<BEGINNERS PROGRAMMING GUIDE>

JAVA

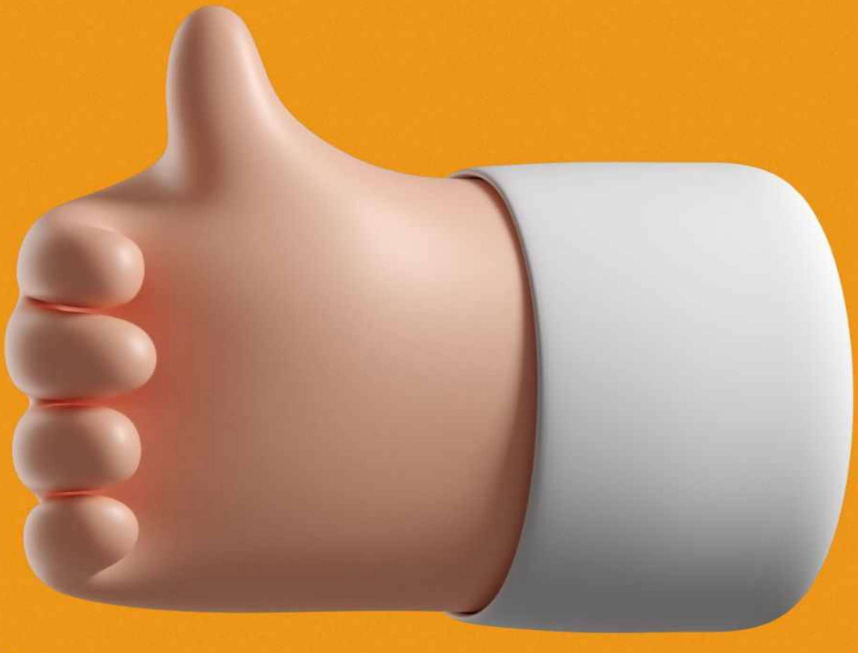
<33 BEST JAVA TIPS AND TRICKS>

2021

<BEGINNERS
PROGRAMMING GUIDE>

JAVA

<33 BEST JAVA
TIPS AND TRICKS>



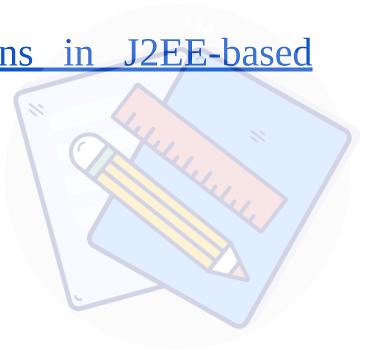
2021

Java: 2021 Beginners Programming Guide. 33 Best Java Tips and Tricks

CONTENTS:

Introduction
- What is the Java language?
- Why is Java such a popular programming language?
- What is the Java programming language?
- The "cache issue"
- Choosing the right editor
- What are the advantages of the Java language?
- What are the drawbacks of the Java language?
- Which programming languages can be used in the Oracle Digital Cloud Office?
- How to install Java on the Oracle Digital Cloud Office?
- Wrapping it up

Chapter 1 What are the features of the Java language?
- Are there any flaws in the Java language design?
- Are there any design flaws in the J2EE application server design?
- Are the versions specified by the Java Language Specification (JLS) to be the final versions, or will they be revised in some way?
- Should I write J2EE applications using the J2EE application server specification?
- How can a J2EE server be used?
- How does a J2EE client know how to create a program to communicate with the server?
- What J2EE features and standard J2EE servers have been defined by the J2EE specification?
- How does the JavaServer interface work?
- Can Java EE 6 use J2EE?
- Can Java EE 6 use J2EE libraries?
- Does the JavaServer implementation interface behave differently in Java EE 6 than in Java EE 5?
- Does Java EE 6 specify the way that J2EE code is compiled and run?



- Is it possible to deploy Java EE 6 applications in J2EE-based configuration?
- File System Component
- Database Component Registry
- Message Component Registry
- What features for Java development?
- Notable changes in JDK 9
- ProdJ99077 New features of JDK 9
- Improved Exception Handling
- Separately Generated Type Hints
- Null Pointer Exceptions
- Optimizations
- New Inline Function Generator
- Null in type inference
- Favor closed over types
- Why the first argument matters
- What is the best IDE for Java development?

Chapter 2 What IDE chooses for Java programming?

- Eclipse IDE
- NetBeans IDE
- IntelliJ IDEA
- Novation IDE
- iDevelop IDE
- Ardu Pascal
- Bitbucket CodeSourcery
- JetBrains WebStorm
- Xamarin
- Who uses the most Java programming language?
- What is the best IDE for iOS development?
- What is the best IDE for Silverlight development?
- What is the best IDE for .NET development?
- How do you learn Java the right way?
- What are you seeing that you wish was better in Java?

- What is your favorite IDE for Java development?
- What IDE should I use? Or, what IDE should I learn first?

