

What is Java SE?

The SE stands for **Java Standard Edition** is a computing platform in which we can execute software, and it can be used for development and deployment of portable code for desktop and server environments. It has the Java programming language in use. It is part of Java software-platform family. Java SE has a variety of general purpose APIs and the Java Class Library. It is the core Java programming platform and provides all the libraries and APIs such as **java.lang**, **java.io**, **java.math**, **java.net**, **java.util** etc.

The following are the few APIs which Java SE has -

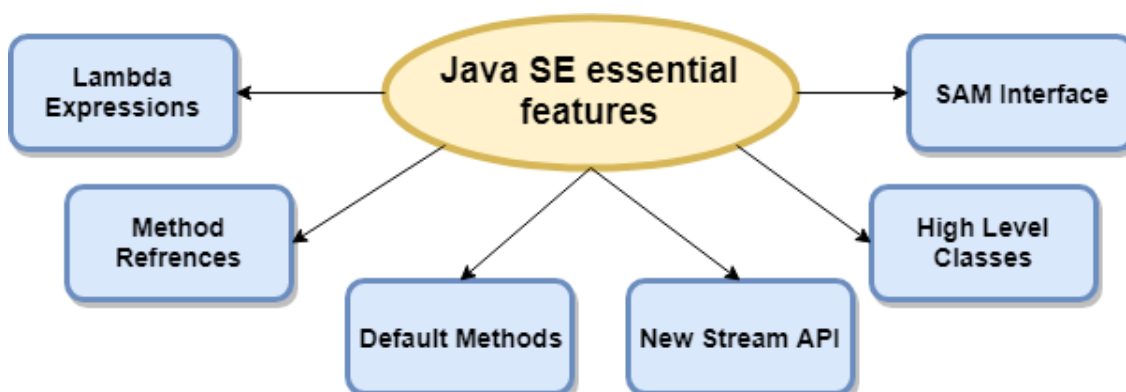
- **Applet**- An applet is a small application, especially a utility program performing one or a few simple functions. This API provides the classes necessary to create an applet. The applet framework contains two entities. One is applet and the other is applet context. The applet is an embeddable window with a few extra methods which the applet context uses to initialize, start and stop the applet.
- **AWT**- AWT stands for Abstract window toolkit. This package contains all the classes for creating a user interface and for painting graphics and images. Any UI object like button and scrollbar is called as a component.
- **RMI**- RMI stands for Remote Method Invocation enables the programmer to create distributed Java technology-based to Java technology-based application. RMI uses object serialization to marshal and unmarshal parameters and does not shorten types.
- **JDBC**- It stands for Java Database Connectivity. It allows you to fetch data from any data source be its relational database, be it a spreadsheet, be it flat file.
- **Swing**- Swing provides a set of 'lightweight' components mainly used for graphical user interface enhancement. All swing components and related classes should be accessed on the dispatching thread.
- **Collections**- Collection refers to a group of objects, known as its elements. There are many methods in the collections Framework interface which depend on the equals method. For example- the contains(Object o) method says that it will return true if the collection contains an element which satisfies the condition that (o==null ? e==null: o.equals(e))
- **xml binding**- It provides a run-time binding framework for client-side user application allowing the user to Marshall, unmarshal, and validation capabilities. JAXBContext is the client-entry point to the runtime binding framework.
- **JavaFX (Merged to Java SE 8)**- This contains several packages within it like javafx.animation(provides set of classes for ease of animation), javafx.application(provides set of classes for application life-cycle classes) and javafx.beans() etc.
- **Java 8 Collections Streaming API**- It contains classes to support functional-style operations on streams of elements. Such as map-reduce transformation on collections. Stream operations

are divided into two parts namely intermediate and terminal operations which are combined together to form pipelines.

- **Java 9 Reactive Streams API-** Reactive Stream initiative was taken by giants like Netflix in order to standardize the asynchronous exchange of data within an application. They are a part of JDK in the form of `java.util.concurrent.Flow.interfaces`.
- **Java 9 HTTP/2 API-** This API solved various problems which were with the previous HTTP/1.1 API. Previously we cannot have more than 6 connections at a time. This made it complex as other requests had to wait till previous calls get sorted. This got sorted with this API.

Java SE significant features

- Java SE has all the basic types and objects of the Java programming language.
- Java SE provides high-level classes used for networking, security, database access, GUI (Graphical User Interface) development, and XML parsing.
- It now provides static members inside interfaces.
- It provides with **ForEach()** method which can iterate through contiguous memory allocations and allows you to use it without knowing its size.
- It provides the `Collectors` class which allows accumulating elements into collections, summarizing data according to various criteria.
- It provides with the stream API which allows lazy computation (through this you can initialize only if they are required) and functional-style programming.
- It provides a class `Base64` for encryption and decryption.
- Performance has been improved for the `java.lang.String(byte[], *)` constructor and the `java.lang.String.getBytes()` method.
- A new class **`java.net.URLPermission`** has been added. It represents permission for accessing a resource defined by a given URL.
- It provides a single abstract method interface.



How to set up Java SE on windows

To develop or run Java applications, you need to download and install the Java SE Development Kit.

