
ISBAT UNIVERSITY
BLENDED LEARNING PLATFORM

# JAVA PROGRAMMING ESSENTIALS

with NetBeans

## Unit One

## Introduction To JAVA

by Mr. Bulega Herbert Moses

Professional Certificate in Software Engineering [PCSE]

# UNIT ONE CONTENT OUTLINE



- Introduction to **Java Programming**
- Explain the **structured programming paradigm**
- Explain the **features of Java as an OOP** language
- Describe **Java platform and its components**
- List the **different editions of Java**
- What is **JDK, JRE and JVM**
- **Downloading and installing Java Development Kit (JDK)**
- Identify the benefits of the NetBeans IDE
- Netbeans Download and Installation

# UNIT ONE CONTENT OBJECTIVES



- To equip students with the fundamentals of Java programming.
- To enable students, understand Java features as an OOP language.
- To expose students to group participation to be able to use the Java SDK environment to create, debug and run simple Java programs.
- To expose students to the different Java Editions.
- To teach students Application Development in Java
- To teach students how to downloading and installing Java Development Kit
- To teach students the JDK, JRE and JVM.
- To teach students the benefits of Netbeans as an IDE and how to Install it

# INTRODUCTION TO JAVA PROGRAMMING



**Software** is developed using **programming languages**. There are many programming languages so why **Java?** The answer is that Java allows users to develop and deploy **applications on the Internet for servers**, **desktop computers**, and **hand-held devices**. **Java** is a **high-level** secure **object-oriented programming** language and a platform. **Java** was made by a team led by **James Gosling** at **Sun MicroSystems** now owned by **Oracle**. Before **Java**, its name was **Oak**. Any hardware or software environment where a program runs, is known as a **platform**. Since Java has a runtime environment (**JRE**) and an **API**, it is called a platform. As a **programming language**, we can write programs in **java** using **english** based commands which can be executed on different **types of devices**.

## STRUCTURED PROGRAMMING PARADIGM

- In a structured programming paradigm, application development is decomposed into a hierarchy of subprograms.
- The subprograms are referred to as **procedures**, **functions**, or **modules** in different structured programming languages.
- Each subprogram is created to perform a specific task.
- Examples of structured programming languages are **C**, **Pascal**, **Cobol** etc

## FEATURES OF JAVA AS AN OBJECT-ORIENTED PROGRAMMING LANGUAGE

●**JAVA IS SIMPLE AND FAMILIAR: Java** is simple since it simplifies the programmer's job by avoiding some features in **C** and **C++. Memory management** in java is **automatic** and done by the **JVM** so there are no chances of **memory leakages.** There are no **pointers** in java. Java is familiar since its modeled on the **C** and **C++** languages. Java uses many **C** and **C++** features so java codes look like C++ codes. We can say java is a simplified version of C++.

# FEATURES OF JAVA AS AN OBJECT-ORIENTED PROGRAMMING LANGUAGE



●**JAVA IS PORTABLE:** Portability allows programmers to write the same **source code** for different machines (operating systems). Java provides portability in two ways: **Source code portability** and **Byte code portability**. In Java, we can write **source code** and **byte code** for different OS and also get the output. Whereas in **C** and **C++** there is only source code portability.

●**JAVA IS ARCHITECTURALLY NEUTRAL:** The behavior of **java programs** doesn't change when we move from one system to another which means it provides the same output in every machine since **memory layout decisions** are **not made at compile time**, are **made at run time** by the **JVM**. But in **C** and **C++** programs, the behavior changes when we move our program from one system to another.

●**JAVA IS SECURED: Java** is a secured programming language since **it has no pointers**. A pointer in java is called a **restricted pointer** means there is no **Pointer Arithmetic**. Java programs are executed in a secured environment called **JVM**. **JVM** will provide security to the java programs.

●**JAVA IS ROBUST:** Robust means strong. **Java** is a **strong Type Checking Language** having a **strict Type Checking** during both compilation time and execution time which allows us to develop both error-free applications and programs.

●**JAVA IS DISTRIBUTED:** Java provides a set of **APIs** which allows users to develop **distributed applications**. It means java language is used for developing distributed applications whose resources are shared by more than one client. So, java is a distributed programming language.

●**JAVA IS MULTITHREADED:** A **process** is divided into several small parts known as **threads** or **lightweight processes**. Sending multiple threads to the processor for processing is known as **Multi-Threading**. **Multi-threading** means handling multiple tasks simultaneously. E.g., we can listen to music while scrolling a page at the same time we can download an application from the internet on a computer. Java supports multithreaded programming.

# FEATURES OF JAVA AS AN OBJECT-ORIENTED PROGRAMMING LANGUAGE



●**JAVA IS STRONGLY-TYPED:** The variable types used must be pre-defined and **conversions** to other objects is relatively strict.

●**JAVA IS DYNAMIC:** Linking of a program is of two categories; **Static linking** and **Dynamic linking. Static linking:** linking of all executable blocks is done before executing programs. If there is a small change in the **executable block**, you will need to compile the whole program. So here wastage of memory and the efficiency of the whole program decreases. Example: C & C++. **Dynamic linking**: Loading and linking of all executable blocks done at the time of program execution is called **dynamic linking**. It increases the efficiency of the program. Any small change in the executable block, you will not need to compile the whole program. So there is no waste of memory in java.

●**JAVA IS COMPILED AND INTERPRETED:** We now know a programming language is either **compiled** or **interpreted**. But java combines both approaches. That's why java is called a **two-stage system**. First, java compiler **JAVAC** translates source code into an intermediate code known as **byte code**. But codes are not machine instructions. So in the second stage, this byte code is interpreted by the **java interpreter** (**JVM**) in java. As a result, machine instructions are generated which are directly executed by the machine. Hence java is both interpreted and compiled language.

●**JAVA IS OBJECT-ORIENTED:** Except for the **primitive data types**, all elements in Java are **objects**. Object-oriented is not a programming language, it is a **programming method** or **concept**, or **principle** which defines a set of rules and regulations for organizing data and instructions. The concepts provided by oops are; **Encapsulation**, **Abstraction**, **Polymorphism** and **Inheritance.** A programming language that supports these four features is known as an object-oriented programming language. Java supports these four features so java is object-oriented.

●**JAVA IS PLATFORM INDEPENDENT:** Many programming languages are compatible with only **one platform**. **Java** was specifically designed so that it would run on any computer, whether it was running Windows, Linux, Mac, Unix, or any of the other operating systems.

# FEATURES OF JAVA AS AN OBJECT-ORIENTED PROGRAMMING LANGUAGE



●**AUTOMATIC MEMORY MANAGEMENT: Java** manages **memory allocation** and **de-allocation** for making new objects. The program does not have direct access to memory. The so-called **garbage collector** automatically deletes objects to which no active pointer exists.

## JAVA TERMINOLOGIES USED IN A PROGRAMMING LANGUAGE

1. **Source code:** The developer-written program is called the source code. It is written according to the programming language syntax.
2. **Compiled code:** a compiler-generated program that is converted from source code is called compiled code.
3. **Compiler:** it is a translation program that converts the source into machine language at once.
4. **Interpreter:** it is also a translation program that converts source code into machine language but line by line.
5. **Executable code:** OS understandable executable programs (.exe files)
6. **Compilation:** it's the process of translating source code into compiled code.
7. **Execution:** its the process of running the compiled code to get the output.
8. **Open-source software** is a software whose source code is also released along with software so that you can read or change the source code as per your needs and can re-distribute the software as well.

# DIFFERENCES BETWEEN JAVA, C AND C++ PROGRAMMING LANGUAGES



| JAVA | C | C++ |
|---|---|---|
| Java is a pure object-oriented language. | C is procedural language. | C++ is an object-oriented language. |
| We can create our own package in Java. | Cannot create our own package in C. | Cannot create our own package in C++. |
| Internet programming like Frame and Applet is used in Java. | Frame and Applets are not used in C. | Frame and Applets are not used in C++. |
| Java uses a Compiler and Interpreter. | C uses only the Compiler. | C++ also uses only the Compiler. |
| Multiple Inheritance is not used in Java. Instead of multiple inheritance Java uses Interfaces. | There is no Inheritance in C. | Multiple Inheritance is used in C++. |
| Java is a platform-independent language. | C is platform-dependent. | C++ is also platform-dependent. |
| Java doesn't support any header files. | In C, we use stdio.h header file. | In C++, we use iostream.h and conio.h header files. |
| Java does not support Pointers. | Pointers are used in C. | Pointers are used in C++. |
| Exception Handling is supported by Java. | There is no Exception Handling in C. | C++ supports Exception Handling. |
| Java is used to develop Android Applications. | C is used in Embedded Programming. | C++ is used in Game Development and System Programming. |

# JAVA PLATFORM AND ITS COMPONENTS



**Java** is similarly a **platform**. When you run **a java program** or **application,** java creates **a runtime environment** where your java program or application runs. **Operating systems** such as **Ms. Windows**, **Linux**, **Solaris**, **Mac OS** etc are called **platforms.** These platforms provide **an environment** where you can run different categories of **software** or **applications,** similarly **java** creates an environment **at runtime** where you can only run **java programs** or **applications**. **Java** platform is a **software-only** platform that runs on top of a **hardware-based platform** (**OS**) and they, Java platforms, have a **Java Virtual Machine** (**JVM**) and **application programming interface** (**API**).

## TYPES OF JAVA PLATFORMS

**Java** platforms are basically java software that is used in development and execution of java programs or applications. You can download and use these software. There are four types of java platforms:

1. **Java Platform, Standard Edition (Java SE).**
2. **Java Platform, Enterprise Edition (Java EE)**
3. **Java Platform, Micro Edition (Java ME).**

| Platform | Description |
|---|---|
| Java SE (Standard Edition) | For general purpose use on desktop computers and servers. Some early versions were called J2SE (Java 2 Platform, Standard Edition). |
| Java EE (Enterprise Edition) | For developing distributed applications that run on an intranet or the Internet. Some early versions were called J2EE (Java 2 Platform, Enterprise Edition). |
| Java ME (Micro Edition) | For devices with limited resources such as mobile devices, TV set-top boxes, printers, and smart cards. |

# JAVA JDK, JRE AND JVM

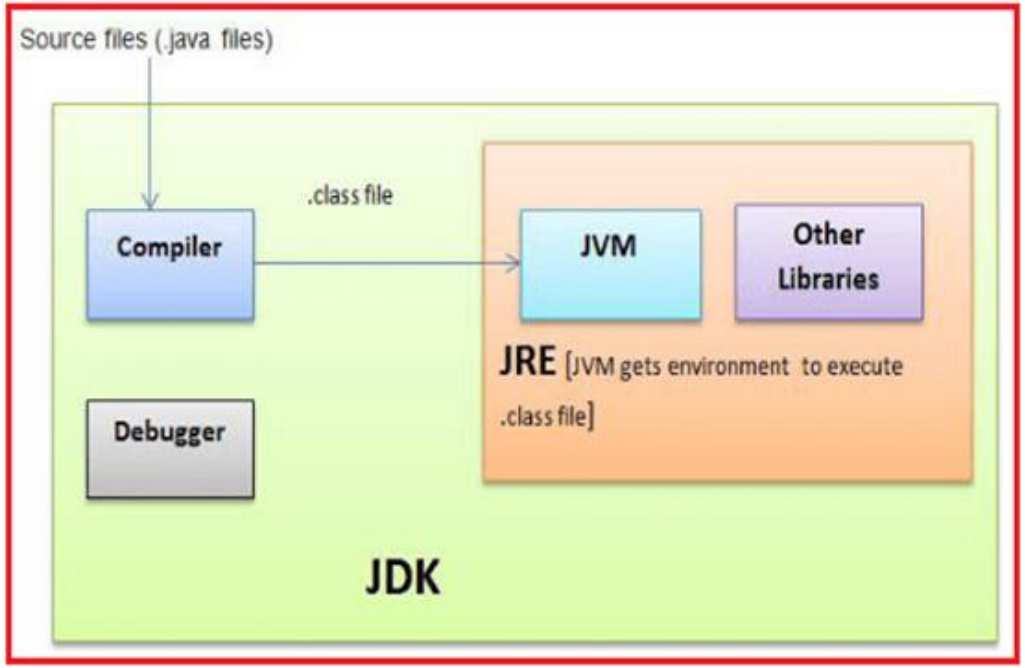


**JDK** (**Java Development Kit**) allows developers to create Java applications and applets that can be executed and run by the **JVM** and **JRE**. **JDK** is a software package you download in order to create Java-based applications. A **JDK** is an implementation of the Java platform specification, including the **compiler** and **class libraries**. **The important features of JDK are:**

- It enables programmers to handle multiple extensions in a single catch block.
- JDK includes all features that JRE has.
- It contains development tools such as a compiler, debugger, etc.
- JDK provides the environment to develop and execute Java source code.
- It can be installed on Windows, Unix, and Mac operating systems.

## JVM (JAVA VIRTUAL MACHINE)

**JVM** (**Java Virtual Machine**) is called a **virtual machine** because it does not physically exist. It is a specification that provides a runtime environment in which **Java bytecode** is executed. It can similarly run programs which are written in other languages and compiled to Java bytecode. It converts **Java bytecode** into machine language. **JVM** is part of the **Java Run Environment** (JRE). It can't be separately downloaded and installed. To install JVM, you need to install JRE. **The important features of JVM are:**



# JAVA JDK, JRE AND JVM



- It enables you to run applications in a cloud environment or on your device.
- **Java Virtual Machine** converts **byte code** to **machine-specific code**.
- It provides basic java functions like memory management, security, garbage collection, and more.
- JVM runs the program by using libraries and files given by the Java Runtime Environment.
- JDK and JRE both contain Java Virtual Machine.
- It can execute the java program line by line hence it is also called an interpreter.
- JVM is easily customizable for example, you can allocate minimum and maximum memory to it.
- It is independent of hardware and the operating system. So, you can write a java program once and run it anywhere.