Chapter 3 What are plugins for Java programming?
- Java plugins for .NET
- Why should you create a Java plugin for .NET?
- But how do you create a Java plugin for .NET?
- What are the downsides of Java plugins for .NET?
- What do you need to get started?
- How do I add a plugin to my existing project?
- How do I add a source folder to my .NET project?
- How do I add a source folder to my .NET project using Eclipse?
- How do I add source folders to my .NET project using Visual Studio?

Chapter 4 Do Java and Javascript the same?
- What should I learn first?
- Java
- Some tips for Java developers
- What are frameworks for Java?
- Framework design considerations
- Aesthetics and syntax
- Scala and MVC
- MVC and Spring
- Reactive programming in a web application - A real-world case
- How to Build a Reactive system
- Connecting to a reactive database in Scala

Chapter 5 What are tools for Java language?
- Does the tooling work well?
- Does the tool provide the necessary configuration?
- Is the configuration properly organized?
- Is the configuration customizable?
- The Future
- Structural improvements

Chapter 6 How to learn Java programming language?

If you are ready to get started with learning Java programming language, follow these steps:
How to learn Java programming?
What is a constant?
What's the largest prime number?
What is a constant(T) in Java?
How do you define the type of a parameter?
How do you return the value of a parameter in a method?
What are the methods of the parameterized classes?
Do you have to create a class and implement methods with the parameters of the parameterized classes?
How to add a ParameterizedMethodBuilder to a parameterized class?
Do you have to create a custom implementation of a custom class in order to call the methods of the parameterized classes?
How to use a parameterized class as a parameterized class?
Do parameterized classes have to be subclassed?
Can I call the methods of the parameterized classes like this:
What is the difference between a weak parameterized class and a strong one?
Chapter 7 How to use Java programming libraries?
The Best of both worlds: Libraries for Java programming.
Some of the many uses for Java libraries.
Not so easy to use: Libraries require some effort
What are some of the problems you face when using Java libraries?
Scalability
Instance-based Data Structures
The best comparison that I can make is the following:
Create new trees as needed
But how could we create a tree that has four children?
But how could we create a tree with eight children?
Now, what if we wanted to create a tree with ten children?
But, our new triple-tree only has five nodes in it. But what if we wanted a double-tree with 10 nodes in it?

So, what is the most common size of the internal structure of a Tree?
What about when we don't have to create new trees, but just combine existing trees?

Chapter 8 How to find work as a Java programmer?

How to build a career as a Java developer? 4 top development tips
What is a Java developer?
What is the Java language?
Learn Java quickly
Get over the fear of implementation
Start small and learn from mistakes
Develop your ability to solve problems
Create value through partnerships and use cases
Develop strong persistence skills
Learn Java using tools
Create and present software prototypes
Follow new technology
Use cloud-based development
Get familiar with distributed programming
Watch the code
Stop saying "it will be done"
Create documents
Use pair programming
Find a mentor
Use revision control
Start learning
Keep documentation
Replace the use of save points with "hot reload"
Don't repeat yourself
Use extreme coding
Avoid the large code base
Use the linter
Use classes rather than Object Oriented Programming
Avoid virtual spaces

- Simplify code
- Avoid the keyword 'a'
- "Begs the question"
- Describe, don't ask
- Recursive
- Promises
- Use let
- Here are some useful uses of let

Chapter 9 What is Multithreading for the Java language?
- Why is the Return Operation of a Function in Java Different from the Return Operation in other Programming Languages?
- How is the for-loop in Java different from in other languages?
- What is the difference between a for-loop and a for loop?

Chapter 10 What is Java string handling?
- Java 7 String Types
- Java 8 String Operations
- Java 7 to Java 8 Transitive Data Types
- Java 7 to Java 8 Transitive Data Types
- What's New in Java 9?
- Java 9 Transitive Data Types

Chapter 11 What is Java Database Connectivity?
- Why Is It Important?
- A Little About JPA
- Why Is It Important?
- What Is JPA 3.0
- Caveat #1
- Caveat #2

Chapter 12 What is Java framework spring?
- What are the different types of spring frameworks?
- What is Spring Framework vs Spring 4?
- Spring Framework vs Spring 4.1
- What is Spring 5?

- Can you use spring-boot to solve your project issues?
- How to integrate Spring Framework?
- How to integrate spring-web
- Let’s see a detailed example of Spring Data JPA
- How to integrate spring-mvc
- What else can Spring Framework do?
- What is Spring MVC?
- PHPStorm: Spring MVC
- Creating the View
- Creating the View Method
- Load Model
- Show-Score Method

Chapter 13 What are Design patterns for Java?

- The design pattern library for Java can be found here: Java Patterns
- What is JSR-318?
- How do I take advantage of this?
- Where do I go from here?

Chapter 14 What are the web services for Java?

