Data types

Data type specifies the size and type of values that can be stored in an identifier. The Java language is rich in its data types. Different data types allow you to select the type appropriate to the needs of the application.

Data types in Java are classified into two types:

1. Primitive—which include Integer, Character, Boolean, and Floating Point.
2. Non-primitive—which include Classes, Interfaces, and Arrays.

Primitive Data Types

1. Integer

Integer types can hold whole numbers such as 123 and −96. The size of the values that can be stored depends on the integer type that we choose.

Type	Size	Range of values that can be stored
byte	1 byte	−128 to 127
short	2 bytes	−32768 to 32767
int	4 bytes	−2,147,483,648 to 2,147,483,647
long	8 bytes	9,223,372,036,854,775,808 to 9,223,372,036,854,755,807

The range of values is calculated as $-(2^{n-1})$ to $(2^{n-1})-1$; where n is the number of bits required. For example, the byte data type requires 1 byte = 8 bits. Therefore, the range of values that can be stored in the byte data type is $-(2^{8-1})$ to $(2^{8-1})-1$
= -2^7 to (2^7) -1
= −128 to 127

2. Floating Point

Floating point data types are used to represent numbers with a fractional part. Single precision floating point numbers occupy 4 bytes and Double precision floating point numbers occupy 8 bytes. There are two subtypes:

Type	Size	Range of values that can be stored
float	4 bytes	3.4e−038 to 3.4e+038
double	8 bytes	1.7e−308 to 1.7e+038

3. Character

It stores character constants in the memory. It assumes a size of 2 bytes, but basically it can hold only a single character because char stores unicode character sets. It has a minimum value of ‘u0000’ (or 0) and a maximum value of ‘uffff’ (or 65,535, inclusive).

4. Boolean

Boolean data types are used to store values with two states: true or false.

Java Tokens

A token is the smallest element in a program that is meaningful to the compiler. These tokens define the structure of the language. The Java token set can be divided into five categories: Identifiers, Keywords, Literals, Operators, and Separators.

1. Identifiers

Identifiers are names provided by you. These can be assigned to variables, methods, functions, classes etc. to uniquely identify them to the compiler.

2. Keywords

Keywords are reserved words that have a specific meaning for the compiler. They cannot be used as identifiers. Java has a rich set of keywords. Some examples are: boolean, char, if, protected, new, this, try, catch, null, threadsafe etc.

3. Literals

Literals are variables whose values remain constant throughout the program. They are also called Constants. Literals can be of four types. They are:

a. String Literals

String Literals are always enclosed in double quotes and are implemented using the java.lang.String class. Enclosing a character string within double quotes will automatically create a new String object. For example, `String s = "this is a string";`. String objects are immutable, which means that once created, their values cannot be changed.

b. Character Literals

These are enclosed in single quotes and contain only one character.

c. Boolean Literals

They can only have the values `true` or `false`. These values do not correspond to 1 or 0 as in C or C++.

d. Numeric Literals

Numeric Literals can contain integer or floating point values.

4. Operators

An operator is a symbol that operates on one or more operands to produce a result.

They will be discussed in greater detail in the next article.

5. Separators

Separators are symbols that indicate the division and arrangement of groups of code. The structure and function of code is generally defined by the separators. The separators used in Java are as follows:

parentheses ( )

Used to define precedence in expressions, to enclose parameters in method definitions, and enclosing cast types.

braces { }

Used to define a block of code and to hold the values of arrays.

brackets [ ]

Used to declare array types.

semicolon ;

Used to separate statements.

comma ,

Used to separate identifiers in a variable declaration and in the `for` statement.

period .

Used to separate package names from classes and subclasses and to separate a variable or a method from a reference variable.

Variables

There are different types of variables in Java. They are as follows:

1. Instance Variables (Non-Static Fields)

Objects store their individual states in “non-static fields”, that is, fields declared without the `static` keyword.

Non-static fields are also known as instance variables because their values are unique to each instance of a class. For example, the `currentSpeed` of one bicycle is independent from the `currentSpeed` of another.