## JRE (JAVA RUNTIME ENVIRONMENT)

**JRE (Java Runtime Environment)** is also written as Java **RTE**. **JRE** is a piece of software that is designed to run other software. It is used to provide a runtime environment. It is the implementation of **JVM**. It physically exists. It has a set of libraries and other files that JVM uses at runtime. All **JDK versions** come bundled with **Java Runtime Environment**, so you do not need to download and install the **JRE** separately on your PC. The implementation of JVM is also actively released by other companies besides Sun Microsystems. **The important features of JRE are:**

- Java Runtime Environment is a set of tools used by the JVM to run.
- JRE contains deployment technology, including Java Web Start and Java Plug-in.
- Developers can easily run a source code in JRE, but they cannot write and compile a Java program.
- It has integration libraries like **Java Database Connectivity (JDBC), Remote Method Invocation (RMI), Java Naming and Directory Interface** (JNDI) etc.
- JRE has JVM and Java Hot Spot virtual machine clients.

# DIFFERENCES BETWEEN THE JDK, JRE AND JVM IN JAVA

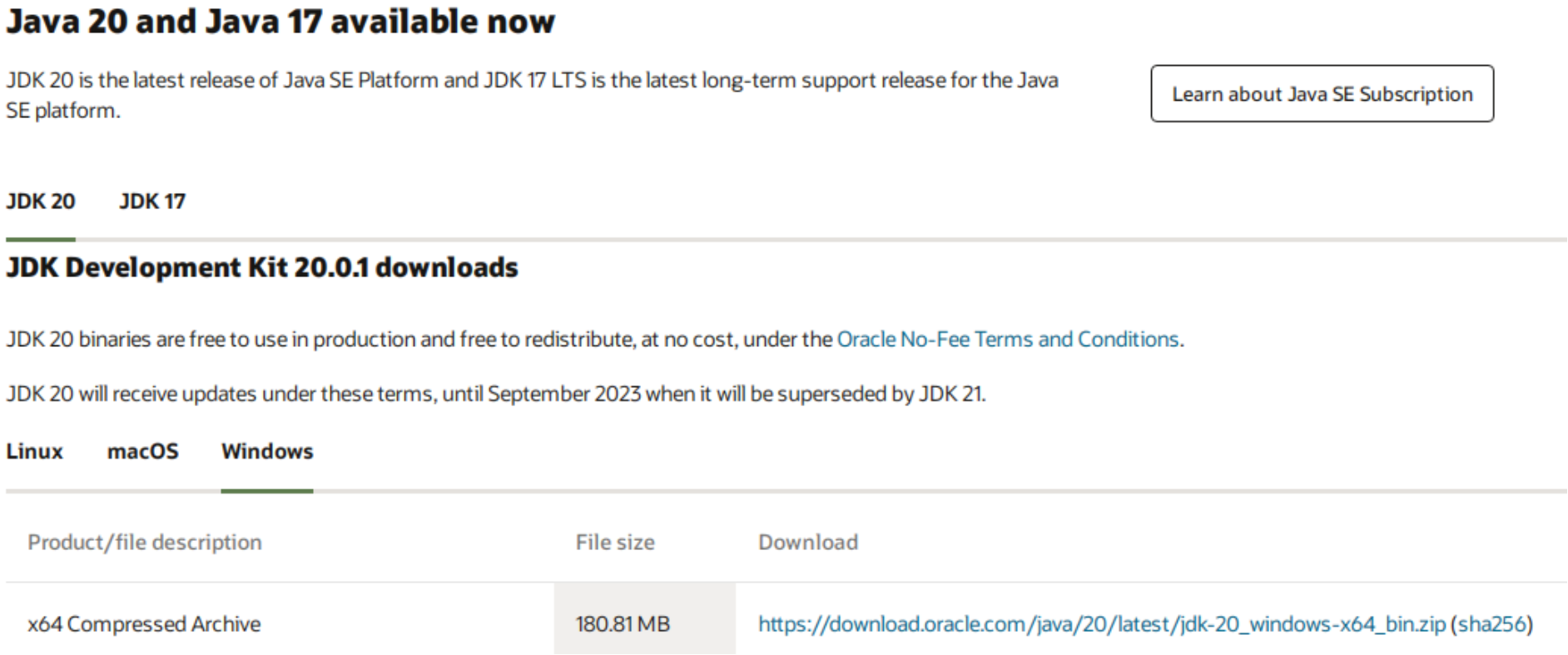
| JDK | JRE | JVM |
|---|---|---|
| The full form of JDK is the Java Development Kit. | The full form of JRE is the Java Runtime Environment. | The full form of JVM is Java Virtual Machine. |
| JDK is platform-independent. | JRE is also platform-independent. | JVM is also platform-independent. |
| It is the superset of JRE. | It is the superset of JVM. | It is the subset of JRE. |
| JDK comes with the installer. | JRE only contains the environment to run the source code. | JVM is bundled with JDK and JRE software. |
| It contains tools for developing, debugging, and monitoring java code. | It contains class libraries and other supporting files that JVM requires to execute the program. | Software development tools are not included in JVM. |

We shall use the **Java SE** in our Java programming language. There are many versions of **Java SE**. Sun releases each version with a JDK **Java Development Toolkit**. For **Java SE16**, the **Java Development Toolkit** is called **JDK 16** (also called **Java 16** or **JDK 16**). **JDK** consists of a collection of separate programs, each invoked from a command line, for developing and testing Java programs. Besides a **JDK**, you can also use a **Java development tool** for example **NetBeans** or **Eclipse** software that provide an **integrated development environment** (**IDE**) for rapid Java programs development. **Editing**, **compiling**, **building**, **debugging**, and online help **are all integrated** in one **graphical user interface**. Just enter a source code in one window or open an existing file, then click a button, menu item, or function key to compile and run the program.

# ENVIRONMENTAL SETUP FOR JAVA APPLICATION DEVELOPMENT

## DOWNLOAD AND INSTALL JDK

**Step1:** Go to link **https://www.oracle.com/java/technologies/javase-downloads.html** and click on download JDK (for JAVA latest version) as shown in the below image.



**Java 20 and Java 17 available now**

JDK 20 is the latest release of Java SE Platform and JDK 17 LTS is the latest long-term support release for the Java SE platform.

Learn about Java SE Subscription

JDK 20 JDK 17

**JDK Development Kit 20.0.1 downloads**

JDK 20 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

JDK 20 will receive updates under these terms, until September 2023 when it will be superseded by JDK 21.

Linux macOS Windows

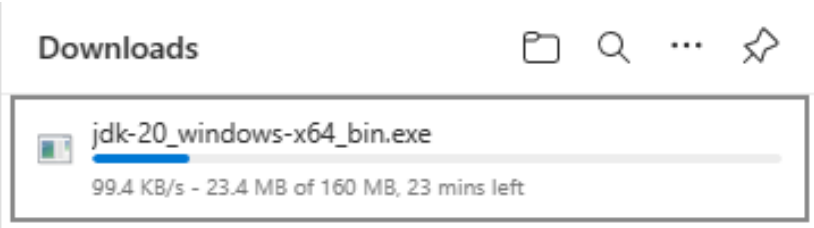
| Product/file description | File size | Download |
|---|---|---|
| x64 Compressed Archive | 180.81 MB | https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.zip (sha256) |

# ENVIRONMENTAL SETUP FOR JAVA APPLICATION DEVELOPMENT

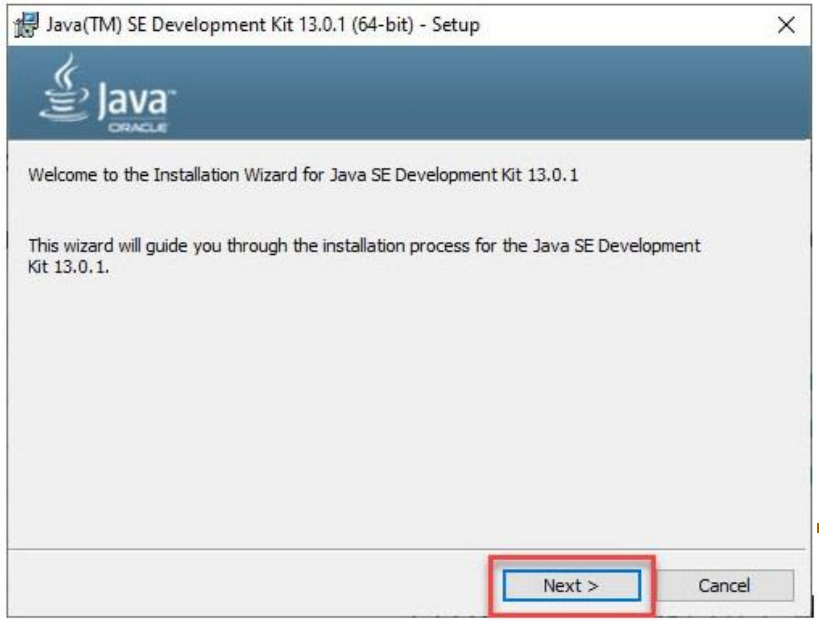
## DOWNLOAD AND INSTALL JDK

**Step2:** Next,

1. Download the latest Java JDK for your version (32 or 64 bit) of java for Windows.




**Step3:** Once the download is complete, run the exe for install JDK. Click Next.




# ENVIRONMENTAL SETUP FOR JAVA APPLICATION DEVELOPMENT

**Step5:** Once the installation is complete click Close.

The PATH variable gives the location of executables like **javac**, **java**, etc. It is possible to run a program without specifying the PATH but you need to give full path of executable like **C:\Program Files\Java\jdk-13.0.1\bin\javac A.java** instead of simple **javac A.java**. The CLASSPATH variable gives the location of the Library Files.

## Following are the steps to set Environment Variables in Java:

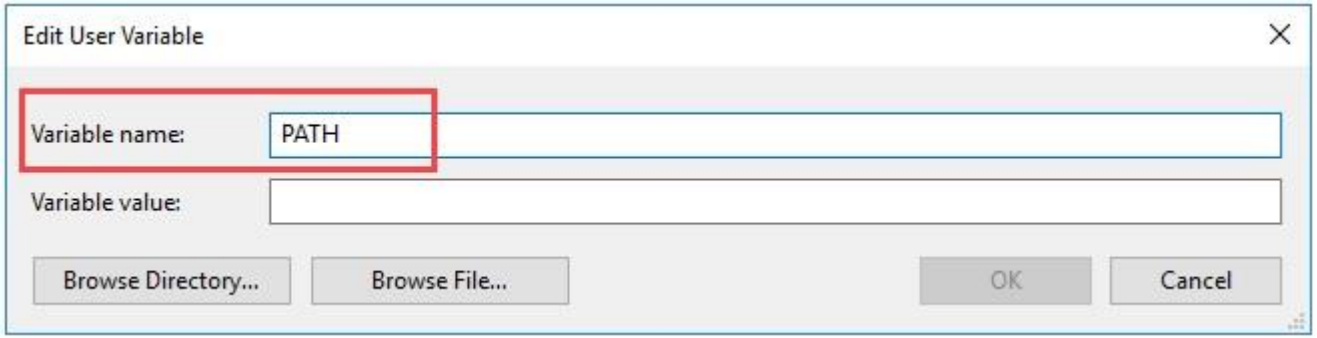
**Step1:** Right Click **on My Computer** and select the **properties**.

