

Java

Data Types

Data types are divided into two groups:

- Primitive data types - includes byte, short, int, long, float, double, boolean and char
- Non-primitive data types - such as String, Arrays and Classes

VARIABLES IN JAVA

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type. In Java there are four types of variables:

- Non-static fields
- Static fields
- Local variables
- Parameters

A non-static field is a variable that belongs to an object. Objects keep their internal state in non-static fields. Non-static fields are also called instance variables, because they belong to instances (objects) of a class. Non-static fields are covered in more detail in the text on [Java fields](#).

A static field is a variable that belongs to a class. A static field has the same value for all objects that access it. Static fields are also called class variables. Static fields are also covered in more detail in the text on [Java fields](#).

A local variable is a variable declared inside a method. A local variable is only accessible inside the method that declared it. Local variables are covered in more detail in the text on [Java methods](#).

A parameter is a variable that is passed to a method when the method is called. Parameters are also only accessible inside the method that declares them, although a value is assigned to them when the method is called. Parameters are also covered in more detail in the text on [Java methods](#).

VARIABLE TYPES

In Java there are four types of variables:

- Non-static fields
- Static fields
- Local variables
- Parameters

A non-static field is a variable that belongs to an object. Objects keep their internal state in non-static fields. Non-static fields are also called instance variables, because they belong to instances (objects) of a class. Non-static fields are covered in more detail in the text on [Java fields](#).

A static field is a variable that belongs to a class. A static field has the same value for all objects that access it. Static fields are also called class variables. Static fields are also covered in more detail in the text on [Java fields](#).

A local variable is a variable declared inside a method. A local variable is only accessible inside the method that declared it. Local variables are covered in more detail in the text on [Java methods](#).

A parameter is a variable that is passed to a method when the method is called. Parameters are also only accessible inside the method that declares them, although a value is assigned to them when the method is called. Parameters are also covered in more detail in the text on [Java methods](#).

JDK VS JRE VS JVM

1. JDK

Java Development Kit aka JDK is the core component of Java Environment and provides all the tools, executables, and binaries required to compile, debug, and execute a Java Program.

JDK is a platform-specific software and that's why we have separate installers for Windows, Mac, and Unix systems.

We can say that JDK is the superset of JRE since it contains JRE with Java compiler, debugger, and core classes.

JVM

JVM is the heart of Java programming language. When we execute a Java program, JVM is responsible for converting the byte code to the machine-specific code.

JVM is also platform-dependent and provides core java functions such as memory management, garbage collection, security, etc.

JVM is customizable and we can use java options to customize it. For example, allocating minimum and maximum memory to JVM.

JVM is called **virtual** because it provides an interface that does not depend on the underlying operating system and machine hardware.

This independence from hardware and the operating system makes java program write-once-run-anywhere.

JRE

JRE is the implementation of JVM. It provides a platform to execute java programs. JRE consists of JVM, Java binaries, and other classes to execute any program successfully.

JRE doesn't contain any development tools such as Java compiler, debugger, JShell, etc.

If you just want to execute a java program, you can install only JRE. You don't need JDK because there is no development or compilation of java source code is required.

Now that we have a basic understanding of JDK, JVM, and JRE, let's look into the difference between them.

Let's look at some of the important differences between JDK, JRE, and JVM.

1. JDK is for development purpose whereas JRE is for running the java programs.
2. JDK and JRE both contains JVM so that we can run our java program.
3. JVM is the heart of java programming language and provides platform independence.

Loops

FOR Loop

The Java for loop is a control flow statement that iterates a part of the programs multiple times.

WHEN TO USE FOR LOOP

If the number of iteration is fixed, it is recommended to use for loop.

SYNTAX:

```
for(init;condition;incr/decr){
```

```
// code to be executed
```

```
}
```

WHILE

The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.

WHEN TO USE WHILE LOOP

If the number of iteration is not fixed, it is recommended to use while loop.

SYNTAX

```
while(condition){
```

```
//code to be executed
```

```
}
```

DO-WHILE

The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.

WHEN TO USE DO WHILE

If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.

SYNTAX

```
do{
```

```
//code to be executed
```

```
}while(condition);
```