2. Class Variables (Static Fields)

A class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated. A field defining the number of gears for a particular kind of bicycle could be marked as static since, conceptually, the same number of gears will apply to all instances. The code `static int numGears = 6;`would create such a static field.

3. Local Variables

A method stores its temporary state in local variables. The syntax for declaring a local variable is similar to declaring a field (for example, `int count = 0;`). There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared—between the opening and closing

braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.

4. Parameters

They are the variables that are passed to the methods of a class.

Variable Declaration

Identifiers are the names of variables. They must be composed of only letters, numbers, the underscore, and the dollar sign ($). They cannot contain white spaces. Identifiers may only begin with a letter, the underscore, or the dollar sign. A variable cannot begin with a number. All variable names are case sensitive.

Syntax for variable declaration

datatype1 variable1, datatype2 variable2, … datatypen variablen;

For example:

```
int a, char ch;
```

Initialisation

Variables can be assigned values in the following way: `Variablename = value;`

For example;

```
ch='a';
a=0;
```

Arrays

An array is a group of variables that share the same data type, and are referred to by a common name. Arrays of any type can be created and may have one or more dimensions.

A specific element in an array is accessed by its index. The array index ranges from 0 to n−1; therefore, in an array of size 10, the first element is stored at index 0 and the last or the 10th element at index 9.

The following program, `Printarr`, creates an array of integers, puts some values in it, and prints each value to standard output.

```
class Printarr {
  public static void main(String[] args) {
    // declares an array of integers
    int[ ] A;

    // allocates memory for 5 integers
    A = new int[5];

    // initialize elements
    A[0] = 15;//first element

    A[1] = 20;//second element

    A[2] = 25;//third element

    A[3] = 30;//fourth element

    A[4] = 50;//fifth element

    System.out.println("Element at index 0: "
                       + A[0]);
    System.out.println("Element at index 1: "
                       + A[1]);
    System.out.println("Element at index 2: "
                       + A[2]);
    System.out.println("Element at index 3: "
                       + A[3]);
    System.out.println("Element at index 4: "
                       + A[4]);
  }
}
```

The output from this program is:

```
Element at index 0: 15
Element at index 1: 20
Element at index 2: 25
Element at index 3: 30
Element at index 4: 50
```

Copying Arrays

The data from one array can be copied into another array by using the `arraycopy`method of the System class:

```
public static void arraycopy(Object src,
                         int srcPos,
                         Object dest,
                         int destPos,
                         int length)
```

The two Object arguments specify the array to copy from, and the array to copy to. The three int arguments specify the starting position in the source array, the starting position in the destination array, and the number of array elements to copy.

The following program, `Copyarr`, declares an array of char elements, spelling the word “array”. It uses `arraycopy` to copy three elements of the first array into the second array:

```
class Copyarr {
  public static void main(String[] args) {
    char[] source = { 'a', 'r', 'r', 'a', 'y' };
    char[] target = new char[3];

    System.arraycopy(source, 0, target, 0, 3);
    System.out.println(new String(target));
  }
}
```

The output from this program is:

```
arr
```

Summary

- Java is a pure, object oriented language introduced by Sun Microsystems of USA, and has a number of characteristics that make it suitable for Internet programming.
- The platform independent feature of Java is achieved through bytecode.
- The eight primitive data types are: byte, short, int, long, float, double, boolean, and char. The java.lang.String class represents character strings.
- Instance variables (non-static fields) are unique to each instance of a class.
- Class variables (static fields) are fields declared with the `static` modifier; there is exactly one copy of a class variable, regardless of how many times the class has been instantiated.

- Local variables store temporary state inside a method.

The next tutorial will talk about Operators, Expressions Statements, and Blocks. Operators may be used in building expressions, which compute values; expressions are the core components of statements; statements may be grouped into blocks of code.

Learn more about Java with our tutorial Discovering the Differences Between Blocks, Procs and Lambdas on SitePoint.