**Step2:** Click on Advanced system settings.

**Step3:** Click on Environment Variables.

**Step4:** Click on the new Button of User variables.

**Step5:** Type PATH in the Variable name.

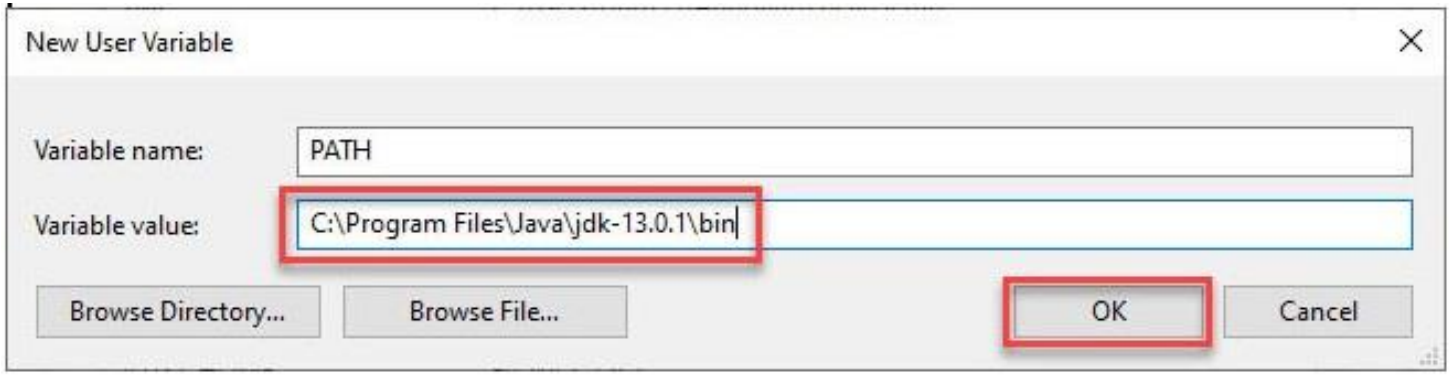


# ENVIRONMENTAL SETUP FOR JAVA APPLICATION DEVELOPMENT



**Step6:** Copy the path of the bin folder which is installed in the JDK folder.

**Step7:** Paste Path of bin folder in Variable value and click on OK Button.




**Step9:** Click on the OK button

**Step10:** Go to the command prompt and type javac commands.
If you see a screen like below, Java is installed.

# JAVA PROGRAMMING APPLICATION TYPES

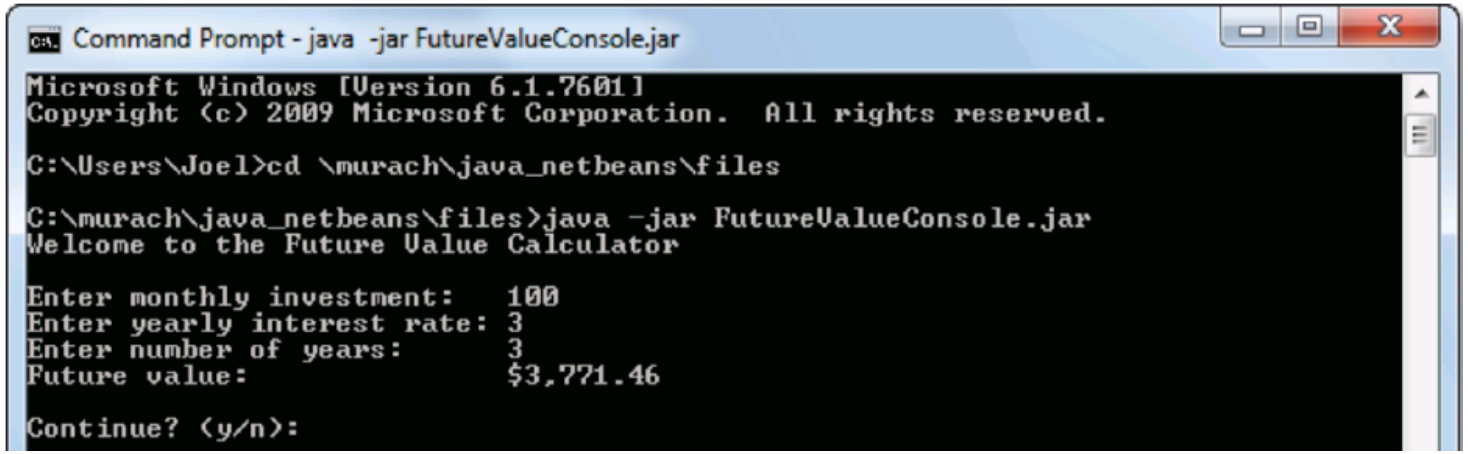


You can use the Java programming language to create any type or category of application (also known as an app or a program). You can develop applications from desktop applications to web applications and mobile apps.

## 1.DESKTOP APPLICATIONS

You can create two types or categories of desktop applications with the Java Language. Java desktop applications run directly on your desktop computer. **1.CONSOLE APPLICATION** and **2.GRAPHICAL USER INTERFACE (GUI) APPLICATION**.

1.A **console application** runs in the **console** or **command prompt** that's available in your **operating system**. The console provides a simple way to get input from a user and to display output to the user. Below, for example, a user has entered 3 digits in a console application, and the application has performed a calculation and displayed the result. When learning Java, it's common to work with console applications until you have a solid understanding of the Java language.



# TYPES OF JAVA PROGRAMMING LANGUAGE APPLICATIONS

2.A **java graphical user interface (GUI) desktop application** development needs some significant Java coding skills. Once you have a solid understanding of the Java programming language, then you can create **a desktop application** that uses a **graphical user interface** (GUI). Below is an example of a GUI application that performs some tasks. However, GUI applications are more user-friendly and intuitive.

**NOTE**: A **console application** uses the consol to interact with the user. A **(GUI) graphical user interface** application uses a graphical user interface to interact with the user.

# TYPES OF JAVA PROGRAMMING LANGUAGE APPLICATIONS

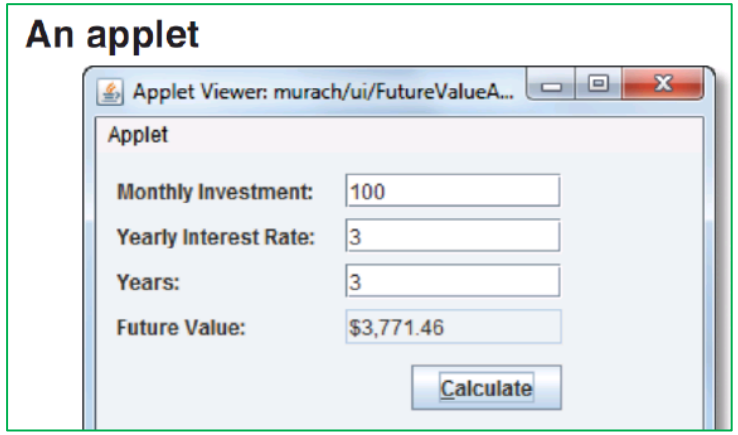
## 2.WEB APPLICATIONS

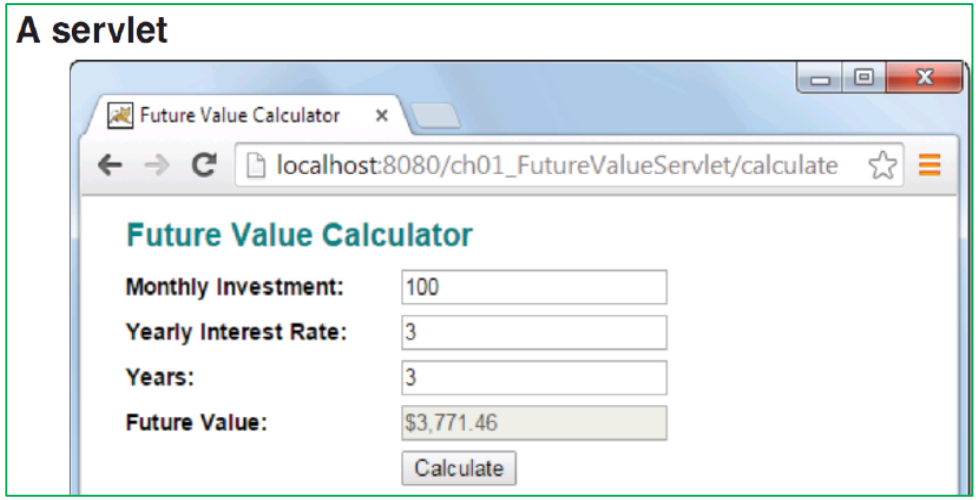
You can create two types of **web applications** with the Java programming Language. Java web applications run directly on the server.
**1**.**APPLET WEB APPLICATION** and **2**.**SERVLET WEB APPLICATION**.

**1**.An **applet web application** can be made with **Java** and stored in an HTML page and runs inside a **Java-enabled browser**. In the old days of **Java**, which were also the old days of the **Internet**, one exciting features of **Java** was to create a special type of web-based application known as an **applet**. **An applet** works like the GUI application. But, unlike that GUI application, an applet is stored in an HTML page and run inside a Java-enabled browser. But with tightening security restrictions, fewer and fewer browsers support applets, even if you install the plug-in that was designed to allow applets to run in browsers. Due to these security restrictions, applets are effectively obsolete. A better way to provide access to enterprise databases is to use Java EE to create web applications that run on the server. These applications are often based on servlets.

**2**.A **servlet** is a special type of Java web application that runs on the server and can be called by a client program, which is usually a **web browser**. Servlets work like the applet. But, **servlet** runs on a **server computer**, not the **client computer**. To start, the **web browser** on the client computer sends a request to the servlet that's running on the server computer. When the servlet receives this request, it performs the calculation and returns the result to the browser, typically in the form of an HTML page. it's common for servlets to work with a database. E.g. suppose a browser requests or asks a servlet to displays all unprocessed invoices that are stored in a database. Then, when the servlet is executed, it reads data from the database, formats that data within an HTML page, and returns the HTML page to the browser. When you create a servlet-based application, all the processing takes place on the server and only HTML, CSS, and JavaScript is returned

to the browser. That means that anyone with an Internet or intranet connection, a web browser, and adequate security clearance can access and run a servlet-based application. To make it easy to store the results of a servlet within an HTML page, the Java EE specification provides for **JavaServer Pages** (JSPs). As a result, it's common to use JSPs with servlets.




## REMEMBER

- An **applet** is a type of Java web application that runs **inside a web browser**. In the past, it was possible to run applets in most web browsers. Today, fewer and fewer web browsers support applets. As a result, they are effectively obsolete.
- A **servlet** is a type of Java web application that runs **on a web server**. A servlet accepts requests from clients and returns responses to them. Typically, the clients are web browsers.

# TYPES OF JAVA PROGRAMMING LANGUAGE APPLICATIONS

## 3.MOBILE APPLICATIONS

You can also use **Java** to develop **mobile apps**, which are applications that run on **mobile devices** e.g. **smartphones**, **tablets** etc. In particular, **Java** is typically used to write code for apps that run on **Android devices**. An **app** works like traditional application. But, the **user interface** is modified so that it's appropriate for a mobile device and can also work with a touch-screen device that has a small screen with no keyboard. As a result, the user can use the keypad that's displayed onscreen to enter numbers and can press the **Done button** on this keypad to perform the calculation. The **Android operating system** has its own **virtual machine** that supports a subset of **Java**, including most features of **Java**. As a result, when you use **Java** to develop **Android apps**, you cannot use all the **Java features**, mainly the newest. That's because the **Android virtual machine** is not a **Java virtual machine**. In other words, the Android virtual machine can't run compiled Java code, and a Java virtual machine can't run compiled Android code. Still, you can use most features of Java to write code for Android apps, and it's easy enough to compile that code so the Android virtual machine can run it.

# INTRODUCTION TO IDEs FOR JAVA SOFTWARE DEVELOPMENT

## WHAT IS AN (IDE) INTEGRATED DEVELOPMENT ENVIRONMENT

To develop **Java applications**, you typically use an **Integrated Development Environment (IDE).** Although you can use **a simple text editor** with **command line tools**, an **IDE** provides features that can make developing Java applications much easier. The following **IDEs** stated here are either free or have a free edition. That makes them so attractive to **students** and **programmers** who are learning on their own. These **IDEs** also run on all modern **operating systems**. The first two **IDEs NetBeans** and **Eclipse**, are the most popular **Java IDEs**.

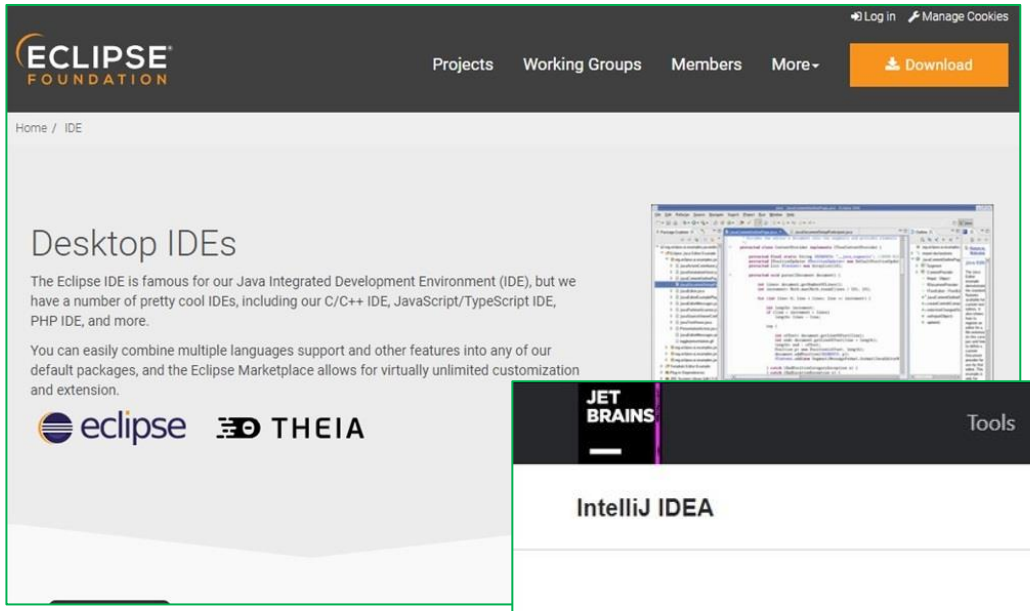
**Java IDE** (**Integrated Development Environment**) is a software application that allows users to easily **write** and **debug** Java programs. Most **IDEs** have features i.e. **syntax highlighting** and **code completion** that helps users to code more easily. Generally, **Java IDEs** have a **code editor**, **a compiler**, **a debugger**, and **an interpreter** that developers access via a **single graphical user** interface. **Java IDEs** also provide **language-specific elements** i.e. **Maven**, **Ant building tools**, **Junit** and **TestNG for testing**. **Java IDE** offers massive support for the application development process. Through using **IDEs**, **software developers** can save time and effort and set up a standard development process. **IntelliJ IDEA**, **NetBeans**, **Eclipse** and many other **IDEs** are popular in the Java **IDEs** that can be used according to our requirements. In this topic, we will discuss the best Java IDE's that are used by the users.

