Apache NetBeans Java EE & Java Web Learning Trail

<https://netbeans.apache.org/kb/docs/java-ee.html>

Java Tutotial from NetBeans

<https://netbeans.apache.org/kb/docs/java/>

## [First Cup of Java EE 8 Tutorial](https://javaee.github.io/firstcup/)

<https://javaee.github.io/firstcup/>

<https://javaee.github.io/firstcup/intro002.html#software-compatibility>

<https://javaee.github.io/firstcup/java-ee002.html>

## The Java EE 8 Tutorial

<https://javaee.github.io/tutorial/>

<https://javaee.github.io/tutorial/overview004.html>

**Servlets** are Java programming language classes that dynamically process requests and construct responses.

<https://javaee.github.io/tutorial/overview005.html>

**Containers** are the interface between a component and the low-level, platform-specific functionality that supports the component.

## Java SE 8 Doc(<https://docs.oracle.com/javase/8/>)

Java 代码规范

<https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html>

***Class names*** should be ***nouns***, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use ***whole words***-avoid acronyms and abbreviations.

***Methods*** should be ***verbs***, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

## The Java Tutorials(<https://docs.oracle.com/javase/tutorial/>)

<https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS.

<https://docs.oracle.com/javase/tutorial/java/concepts/interface.html>

In its most common form, an interface is a group of related methods with empty bodies.

Implementing an interface allows a class to become more formal about **the behavior it promises to provide**.

Interfaces form a **contract** between the class and the outside world, and this contract is **enforced at build time by the compiler**.

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>

A type's fields, methods, and nested types are collectively called its members.

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/index.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html>

Parameters refers to the list of variables in a method **declaration**. Arguments are the **actual values** that are passed in when the method is invoked. When you invoke a method, the **arguments** used must match the declaration's **parameters** in type and order.

A parameter can have the same name as one of the class's fields. If this is the case, the parameter is said to shadow the field. **Shadowing fields can make your code difficult to read** and is conventionally used only within constructors and methods that set a particular field.

**Reference data type parameters**, such as objects, are also passed into methods by **value**. This means that when the method returns, the passed-in reference still references **the same object as before**. However, **the values of the object's fields can be changed in the method**, if they have the proper access level.

<https://docs.oracle.com/javase/tutorial/java/javaOO/usingobject.html>

The object is **unreferenced**, and its resources are free to be recycled by the Java Virtual Machine.

Remember, invoking a method on a particular object is the same as sending a message to that object.

An object is eligible for garbage collection when there are no more references to that object.

References that are held in a variable are usually dropped when the variable goes out of scope. Or, you can explicitly drop an object reference by setting the variable to the special value null.

<https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/thiskey.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/localclasses.html>

？

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

**Collections** are used to store, retrieve, manipulate, and communicate aggregate data.

接口的使用技巧，逐步精简代码

A stream is a sequence of elements. Unlike a collection, it is not a data structure that stores elements. Instead, a stream carries values from a source, such as collection, through a pipeline.

<https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html>

<https://docs.oracle.com/javase/tutorial/java/annotations/basics.html>

The annotation type can be one of the types that are defined in the java.lang or java.lang.annotation packages of the Java SE API. It is also possible to define your own annotation type.

Annotations can **be applied to declarations**: declarations of classes, fields, methods, and other program elements. When used on a declaration, each annotation often appears, by convention, on its own line.

As of the Java SE 8 release, annotations can also **be applied to the use of types**.

While it is not required to use this annotation when overriding a method, it helps to prevent errors. If a method marked with @Override fails to correctly override a method in one of its superclasses, the compiler generates an error.

<https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html>

Access level modifiers determine whether other classes can use a particular field or invoke a particular method. There are two levels of access control:

At the top level—**public**, o**r package-priv**ate (no explicit modifier).

A class may be declared with the modifier public, in which case that class is visible to all classes everywhere. If a class has no modifier (the default, also known as package-private), it is visible only within its own package (packages are named groups of related classes — you will learn about them in a later lesson.)

At the member level—**public**, **private**, **protected**, or **package-private** (no explicit modifier).

At the member level, you can also use the public modifier or no modifier (package-private) just as with top-level classes, and with the same meaning. For members, there are two additional access modifiers: private and protected. The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package (as with package-private) and, in addition, by **a subclass of its class in another package**.

<https://docs.oracle.com/javase/tutorial/java/IandI/interfaceDef.html>

All abstract, default, and static methods in an interface are implicitly **public**, so you can omit the public modifier.

All constant values defined in an interface are implicitly public, static, and final.

<https://docs.oracle.com/javase/tutorial/java/IandI/usinginterface.html>

The line of code, shown in bold in the previous example, casts other to a RectanglePlus instance. **Type casting** tells the compiler what the object really is. Invoking getArea directly on the other instance (other.getArea()) would fail to compile because the compiler does not understand that other is actually an instance of RectanglePlus.

<https://docs.oracle.com/javase/tutorial/java/IandI/interfaceAsType.html>

If you define a reference variable whose type is an interface, any object you assign to it must be an instance of a class that implements the interface.

<https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html>

Static Methods

## FQA

<https://askinglot.com/what-is-a-java-type>

**Likewise, people ask, what does type mean in Java?**

The Java programming language is a statically typed language, which means that every variable and every expression has a type that is known at compile time.

**What is class type in Java?**

Java classes, as you probably already know, are created to solve object oriented problems. **Any object of a class has the type “class”.** Because an object of a class is more complex than a simple integer, boolean, or other “primitive” type, a variable naming an object is known to be a class type.

**Likewise, people ask, what does type mean in Java?**

There are two types of data types in Java:

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

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

<https://www.quora.com/What-skills-do-you-need-to-become-a-Java-web-developer>

**What skills do you need to become a Java web developer?**