DEPARTMENT OF COMMERCE (CA)
JAVA PROGRAMMING AND HTML (Semester-IV)
II M.COM (CA) Sub Code-18MCC42C
UNIT – II

Constants-Variables-Data Types-Arithmetic, relational, logical, assignment operators if, if…else, else…if ladder-while, do, for-jumps in loops-Defining a class-Creating objects Method declaration-fields declaration.

CONSTANTS

A constant is a variable whose value cannot change once it has been assigned. Java doesn't have built-in support for constants. A constant can make our program more easily read and understood by others. In addition, a constant is coached by the JVM as well as our application, so using a constant can improve performance. To define a variable as a constant, we just need to add the keyword “final” in front of the variable declaration. Constants by just adding the keyword “final” when we declare the variable.

Example

Unlike variables, constants cannot change their initial values. Once initialized, they cannot be modified. Constants are created with the final keyword.

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. There are two types of data types in Java: primitive and non-primitive.

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.

TYPES OF VARIABLES

There are three types of variables in Java:

1. Local variable
2. Instance variable
3. Static variable

1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists. A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static. It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

DATA TYPES IN JAVA

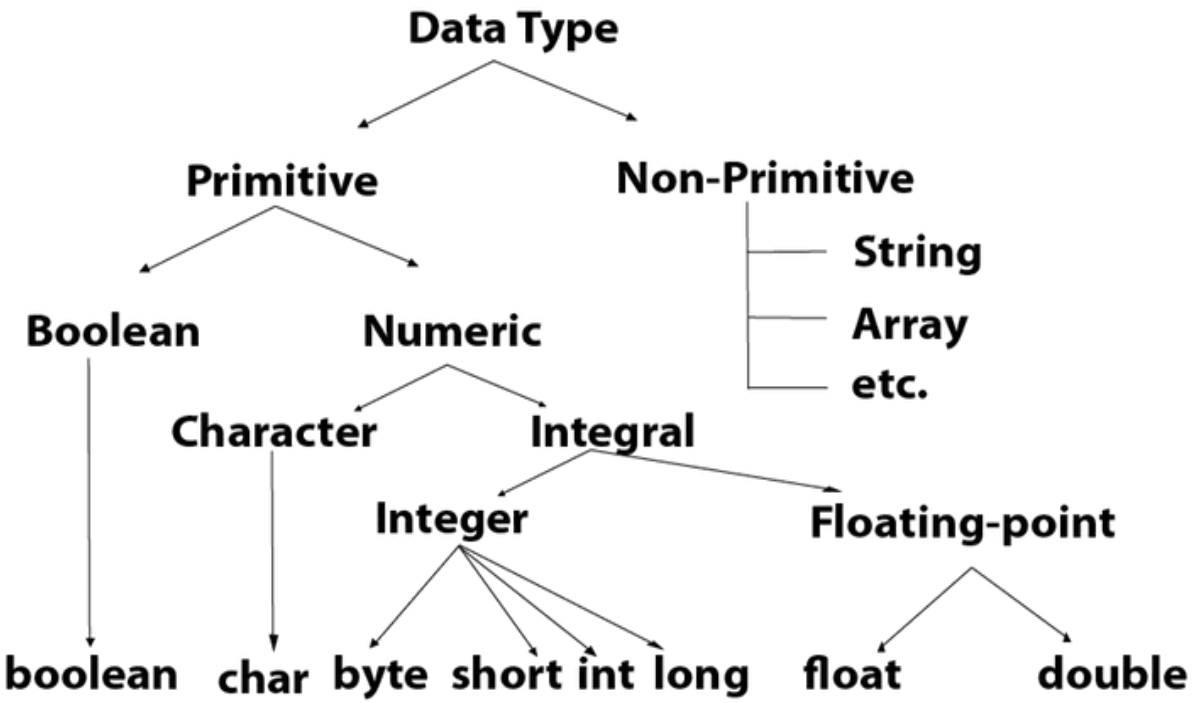
Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1.Primitive data types: The primitive data types include Boolean, char, byte, short, int, long, float and double.

2.**Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language. Java is a statically -type programming language. It means all variable must be declared before its use.t hats because we need to declare variables types and name.



Data Type	Default Value	Default Size
Boolean	false	1 bit
char	'\u0000'	2 bytes

byte	0	1 byte
Short	0	2 bytes
int	0	4 bytes
long	0L	8 bytes
float	0.0f	4 bytes
double	0.0d	8 bytes

There are 8 types of primitive data types:

1, Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions. The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: Boolean one = false

2. Byte Data Type

The byte data type is an example of primitive data type. It is an 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0. The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

Example: byte a = 10, byte b = -20

3.Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0. The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example: short s = 10000, short r = -5000

4.Int Data Type

The int data type is a 32-bit signed two's complement integer. Its value-range lies between -2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive). Its minimum value is -2,147,483,648and maximum value is 2,147,483,647. Its default value is 0. The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

Example: int a = 100000, int b = -200000

5.Long Data Type

The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 -1) (inclusive). Its minimum value is - 9,223,372,036,854,775,808and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example: long a = 100000L, long b = -200000L

6.Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in

large arrays of floating-point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.
Example: float f1 = 234.5f

7.Double Data Type
The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.
Example: double d1 = 12.3

8.Char Data Type
The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive). The char data type is used to store characters.
Example: char letterA = 'A'

OPERATORS

Arithmetic Operators - Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication and division. There are five arithmetic operators available in C (+, -, *, /, %). All arithmetic operators are binary operators, i.e.; they operate on two operands to produce the result.