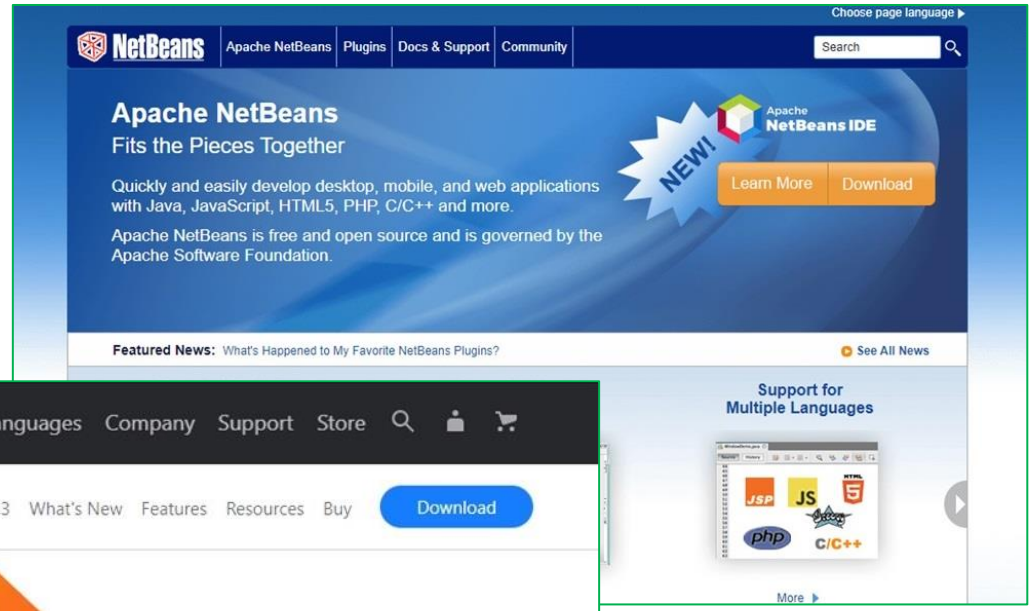
## DO YOU NEED AN IDE TO WRITE JAVA CODE?

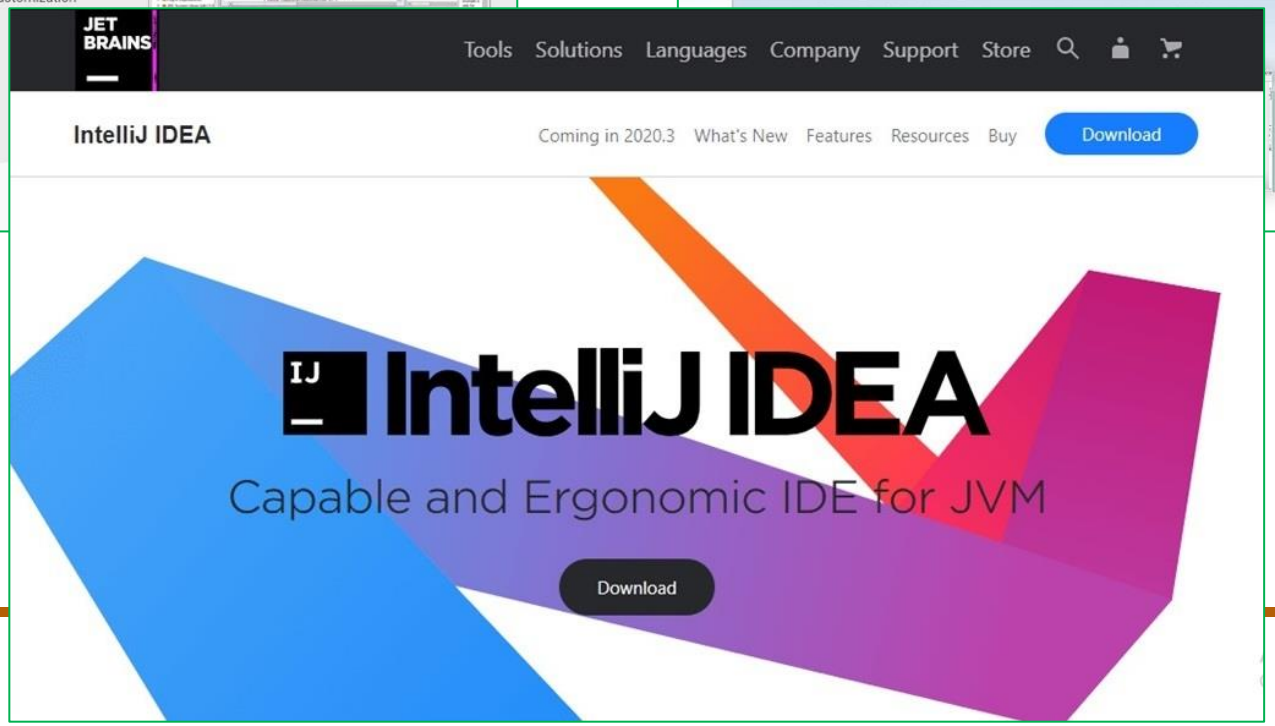
NO, you don't need to write **Java code** in an **IDE**. You can use a **text editor**, and then compile the code with **javac.** You could be writing code for a **nuclear power station**, or **company** with Windows **Notepad**, and **javac** (a **Java compiler command**). There are **Java developers** who **write code** in **Vim** (the Linux **text editor**), and run it all in the **Terminal** and they still write **excellent code**. But If you're just starting to learn **Java**, use a text editor first. It's better for your learning to write code using a plain text editor.

