CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 1: Java Fundamental and Programming Concepts

Chapter 4

Objective: After completing this lesson you will be able to : * Get familiar with Java Keywords and Statements, Identifier, literals, and data types	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 120 minutes	**Practical Duration:** 0 minute
Total Duration: 120 minutes	

Chapter 4

4.1 Keywords and Statements, Identifier, Literals

Java Keywords – Keywords in Java are also known as reserved words. Keywords are designated words which perform predefined actions when used in code. Java keywords cannot be used as object names or variables. There are a total of 50 Java keywords used by programmers today. Take a look below at what function each keyword performs.

List of Java keywords

- **abstract** - Designates a method or class for later implementation within a subclass
- **assert** - Assert denotes a true–false statement predicate in a Java program. It indicates that a programmer considers the predicate as always true. Execution is stopped if assertion is 'false' during the runtime.
- **boolean** - Data type that holds only True and False values only
- **break** - A control statement for breaking loops
- **byte** - A data type capable of holding 8-bit data values
- **case** - Used to mark text blocks in switch statements
- **catch** - Catches try statement generated exceptions
- **char** - A data type capable of holding unsigned Unicode 16-bit characters
- **class** - Used for declaring a newly created class
- **continue** - Re-transfers control back outside of a loop
- **default** - Used for specifying the default code block in a switch statement
- **do** - Used for starting do-while loop
- **double** - A data type capable of holding 64-bit floating-point numbers
- **else** - Used for indicating alternative branches of an 'if' statement
- **enum** - Used for declaring an enumerated type. Base class can be extended through enumerations.
- **extends** - Denotes that a class has been derived from another class or interface.
- **final** - Denotes that a variable possesses a constant value, and that there will be no overriding of a method.
- **finally** - Denotes a code block to be executed always, within a try-catch structure.
- **float** - A data type capable of holding a 32-bit floating-point number
- **for** - Used for initiating a for loop
- **goto** - A reserved keyword with no specific functions
- **if** - Used for testing true/false expression and its branches

- **implements** – Denotes that an interface is implemented by a class
- **import** – Used to reference other classes
- **instanceof** – Denotes if an object is a specific class instance or an interface implementation
- **int** – A data type capable of holding a 32-bit signed integer
- **interface** – Used for declaring an interface
- **long** – A data type capable of holding a 64-bit integer
- **native** – Signifies that a method has been implemented with native code
- **new** – Used for creating new objects
- **null** – Denotes that a reference is not being used to refer to anything.
- **package** – Used for declaring a Java package
- **private** – Access specifier denoting that a variable or method can be accessed only within the class it is declared in
- **protected** – Access specifier denoting that a variable or method can be accessed only within the class it is declared in, or within the subclass, or in other package classes.
- **public** – Access specifier for interfaces, methods, variables and classes, denoting that a specified item can be accessed throughout an application.
- **return** – Sends out a return value and control back from a called method
- **short** – A data type capable of holding a 16-bit integer
- **static** – Denotes that a method or variable is a class method
- **strictfp** – Used for restricting the rounding and precision of floating point calculations, to facilitate portability.
- **super** – Refers to the base class of a class
- **switch** – Statement for executing code based on test values
- **synchronized** – Defines methods or critical sections within multithreaded code
- **this** – Refers to the current object in a constructor or method
- **throw** – Used for creating a programming exception
- **throws** – Denotes what exceptions can be thrown by a method
- **transient** – Defines that a variable does not belong to an object's persistent state
- **try** – Initiates a block of code for exception testing
- **void** – Specifies the absence of a return value in a method
- **volatile** – Denotes possible asynchronous change of a variable may change
- **While** – Used for starting a while loop

Statements

A Java statement is an instruction that tells the language what actions to perform. It specifies a Java program action, such as printing standard output messages, or writing data on a file.

A Java statement is a full or complete command to be executed by the Java interpreter. Statements run in the order the order they are written in. To get a better idea of Java statements, take a look at the images below.