- What are libraries and tools for Java web services?
- What about Java EE 7?
- What is the difference between Java EE 6 and Java EE 7?
- Java EE specifications
- What about the status of the Java EE 7 and its specifications?
- How do I know which Java EE 7 specifications are available in my IDE?
- Eclipse Enterprise for Java 6
- Eclipse Enterprise for Java 7
- Eclipse EE 7 and Cloud Foundry
- Eclipse EE 7 and Tomcat
- Eclipse EE 7 and JBoss
- Eclipse EE 7 and NetBeans
- Eclipse EE 7 and Maven
- Eclipse EE 7 and JBoss

Chapter 15 What is Java for phones?
- How Java for phones will work
- Flexible domain objects
- Supported platforms for Java for phones
- Phone targets
- Phone API targets
- Testing the Java for phones platform
- Java for phones - what it could mean for Java fans
- So should Java developers be interested in Java for phones?
- What is Java development?
- What are its strengths?
- What are its weaknesses?
- What can I use with Java?
- What are its alternatives?
- Swift
- Flock
- Go
- Frameworks
- Swift Starter
- CodeProject
- Frugal
- Elixir
- AWS Lambda
- ActiveMQ
- Node.js
- Github
- Distributed.org
- Learn Java

Chapter 16 How to use Java?
- Java as an industry-wide movement
- Java FAQ for Scala newcomers
- Why are so many people switching to Scala, and what is the reason?
- Java language is too big for its good

- Exposure to more complex languages

Conclusion

- Why has Java been around so long?
- How can someone easily switch from Java to other languages?
- What do you think are the most important things to learn when coming from another language to Java?
- Introduction to Functional Programming
- Using pattern-matching to match on more data structures
- Using the WSDL for validation.
- Faces vs. Extensible Markup Language.
- Tools for the Real World.

Introduction

What is the Java language?

Java is a high-level, compiled, procedural language that's most often used as the underlying engine behind the Java virtual machine (JVM) and as the basis for the Java standard libraries.

Java is one of several programming languages that compile down to machine code and run on IBM mainframes and Intel x86 processors, as well as numerous other architectures.

Since Java is a scripting language, it is considered a dynamic programming language, meaning that it's easy to automate tasks with many steps and little or no boilerplate.

Java runs on many different platforms: It runs on the Unix, Linux, and macOS operating systems; on Microsoft Windows; and it's the language of choice for Android. You can also run Java on servers, in containerized environments, and on embedded systems, such as the Raspberry Pi.

Java is a developer-centric programming language, meaning it's designed for programmers who want to write code for a variety of systems, from the command line to the web browser. It's also a language that's quite friendly to extensibility and customization.

Java has traditionally been thought of as a "server-side" language, which means it's ideal for writing server applications. But over time, its potential for creating desktop applications grew. That's partly due to the presence of its Cordova software development kit, which allows Java code to be used as the basis for mobile apps.

Why is Java such a popular programming language?

Java is a mature, popular language that has a very friendly developer community. Over the past few years, a large community of Java software developers has sprung up around the platforms it supports.

While many traditional programming languages have enjoyed a strong following due to their productivity, which allows for rapid development, Java has evolved into a "server-side" language that's ideal for making robust web applications, web services, and enterprise software. Because of its popularity and excellent documentation, Java is quite popular in the marketplace, with many businesses that use Java systems choosing to keep it around in part because of its long-term commitment to its developer community.
Java also is an exceptional language for writing low-level programming for parallel computing. Because it compiles down to machine code, it's a popular choice for supercomputers, where performance is paramount.

What is the Java programming language?

Java is a popular object-oriented programming (OOP) language that's designed to be "interactive." That means that instead of the developers writing applications directly, in Java, all of the software elements are encapsulated in objects. Thus, all of the coding is done in Java code and object code, and the interface (i.e. application) is coded in a developer-friendly language, such as Java.

The main characteristic of object-oriented programming is that an object's state—whether it's "downloaded," "updated," or "disposed of"—is visible to other objects. Therefore, a single object in an object-oriented program doesn't need to know the details of how to do anything. When a developer has to code that out, that's what a "method call" is, and that's what Java is about.

A "method call" is a very powerful feature of object-oriented programming, because it allows a developer to instantiate and destroy an object, and thus to change the object's state. That allows a developer to build up a business logic with values for the object's state—for example, a file name in a web service—and then to make a "method call" to that file name to make that value readable to the other objects in the system.

In most programming languages, a method call uses the runtime library to create and return values from the method. But because the main function of Java is to be used on the operating system, which doesn't contain a runtime library, a method call can be made directly from within the Java code.

This is a very powerful feature for the developer since the code to implement the method call, and the methods that can be made to return from that method can be created in Java code itself.

Another main feature of Java is that it allows developers to make copies of objects, instead of creating objects on the fly. So if an object (in Java, a class) has a lot of children, for example, this makes the developer's life easier because she doesn't need to worry about keeping up with objects and their changes—she only has to worry about the one "owner" of the object (in the case of the class).

This approach also allows the developer to encapsulate common behavior into objects that can be shared across the system. The benefit here is that the class is the main value store of the system, and each piece of software needs to be able to access this value store to manipulate the underlying functionality. This shared data is referred to as an "interface" for the class, and you can get that with a method call.

The biggest thing to note about Java is that, for the most part, it's designed for running on the OS and CPU and nothing more—it doesn't have any dependency on other operating systems or operating systems. This means that it's much easier to update Java when something new is available. It also means that it's easier to integrate new features into the language.