# INTRODUCTION TO IDES FOR JAVA SOFTWARE DEVELOPMENT

The following are some of the best Java IDEs that are mostly used today

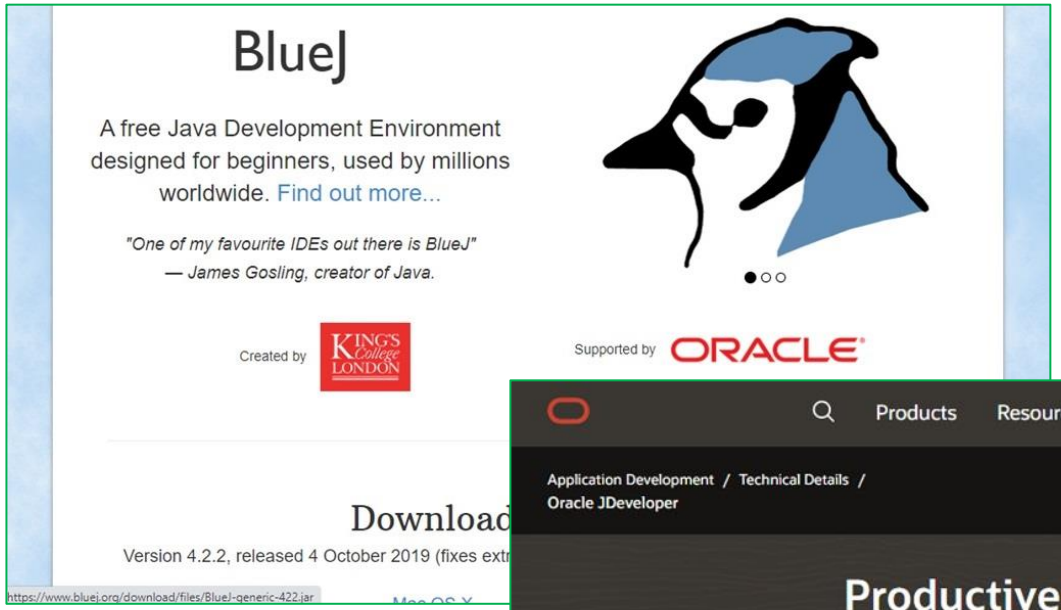


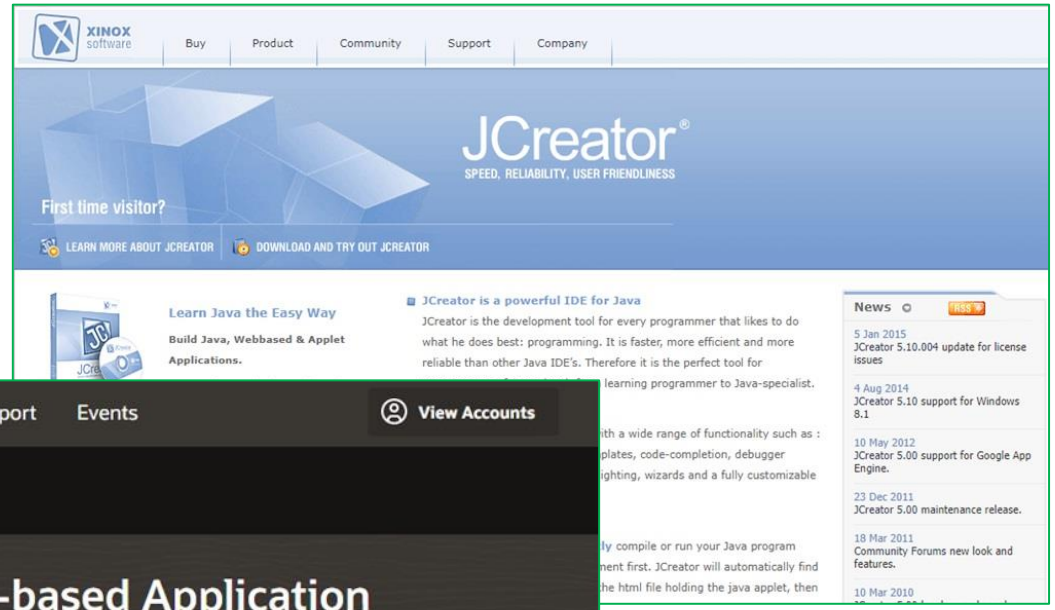



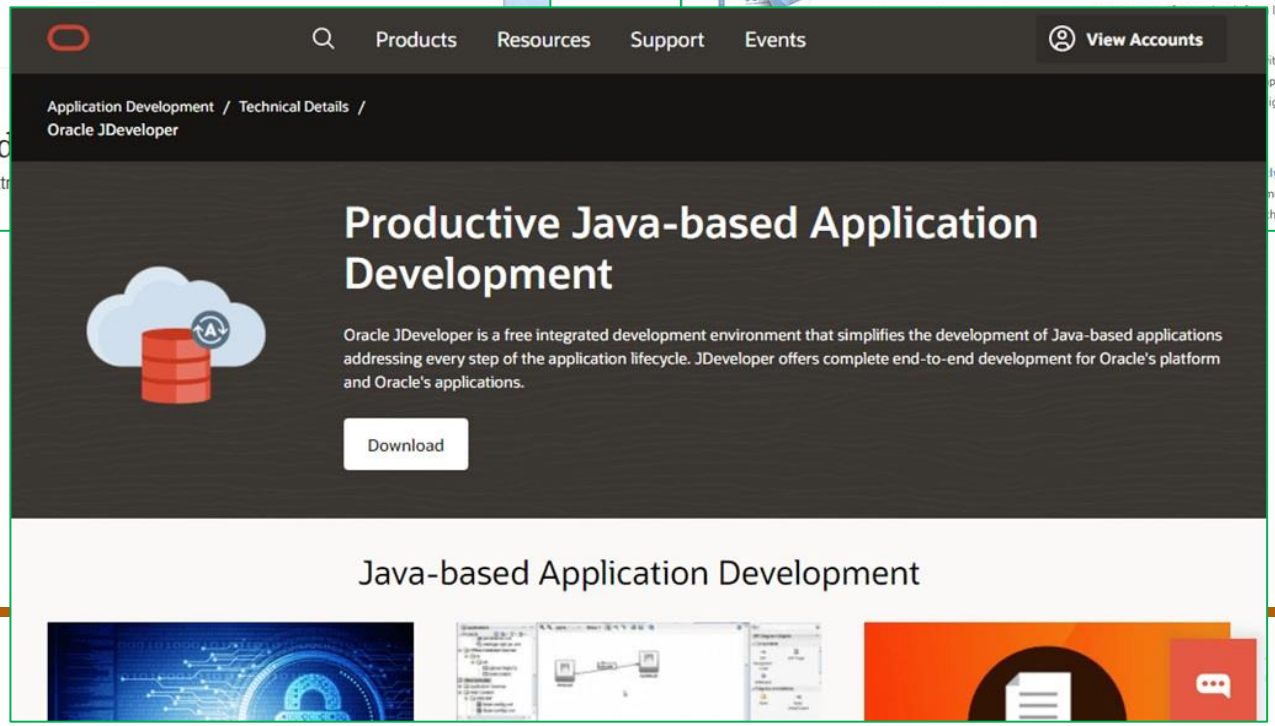



# INTRODUCTION TO IDES FOR JAVA SOFTWARE DEVELOPMENT



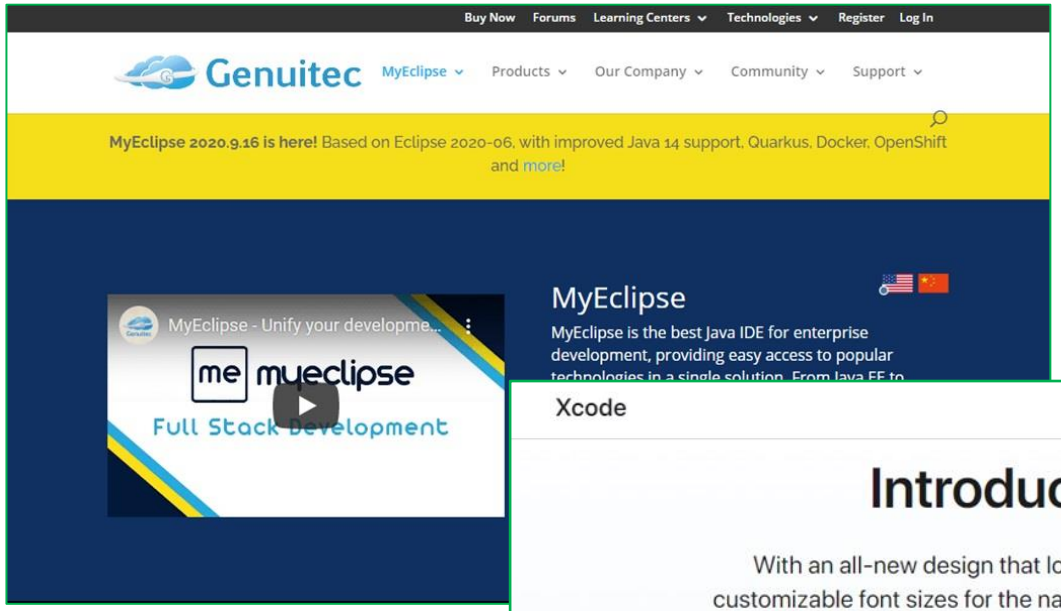


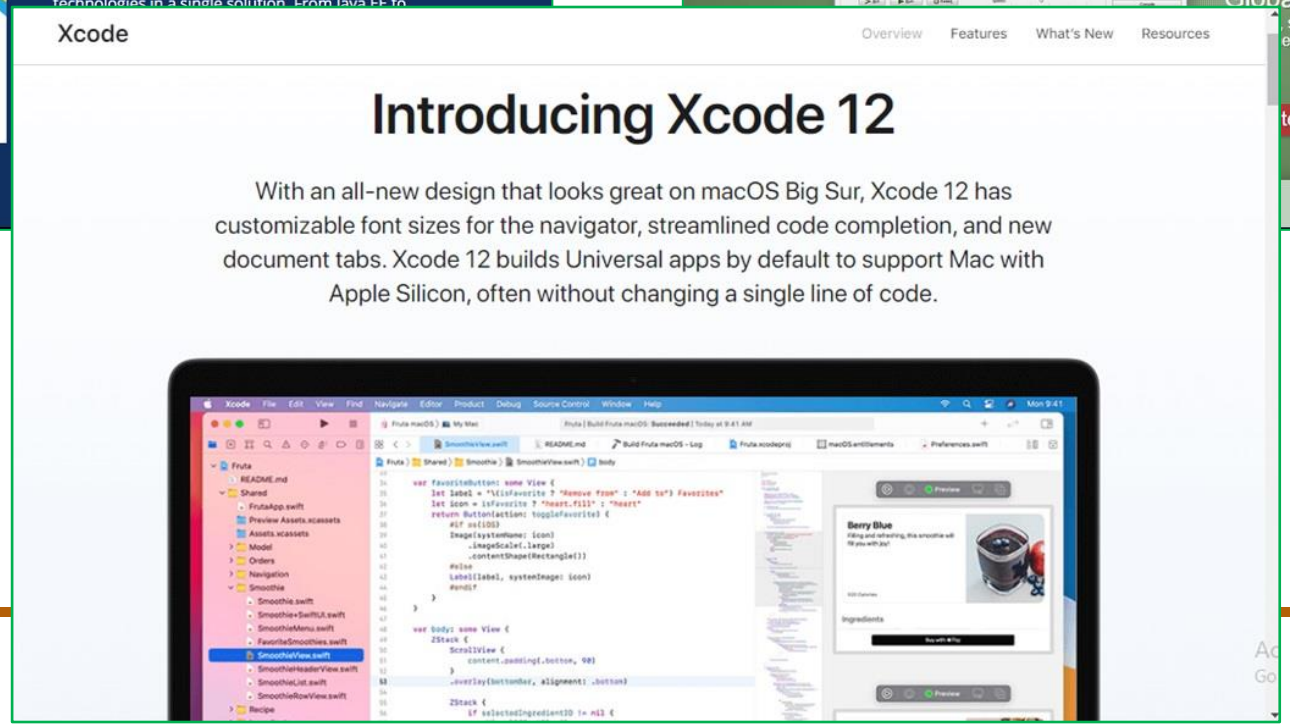



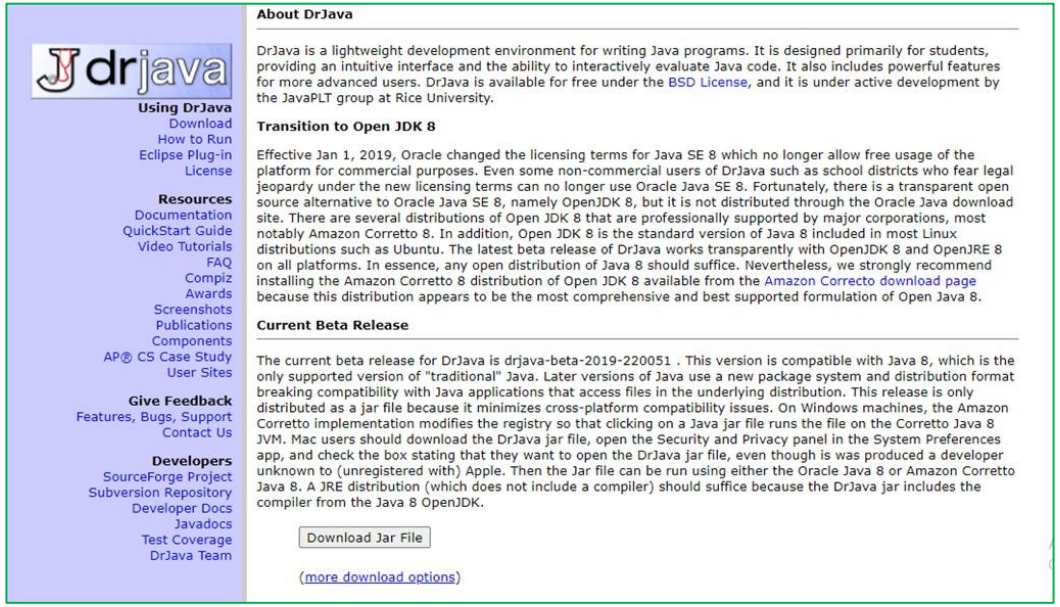
# INTRODUCTION TO IDES FOR JAVA SOFTWARE DEVELOPMENT

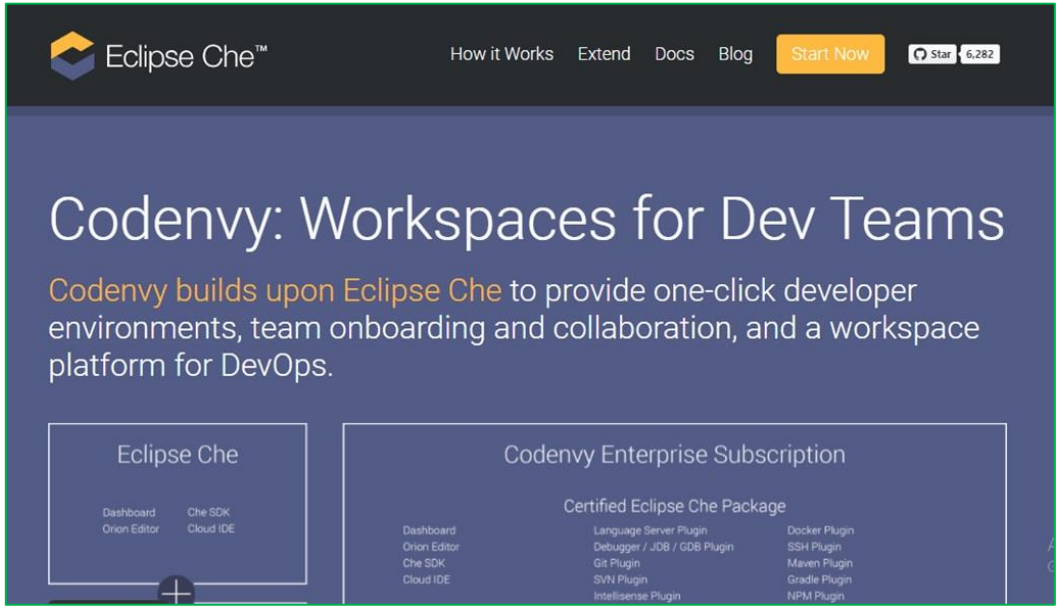











## CHOOSING AN (IDE) INTEGRATED DEVELOPMENT ENVIRONMENT

**What makes a great Java IDE?** A developer's **IDE** can be a very personal choice. (I said earlier that there are many developers who prefer to write code using Vim, the Linux text editor!). But here's what I expect any **IDE** to be:

- **FAST.** It has to be fast. Every second you spend waiting for your **IDE** to process a file, is a second that you might lose concentration.
- **STABLE.** It has to be able to handle multiple editor windows, terminals and build processes simultaneously. And it must handle all of that, without freezing or crashing.

# INTRODUCTION TO IDES FOR JAVA SOFTWARE DEVELOPMENT

- **INTELLIGENT.** A good Java **IDE** knows how to take care of **boring**, **repetitive tasks**, but it also knows to get out of your way when you just need to write code. You don't want to battle with wizards and dialog boxes all the time, just give us some **sensible default options**.
- **KNOWLEDGEABLE.** The **IDE** should know what **good code looks like**, and help me avoid **code smells** – those pieces of code which aren't great. It should also help me to optimize, by suggesting more succinct alternatives for what I'm writing.
- **MODERN.** Java has traditionally been a slowly-evolving language, but the releases have picked up the pace in recent years, and now we see a new Long-Term Support release of Java every 3 years. Any Java **IDE** needs to be able to handle the modern features and frameworks of Java
- **ERGONOMIC.** This is a fancy way of saying that the **IDE** should **feel great to use**.

## INSTALLING AND SETTING JAVA ON WINDOWS 10

To install **JAVA** on **Ms Windows OS** and run **JAVA APPS** on **Ms Windows OS**, you need to get a copy of a **Java Development Toolkit** (**JDK**). **JDK** has both a **Java Runtime Environment** (**JRE**) and the **Java Virtual Machine** (**JVM**). **JDK** also has several other programming tools, e.g. **compiler** and **a Java web server**, for programmers who need to install Java on Windows. The following are the requirements that must be met to download and install Java on Windows:

- You use an updated **Windows 10** or **Windows 11** operating system (**OS**);
- You have enough rights to download the **JDK** and install Java on a **PC**.
- You can access the Internet to download Java's JDK;
- You have at least 500 MB of hard drive space for both download and installation

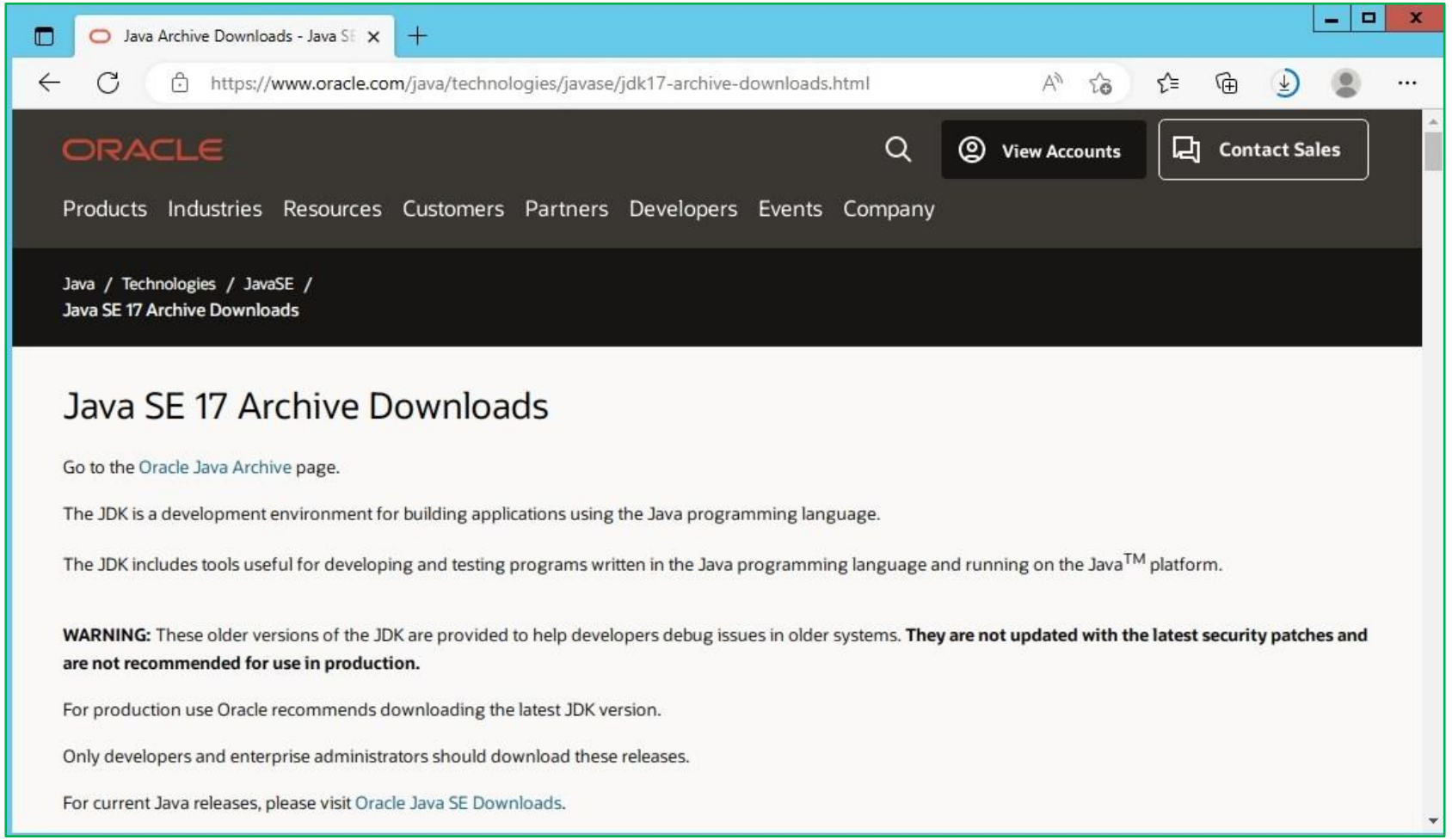
## HOW TO INSTALL JAVA ON WINDOWS

Follow these **five steps** to setup and install **Java** on **Windows**:

