double a = 3.0, b = 4.0;  // c is dynamically initialized double c = Math.sqrt(a \* a + b \* b); System.out.println("Hypotenuse is " + c); } }  **Array:**  An array is a collection of similar types of data.  For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.  Syntax: datatype[ ] arrayname;  Example: double[ ] percentage;  In Java, we can initialize arrays during declaration.  For example,  int[ ] age={12,4,5,2,5};    **String:**  In [Java](https://www.javatpoint.com/java-tutorial), string is basically an object that represents sequence of char values. An [array](https://www.javatpoint.com/array-in-java) of characters works same as Java string. For example:  char[] ch={‘M’,’S’,’B’,’T’,’E’};  String s=new String(ch);  **Java String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.  **Application of Concept/ Examples in real life:**   * Used to declare variables * Assigned to values to variables and constants | **Variable**is name of reserved area allocated in memory.  In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed. |
| Key Take away from this LO:  Concept of java tokens, data types, constants, type casting, variables | | |