Assignment Operator - Assignment operator assigns value of the expression on the right side to left side variable. The base assignment operator is '='. In C, you can use this operator like the following variable = expression Here variable can be any kind of a variable and expression can be a simple constant, another variable or may be a more complex expression, like a formula. The value of the expression will be evaluated and assigned to the variable. But there are some things to note about the assignment operator.

If the value of the expression on the right side is not same data type as variable on the left, then the value of the expression will be converted to same data type as the variable. Suppose, the expression is a floating-point number and the variable is an integer. Then, the floating-point number will be converted to integer and then will be assigned to the variable. It is better to use same data types on both sides unless it is a must, else it may lead to information loss.

Relational operator - Relational operators are binary operators (operates on two operands) and are used to relate or compare two operands. There are four relational operators in C (i.e., <, <=, >, >=). If the relationship between the operands is correct, it will return 1 and returns 0 otherwise. Apart from four relational operators, C has two equality operators (== and! =) as well for comparing operands. Now let's take a look at different relational and equality operators and how they operate on the operands.

Logical operator - We've learned about relational operators, using which we can do comparisons. What if you need to combine two relational expressions, for example, if you want to check if a number is greater than 10 and less than 20? You can use logical operators for that purpose. There are three logical operators in C language, &&.

IF STATEMENT

The syntax of an if-then statement:

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

Here, condition is a Boolean expression. It returns either true or false.

if condition evaluates to true, statements inside the body of if are executed
if condition evaluates to false, statements inside the body of if are skipped

IF...ELSE STATEMENT

The if statement executes a certain section of code if the test expression is evaluated to true. However, if the test expression is evaluated to false, it does nothing. In this case, we can use an optional else block. Statements inside the body of else block are executed if the test expression is evaluated to false. This is known as the if-...else statement in Java.

The syntax of the if...else statement is:

```
if (condition) {
  // codes in if block
}
else {
  // codes in else block
}
```

Here, the program will do one task (codes inside if block) if the condition is true and another task (codes inside else block) if the condition is false.

IF...ELSE...IF STATEMENT

In Java, we have an if...else...if ladder, that can be used to execute one block of code among multiple other blocks.

```
if (condition1) {
  // codes
}
else if(condition2) {
  // codes
}
else if (condition3) {
  // codes
}
.
.
else {
  // codes
}
```

Here, if statements are executed from the top towards the bottom. When the test condition is true, codes inside the body of that if block is executed. And, program control jumps outside the if...else...if ladder.

ELSE…. IF LADDER

Java if-else-if ladder is used to decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:

```
if (condition)
   statement 1;
else if (condition)
   statement 2;
.
.
else
   statement;
```

WHILE

The while loop loops through a block of code as long as a specified condition is true:

Syntax

```
while (condition) {
  // code block to be executed
}
```

DO/WHILE

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

FOR

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
```

}
Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

JUMPS IN LOOPS

The jumping statements are the control statements which transfer the program execution control to a specific statement. Java has three types of jumping statements they are break, continue, and return. These statements transfer execution control to another part of the program.

DEFINING A CLASS

A class--the basic building block of an object-oriented language such as Java--is a template that describes the data and behaviour associated with instances of that class. When you instantiate a class, you create an object that looks and feels like other instances of the same class. The data associated with a class or object is stored in variables; the behaviour associated with a class or object is implemented with methods. Methods are similar to the functions or procedures in procedural languages such as C.

CREATING OBJECT

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.
There are three steps when creating an object from a class −

- Declaration − A variable declaration with a variable name with an object type.
- Instantiation − The 'new' keyword is used to create the object.

- Initialization − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

METHOD DECLARATION

A method's declaration provides a lot of information about the method to the compiler, the runtime system and to other classes and objects. Besides the name of the method, the method declaration carries information such as the return type of the method, the number and type of the arguments required by the method, and what other classes and objects can call the method.
While this may sound like writing a novel rather than simply declaring a method for a class, most method attributes can be declared implicitly. The only two required elements of a method declaration are the method name and the data type returned by the method. For example, the following declares a method named is Empty in the Stack class that returns a Boolean value (true or false):

FIELD DECLARATION

Syntax

A Java field is declared using the following syntax:

[access_modifier] [static] [final] type name [= initial value] ;

The square brackets [] around some of the keywords mean that this option is optional. Only type and name are required.

First an access modifier can be declared for a Java field. The access modifier determines which object classes that can access the field. In the Employee example above there were no access modifiers.

Second, a data type for the Java field must be assigned. In the Employee example above the data types String, int and Date were used.

Third, the Java field can be declared static. In Java, static fields belong to the class, not instances of the class. Thus, all instances of any class will access the same static field variable. A non-static field value can be different for every object (instance) of a class.

Fourth, the Java field can be declared final or not. A final field cannot have its value changed. A final field must have an initial value assigned to it, and once set, the value cannot be changed again. A final field is often also declared static. A field declared static and final is also called a "constant".

Fifth, the Java field is given a name. You can choose this name freely, but there are some restrictions on what characters the name can contain.

Sixth, you can optionally set an initial value for the field.

REFERENCE

1. World Wide Web design with HTML - C. Xavier TMH Publications, 2000.
2. Programming with java2- Xavier, SciTech Publications, 2000.
3. https://www.javatpoint.com/operators-in-java
4. https://www.w3schools.com/java/java_data_types.asp

PREPARED BY Dr.N.SHANMUGAVADIVU