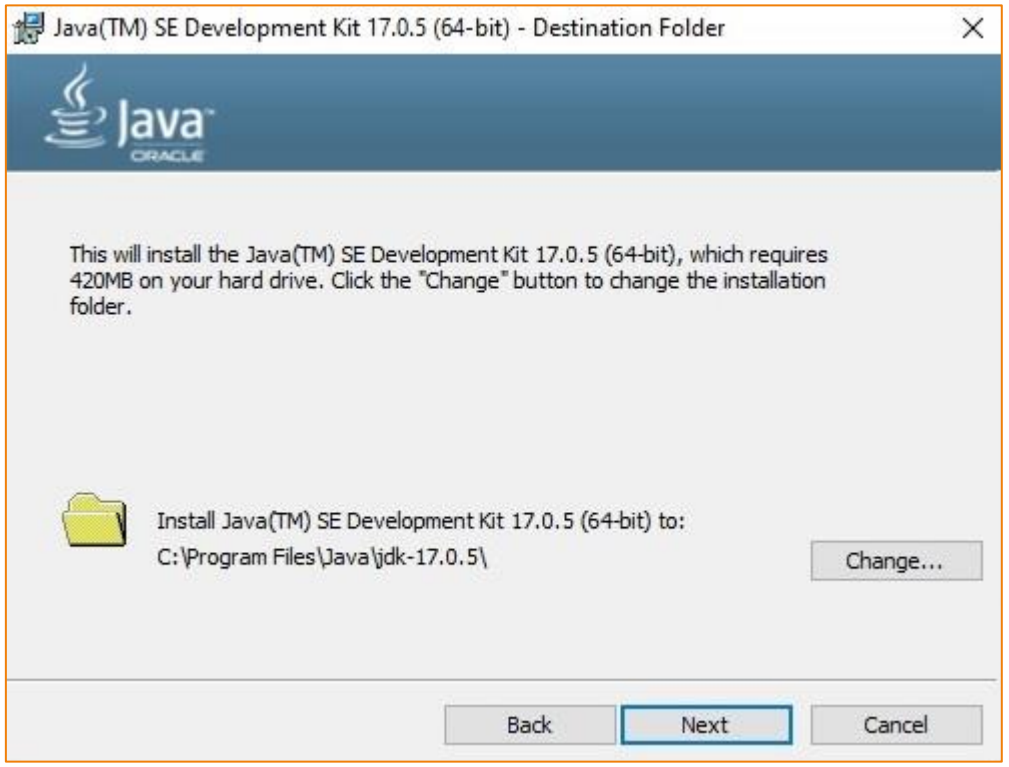
1. **Download Java** for Windows.
2. Run the **Java installer**.
3. Validate the **JAVA_HOME setting**.
4. **Confirm the Java PATH** variable was set properly.
5. **Run a JDK command** to verify Java install was a success.