An example of a Java statement (image source:

```
public class IfThenElseExample {

    public static void main(String[] args) {

        int examScore = 82;
        char grade;

        if (examScore >= 90){
            grade = 'A';
        }
        else if (examScore >= 80){
            grade = 'B';
        }
        else if (examScore >= 70){
            grade = 'C';
        }
        else if (examScore >= 60){
            grade = 'D';
        }
        else {
            grade = 'F';
        }

        System.out.println("The grade is" + grade);

    }

}
```

Statements in Java can be broadly classified into three categories:

Declaration statement – Used to declare a variable

Example –

```
int num;
int num2 = 50;
String str;
```

Expression statement – Has a semicolon at its end

```
/Increment and decrement expressions
num++;
++num;
num--;
--num;

//Assignment expressions
num = 100;
num *= 10;

//Method invocation expressions
System.out.println('This is a statement');
someMethod(param1, param2);
```

Control flow statement – Used for executing statements repeatedly for specific conditions being true

```
if (expression) {
    // statements
}
```

Identifiers

A Java identifier is a name assigned to a class, method, variable, interface or package. It enables programmers to refer items from other sections of a program.

An identifier is a character sequence that contains letters (**A-Z, a-z**), numerical digits (**0-9**), underscore (**_**) or the dollar sign (**%**). The length can range from one to several characters.

Identifiers can only start with letters, a dollar sign or an underscore. Identifiers do not start with digits and do not have tabs or spaces.

Example of identifier

```
public class Teacher
        {
        public static void main(String[] args)
        {
        int number = 5;




        }
        }
```

Identifiers in the example are - Teacher, main, String, number and args.

Literals

A fixed value source code representation is known as a literal. In Java, literals are character sequences comprising of letters, digits and other character types. Literals represent any exact value that is typed and are used along with variables within a Java statement.

Types of Java literals include -

* Integer literals
* Floating literals
* Character literals
* String literals
* Boolean literals

4.2 Data Types (Primitive & Non-primitive)

Data type in Java specifies the type and size of values that can be stored within an identifier. The two data types are -

i) Primitive Data Type

Primitive data types are the basic data types of the Java language. Primitive data has single values and no special capabilities. The eight types of primitive data in Java are - boolean, byte, char, double, float, int, long, and short. The size of primitive data types remains constant regardless of operating systems.

ii) Non Primitive Data Type

Non-primitive data types refer to objects and are also known as reference types. This type of data includes strings, classes, interfaces and arrays. Non-primitive data is created by a programmer during the process of programming. Hence, it is user-defined. Some examples are structure, link list, union, queue and array.

Operators

Pre increment operator - In a pre-increment operator, the increment operator(++) is placed before the variable name (operand). A value has to be incremented first, before it can be used in an expression.

Post increment operator - In a post-increment operation, the increment operator(++) is placed after the variable name (operand). Original operand value is read first, and then value is incremented.

Code example of Pre and post increment operators -

```
class Increment
{
   public static void main(String arg[])
   {
      int p = 5;
      int q = 2;
      int r;
```

```
    int s;
    r = ++q;
    s = p++;
    r++;
    System.out.println('p = ' + p + ' q = ' + q + ' r = ' + r + ' s = ' + s);

  }
}
```

Output

p = 6 q = 3 r = 4 s = 5

Pre increment operator	Post increment operator
Pre increment operator increments variable value by 1, prior to variable value assignment.	Post increment operator increments variable value by 1, following variable value assignment.

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. What are Java keywords also known as?

a) Coding words

b) Reserved words

c) Array words

d) None of the mentioned

2. The total number of Java keywords is ___.

a) 50

b) 60

c) 30

d) 10

3. Byte is a data type that can hold ___ bit data values.

a) 14

b) 8

c) 64

d) 16

4. Else is used for denoting alternative branches of an ___ statement.

a) if-or

b) or

c) if

d) None of the mentioned

5. Short is a data type that holds a __ bit integer.

a) 32

b) 16

c) 8

d) 12

6. The throw keyword is used for creating a programming _____

a) handler

b) protocol

c) exception

d) None of the mentioned

7. Transient defines that a variable is not a part of an object's _____ state.

a) persistent

b) predominant

c) pre-existing

d) None of the mentioned

8. Void specifies the absence of a _____ value in a method.

a) reboot

b) return

c) replaced

d) None of the mentioned

9. What type of value does Primitive data have?

a) single

b) double

c) triple

d) multiple

10. Boolean is a _____ data type.

a) table

b) exclusive

c) non-primitive

d) primitive