Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Java Variables

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

- `String` - stores text, such as "Hello". String values are surrounded by double quotes
- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `float` - stores floating point numbers, with decimals, such as 19.99 or -19.99
- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `boolean` - stores values with two states: true or false

Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax

```
type variableName = value;
```

Where *type* is one of Java's types (such as `int` or `String`), and *variableName* is the name of the variable (such as **x** or **name**). The **equal sign** is used to assign values to the variable.

Final Variables

If you don't want others (or yourself) to overwrite existing values, use the `final` keyword (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

Example

final num=15;

Declare Many Variables

To declare more than one variable of the **same type**, you can use a comma-separated list:

Instead of writing:

```
int x = 5;
int y = 6;
int z = 50;
System.out.println(x + y + z);
```

You can simply write:

```
int x = 5, y = 6, z = 50;
System.out.println(x + y + z);
```

One Value to Multiple Variables

You can also assign the **same value** to multiple variables in one line:

```
int x, y, z;
x = y = z = 50;
System.out.println(x + y + z);
```

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Identifiers

All Java **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

```
// Good
int minutesPerHour = 60;


// OK, but not so easy to understand what m actually is
int m = 60;
```

The general rules for naming variables are:

- Names can contain letters, digits, underscores, and dollar signs
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like Java keywords, such as `int` or `boolean`) cannot be used as names

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Java Data Types

As explained in the previous chapter, a variable in Java must be a specified data type:

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99f;    // Floating point number
char myLetter = 'D';         // Character
boolean myBool = true;       // Boolean
String myText = "Hello";     // String
```

Data types are divided into two groups:

- Primitive data types -
  include `byte`, `short`, `int`, `long`, `float`, `double`, `boolean` and `char`

- Non-primitive data types - such as `String`, Arrays and Classes (you will learn more about these in a later chapter)

Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

There are eight primitive data types in Java:

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Numbers

Primitive number types are divided into two groups:

Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimals. Valid types are `byte`, `short`, `int` and `long`. Which type you should use, depends on the numeric value.

Floating point types represent numbers with a fractional part, containing one or more decimals. There are two types: `float` and `double`.

Even though there are many numeric types in Java, the most used for numbers are `int` (for whole numbers) and `double` (for floating point numbers). However, we will describe them all as you continue to read.

Integer Types

Byte

The `byte` data type can store whole numbers from -128 to 127. This can be used instead of `int` or other integer types to save memory when you are certain that the value will be within -128 and 127:

```
byte myNum = 100;

System.out.println(myNum);
```

Short

The `short` data type can store whole numbers from -32768 to 32767:

```
short myNum = 5000;

System.out.println(myNum);
```

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

Int

The `int` data type can store whole numbers from -2147483648 to 2147483647. In general, and in our tutorial, the `int` data type is the preferred data type when we create variables with a numeric value.

```
int myNum = 100000;

System.out.println(myNum);
```

Long

The `long` data type can store whole numbers from -9223372036854775808 to 9223372036854775807. This is used when int is not large enough to store the value. Note that you should end the value with an "L":

```
long myNum = 15000000000L;

System.out.println(myNum);
```

Floating Point Types

You should use a floating point type whenever you need a number with a decimal, such as 9.99 or 3.14515.

The `float` and `double` data types can store fractional numbers. Note that you should end the value with an "f" for floats and "d" for doubles:

Float Example

```
float myNum = 5.75f;

System.out.println(myNum);
```

Double Example

```
double myNum = 19.99d;

System.out.println(myNum);
```

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Use `float` or `double`?

The **precision** of a floating point value indicates how many digits the value can have after the decimal point. The precision of `float` is only six or seven decimal digits, while `double` variables have a precision of about 15 digits. Therefore it is safer to use `double` for most calculations.

Scientific Numbers

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

Example

```
float f1 = 35e3f;

double d1 = 12E4d;

System.out.println(f1);

System.out.println(d1);
```

Boolean Types

Very often in programming, you will need a data type that can only have one of two values, like:

- YES / NO
- ON / OFF
- TRUE / FALSE

For this, Java has a `boolean` data type, which can only take the values `true` or `false`:

```
boolean isJavaFun = true;

boolean isFishTasty = false;

System.out.println(isJavaFun);     // Outputs true

System.out.println(isFishTasty);   // Outputs false
```

Boolean values are mostly used for conditional testing.

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Characters

The `char` data type is used to store a **single** character. The character must be surrounded by single quotes, like 'A' or 'c':

```
char myGrade = 'B';

System.out.println(myGrade);
```

Alternatively, if you are familiar with ASCII values, you can use those to display certain characters:

```
char myVar1 = 65, myVar2 = 66, myVar3 = 67;

System.out.println(myVar1);//A

System.out.println(myVar2);//B

System.out.println(myVar3);//C
```

Non-Primitive Data Types

Non-primitive data types are called **reference types** because they refer to objects.

The main difference between **primitive** and **non-primitive** data types are:

- Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for `String`).
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be `null`.
- A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
- The size of a primitive type depends on the data type, while non-primitive types have all the same size.

Phone
+91 8767591462| +91 7822869695

Website
www.payastechnologies.com

Strings

The `String` data type is used to store a sequence of characters (text). String values must be surrounded by double quotes:

```
String greeting = "Hello World";

System.out.println(greeting);
```