# DOWNLOADING AND INSTALLING JAVA DEVELOPMENT KIT (JDK)

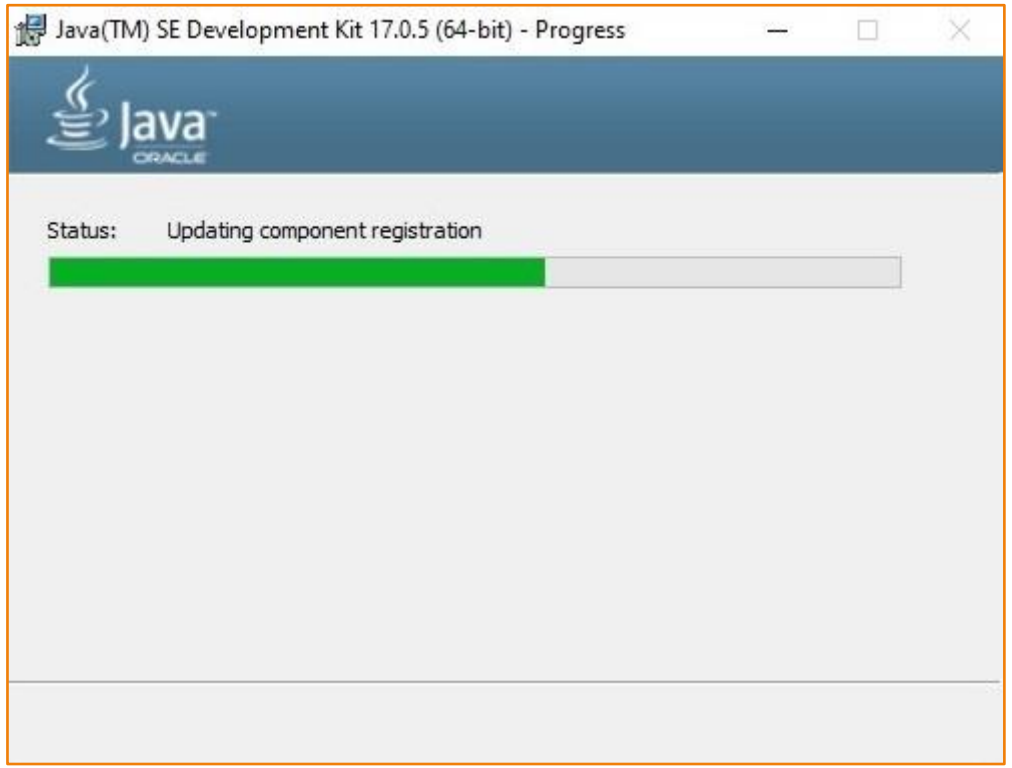




# DOWNLOADING AND INSTALLING JAVA DEVELOPMENT KIT (JDK)

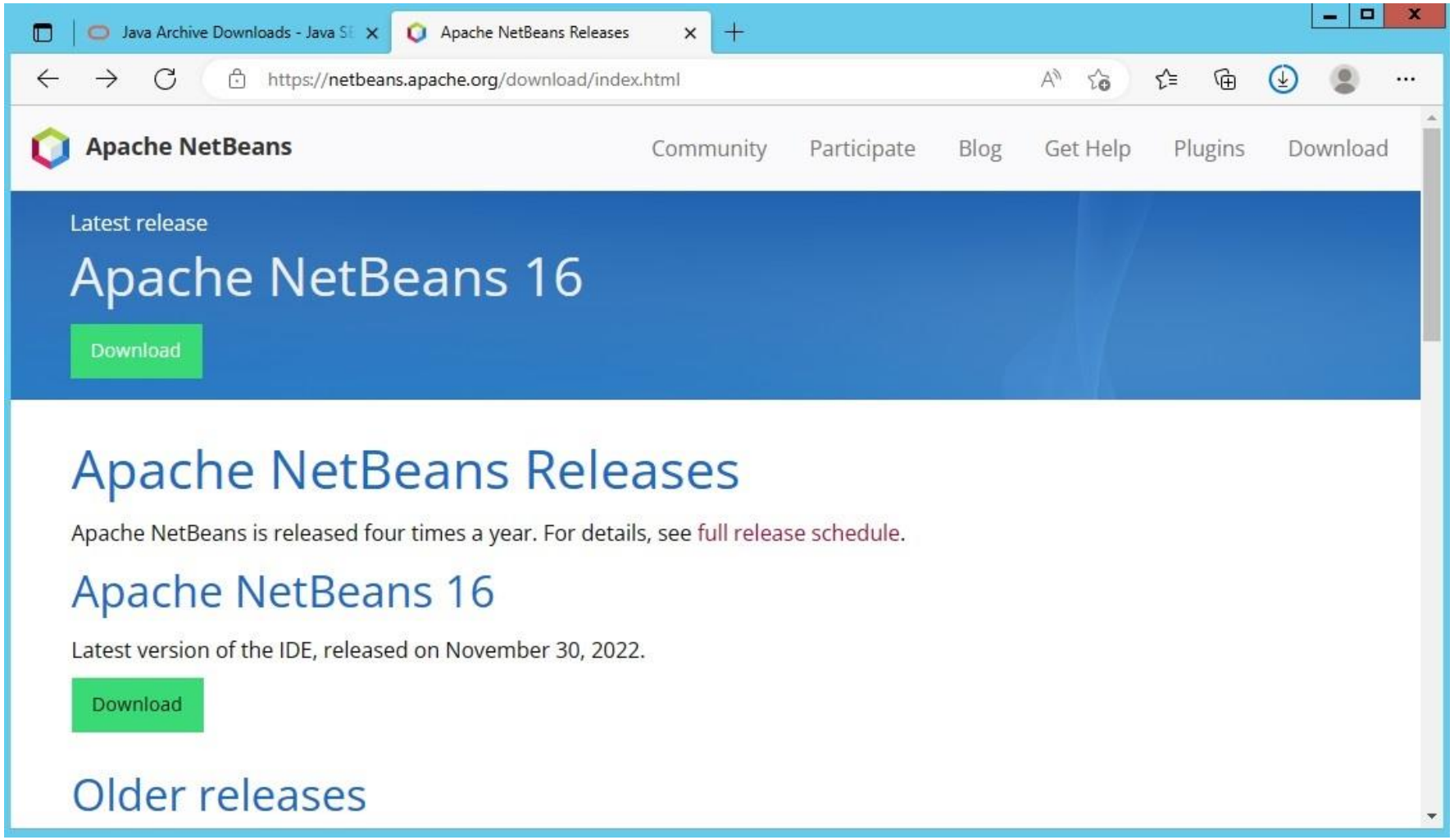



# DOWNLOADING AND INSTALLING JAVA DEVELOPMENT KIT (JDK)



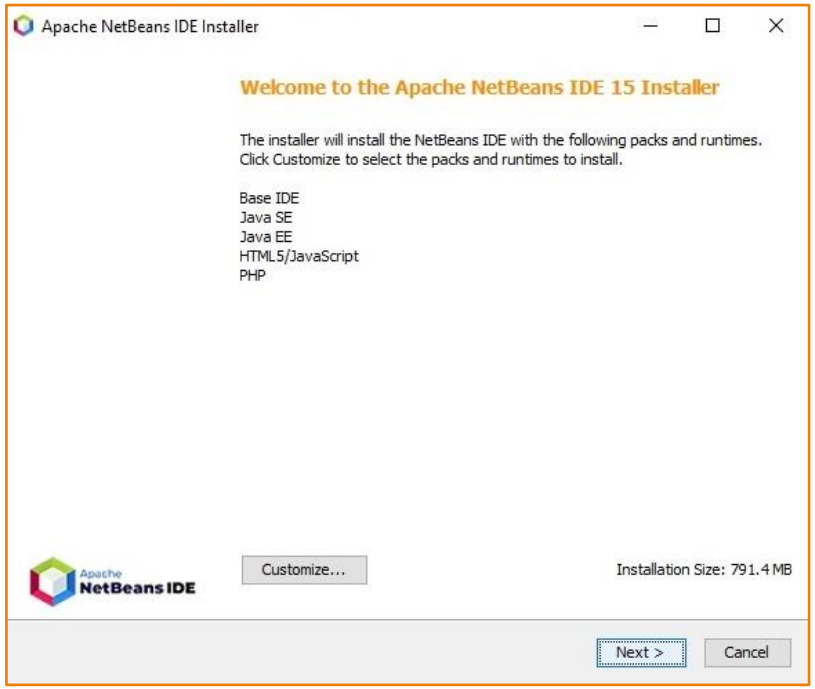

# DOWNLOADING AND INSTALLING THE NETBEANS IDE

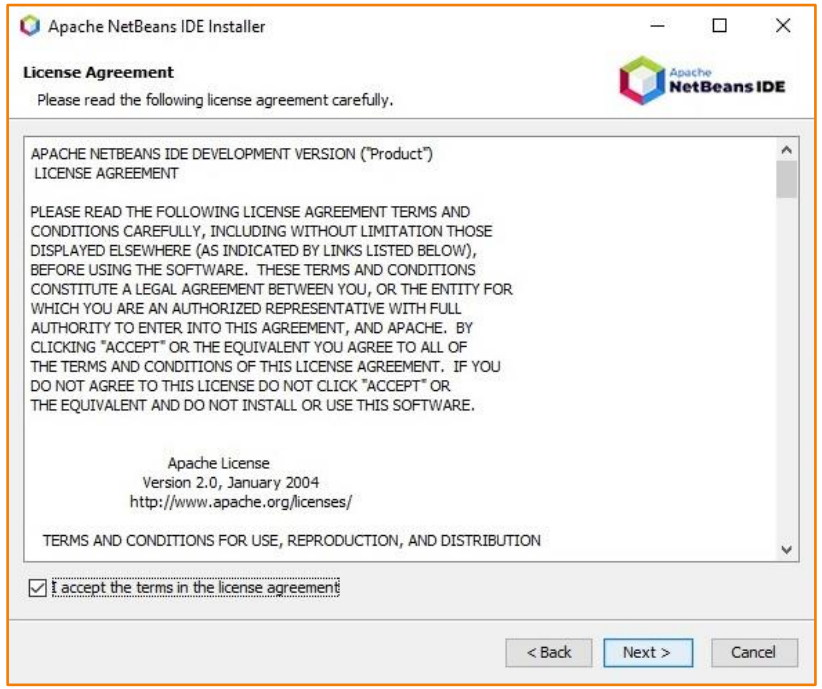


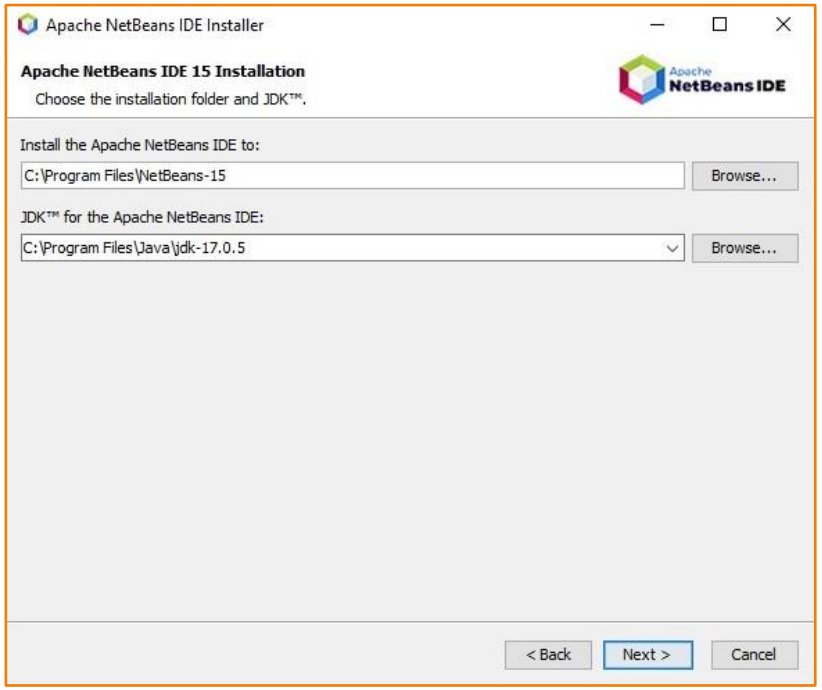


# DOWNLOADING AND INSTALLING THE NETBEANS IDE



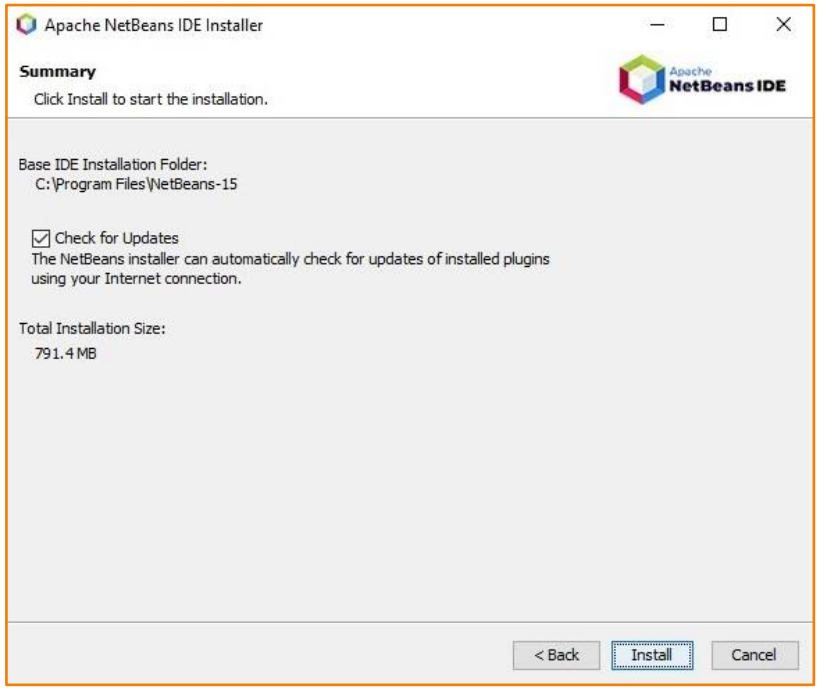


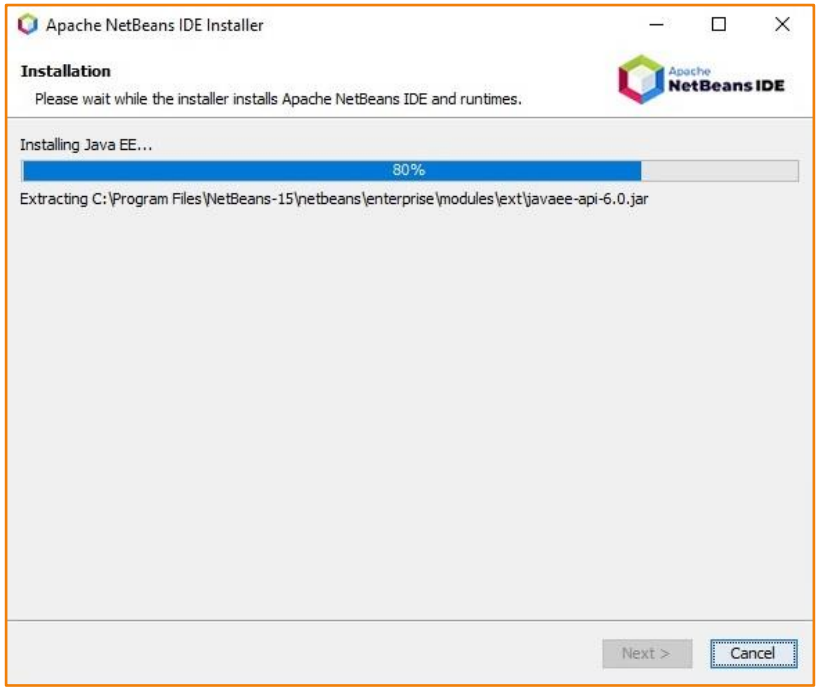



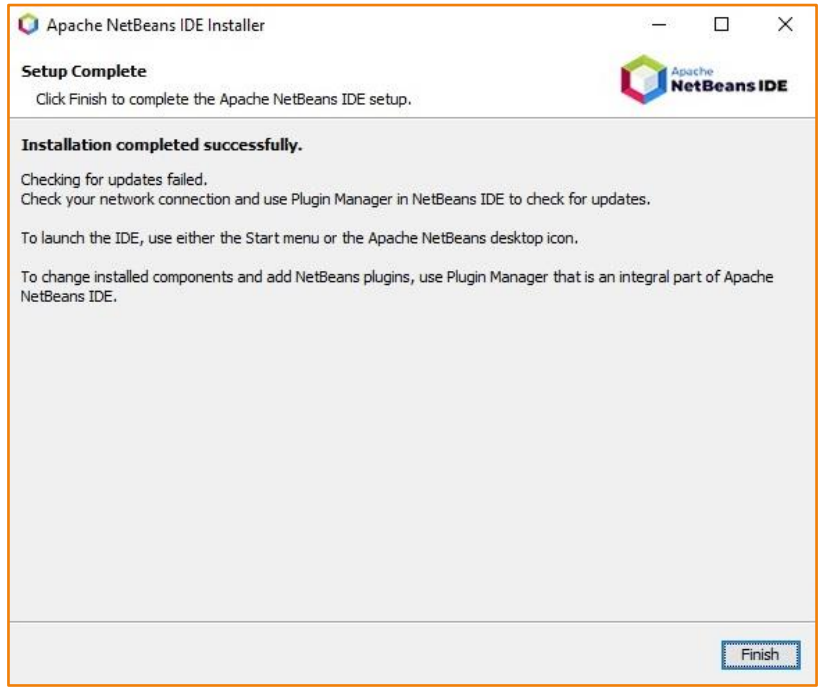



# DOWNLOADING AND INSTALLING THE NETBEANS IDE





# HOW TO SETUP THE JAVA ENVIRONMENT IN WINDOWS

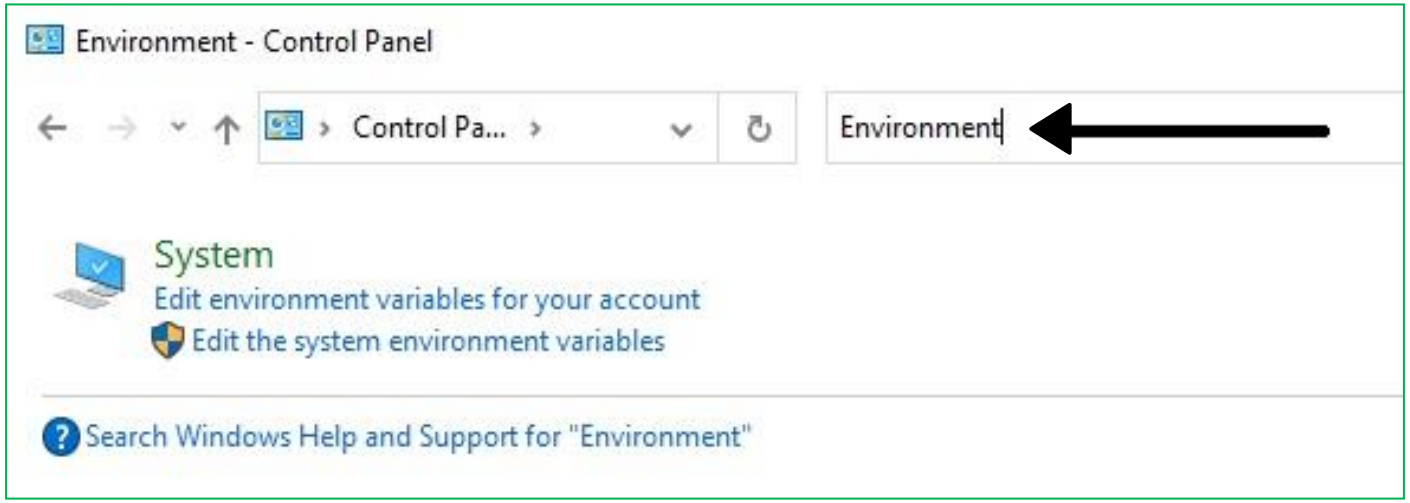
## WHAT IS JAVA_HOME?

**JAVA_HOME** is an environment variable which has installation directory of **Java development kit (JDK)** or **Java Runtime environment (JRE)**. This **environment variable** is setup at operating system level. **WHY DO YOU NEED A JAVA_HOME? JAVA_HOME** environment variable points to the directory where **JAVA** is installed on your computer system, so that java based applications can use the **JAVA_HOME** environment variable to locate **java executables.**

## HOW TO SET JAVA_HOME IN WINDOWS 10?

1.Locate the **downloaded JDK** on your Computer. 2.If you are using 64-bit java, then it will be in **C:\Program Files\Java**

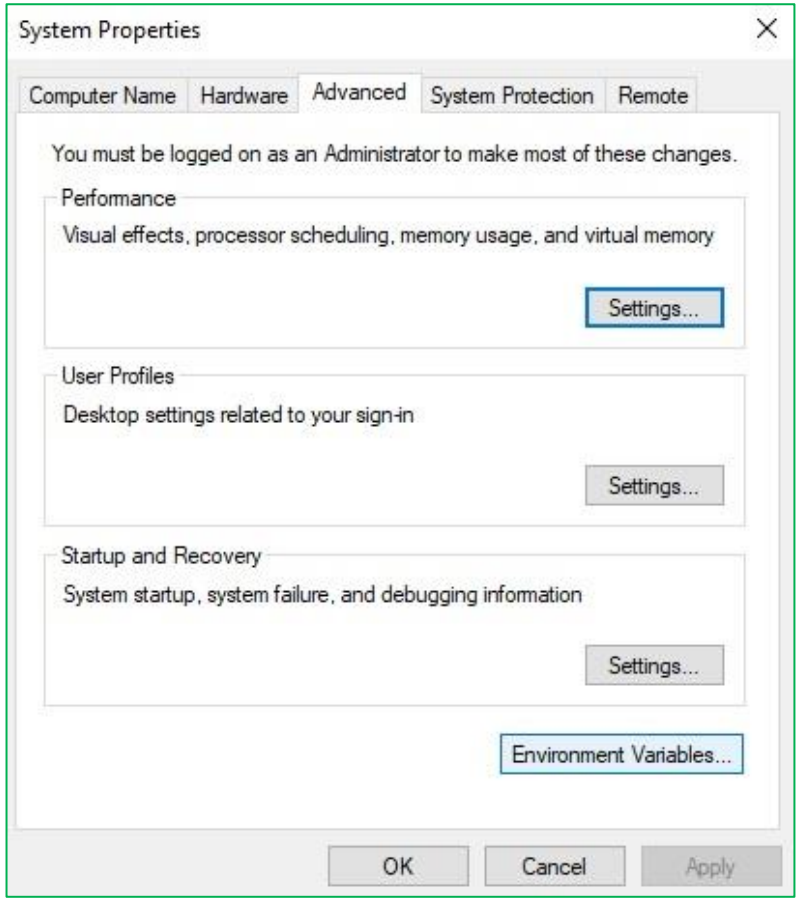
3.Open **Control Panel** and search, type **environment** and click on **Edit the system environment variables**

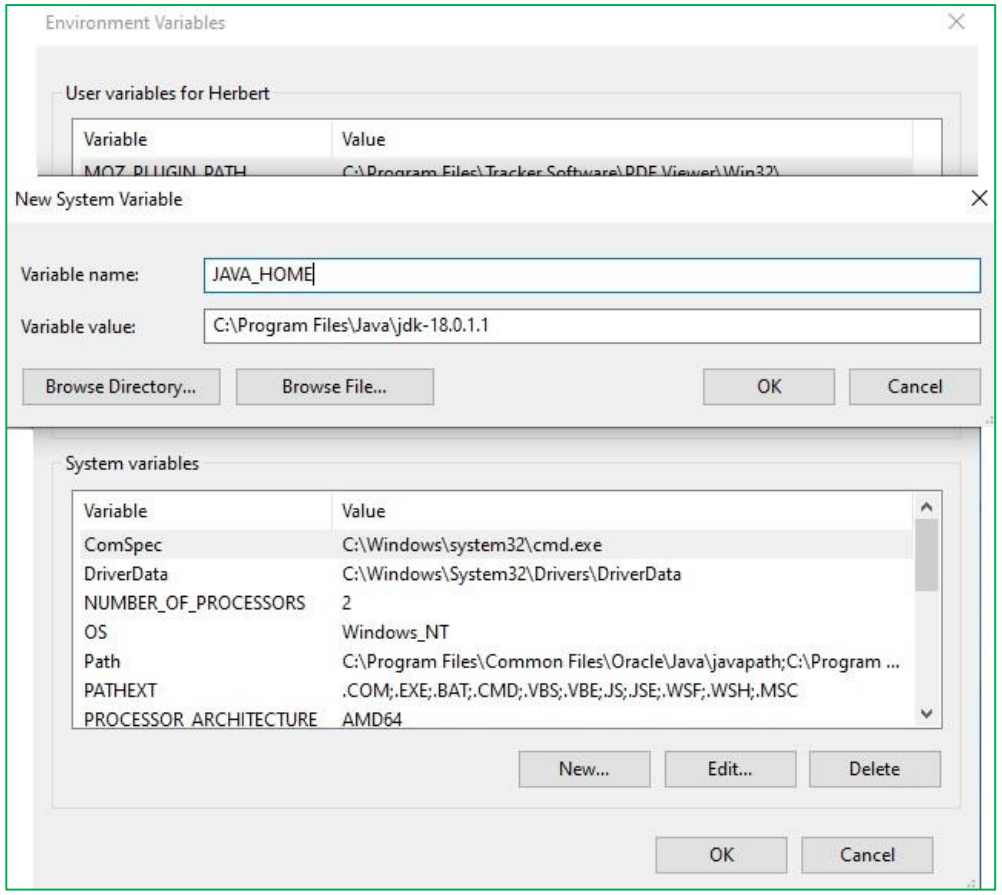


# HOW TO SETUP THE JAVA ENVIRONMENT IN WINDOWS



4.In **System properties** dialog, go to the **Advanced tab** and click on button **Environment Variables**.



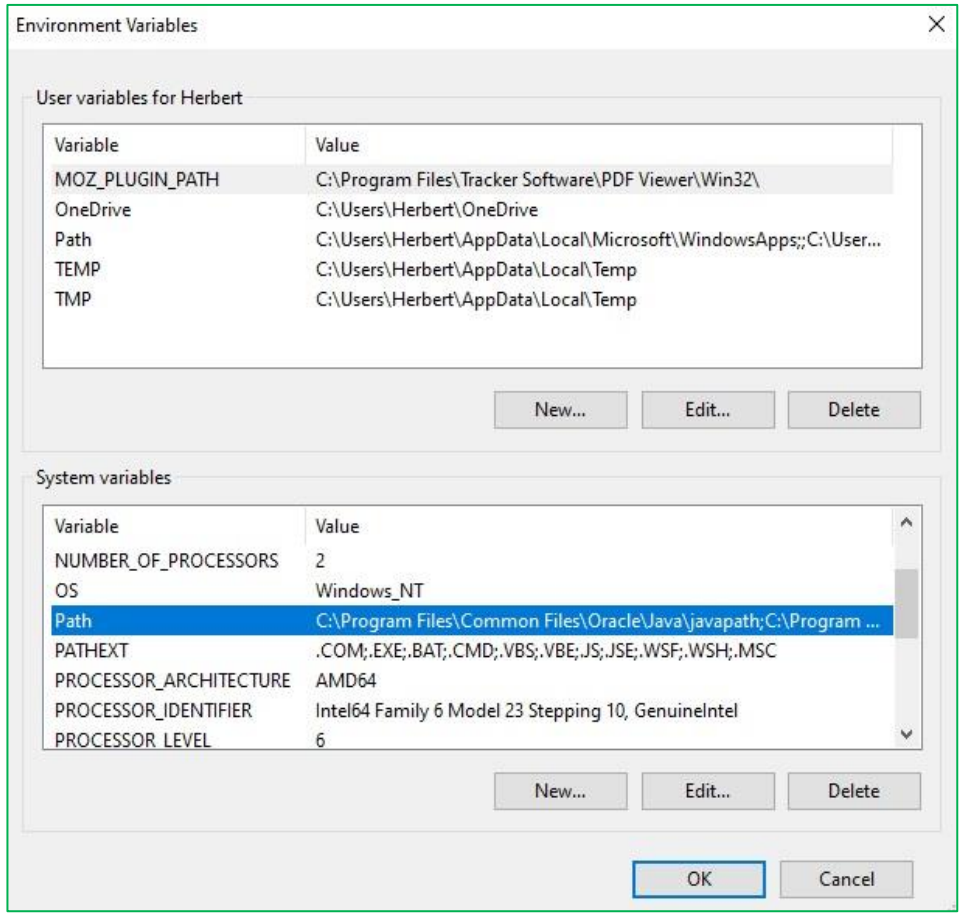



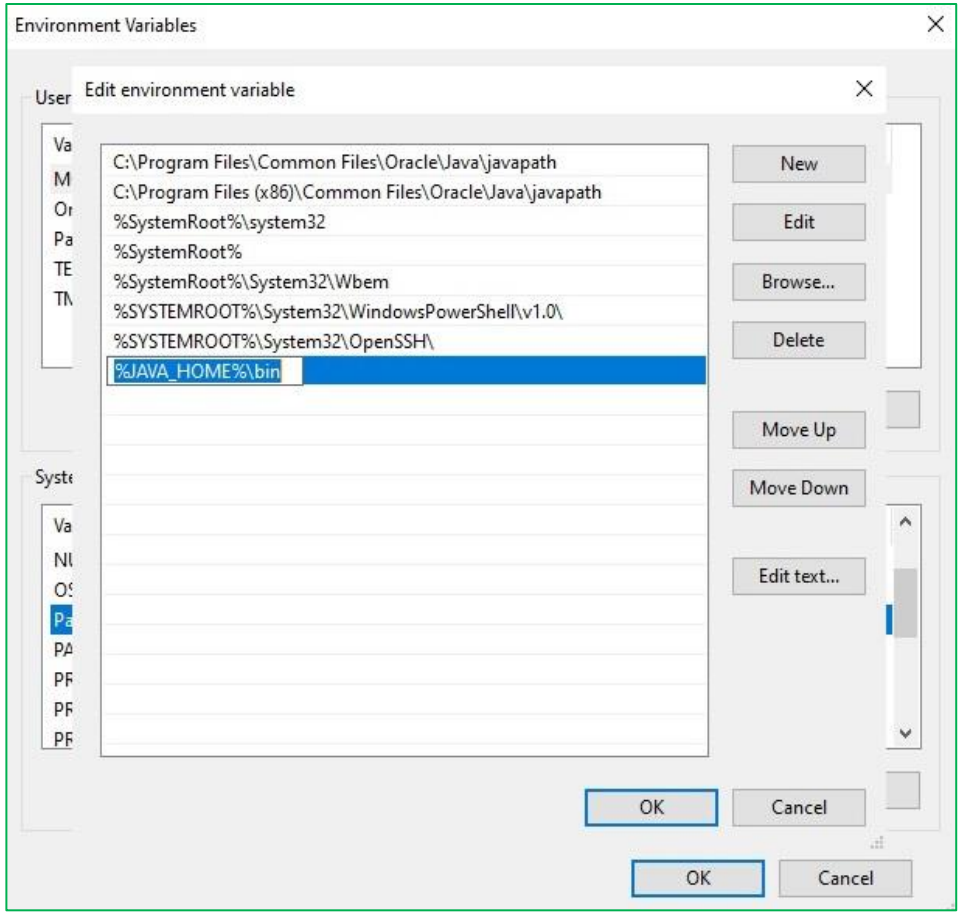

5.In **System variables**, click **NEW**... button to add **JAVA_HOME** environment variable. Type the variable name as **JAVA_HOME** and value as Java installation directory.

# HOW TO SETUP THE JAVA PATH ON WINDOWS



Now let's set the **JAVA** path environment variable in Windows 10. **1**.In **System variable** window, find the **Path** and click on **Edit....**

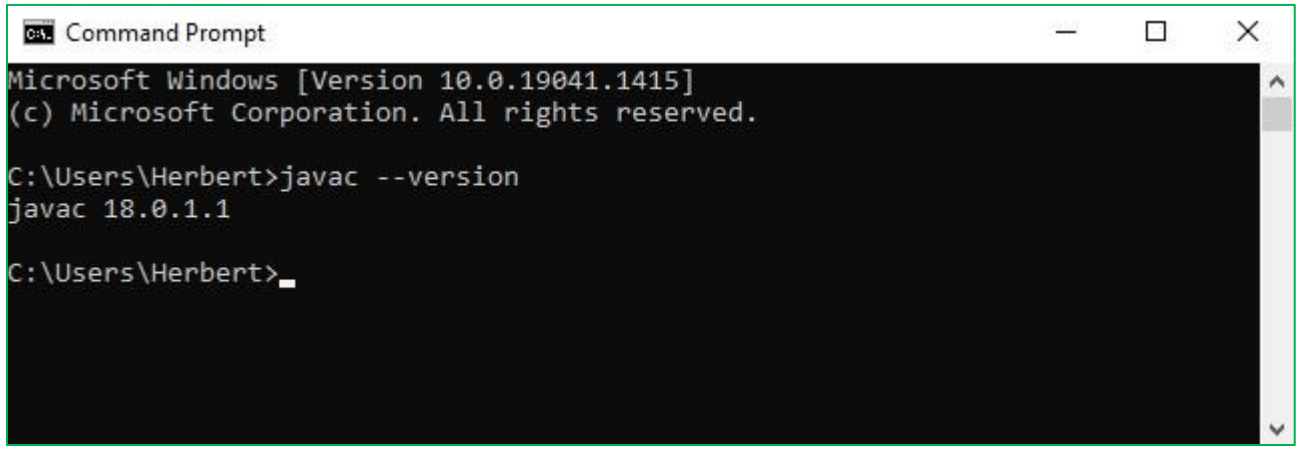


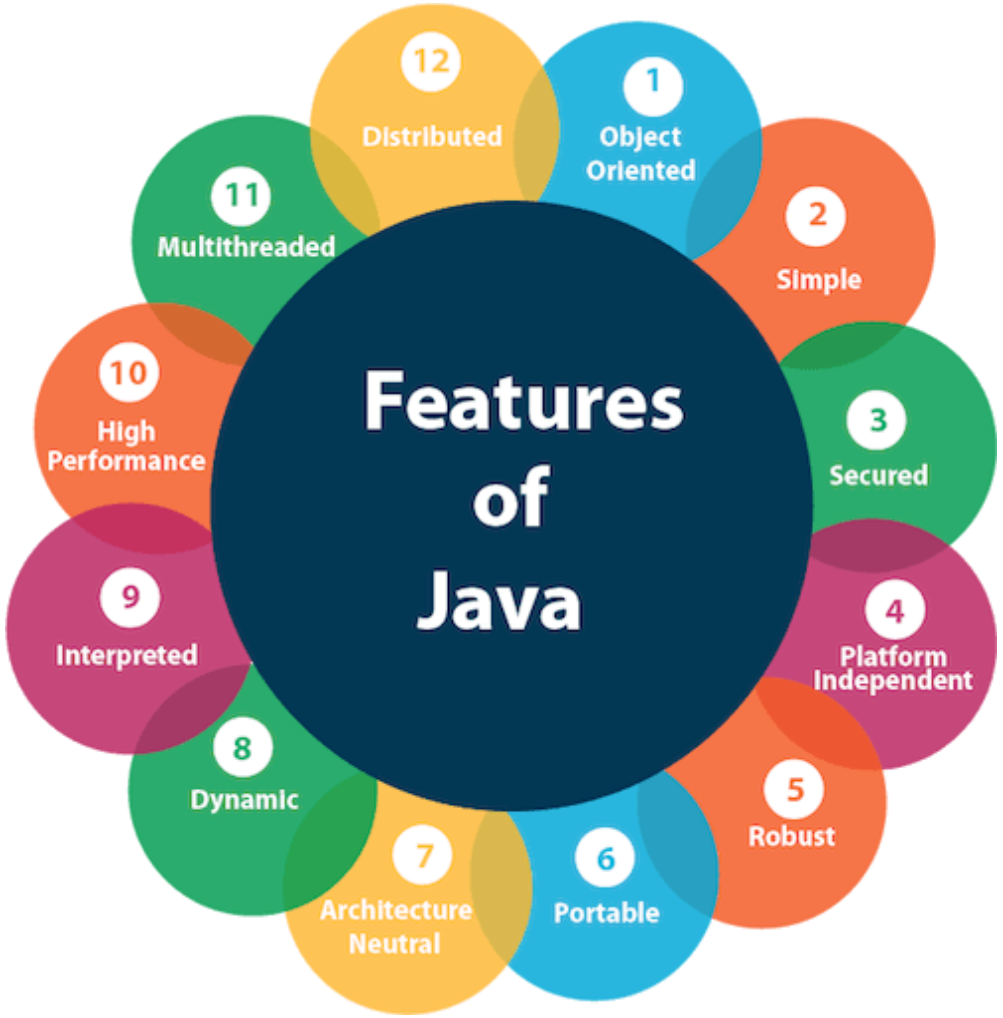

●**Double** click on **New** and add **%JAVA_HOME%\bin. The %** symbol locates the **JAVA_HOME** environment variable and **\bin** locates the **java.exe** and **h=javac.exe**

# HOW TO CHECK THE JAVA VERSION ON YOUR COMPUTER





# FEATURES OF THE JAVA PROGRAMMING LANGUAGE




The main objective of the **Java programming** language creation was to make it a **portable**, **very simple** and **secure** programming language. But there are also some other **excellent features** which play an important role in the popularity of this language. The features of Java are also known as **Java buzzwords**. A summary list of the most important features of the Java language is given here.

Thank you