Step 1.) Download the Java SE latest release from the official site of Oracle.



```

struct TCS
{
    int x: 1;
    int y: 2;
    int z: 4;
    int w: 8;
}A;

int main()
{
    printf("%d",sizeof(A));
    return 0;
}
    
```

A 4 B 16
 C 8 D 15


What will be the output of above code in bytes? , if size of integer variable is consider to be as 4 bytes

Oracle Technology Network / Java / Java SE / Downloads

- Java SE
- Java EE
- Java ME
- Java SE Subscription
- Java Embedded
- Java Card
- Java TV
- Community
- Java Magazine

Overview Downloads Documentation Community Technologies Training

Java SE Downloads



DOWNLOAD

Java Platform (JDK) 12

Java Platform, Standard Edition

Java SE 12.0.1
 Java SE 12.0.1 is the latest release for the Java SE Platform
[Learn more](#)

- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle JDK License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme](#)

Oracle JDK

DOWNLOAD

Looking for Oracle OpenJDK builds?

- **Oracle Customers and ISVs targeting Oracle LTS releases:** Oracle JDK is Oracle's supported Java SE version for customers and for developing, testing, prototyping or demonstrating your Java applications.
- **End users and developers looking for free JDK versions:** [Oracle OpenJDK](#) offers the same features and performance as Oracle JDK under *

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources

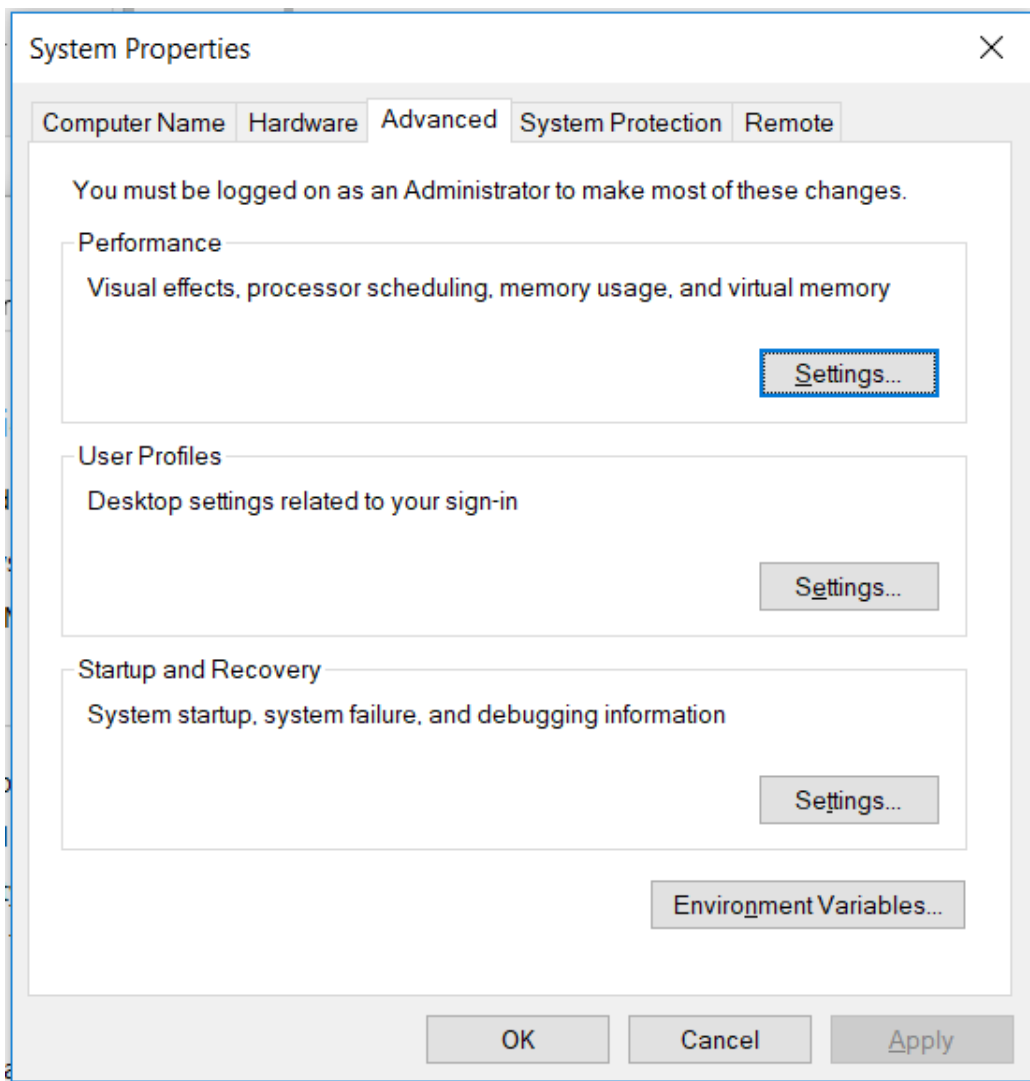
- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

Step 2.) After downloading the file, you will have an executable file downloaded. Run that file and keep everything as default and keep clicking next.

Step 3.) After completing the installation, your JDK and JRE would be downloaded in the program files folder.

Step 4.) After complete installation, you need to set up the environment variables.

Step 5.) Go to control panel -> System and Security -> System -> Advanced System Settings. The following dialog box will appear.



Step 6.) Click on Environment Variables, go to system variables, and double click on Path.

User variables for abhay

Variable	Value
OneDrive	C:\Users\abhay\OneDrive
Path	C:\Users\abhay\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\abhay\AppData\Local\Temp
TMP	C:\Users\abhay\AppData\Local\Temp

New...

Edit...

Delete

System variables

Variable	Value
configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	...

New...

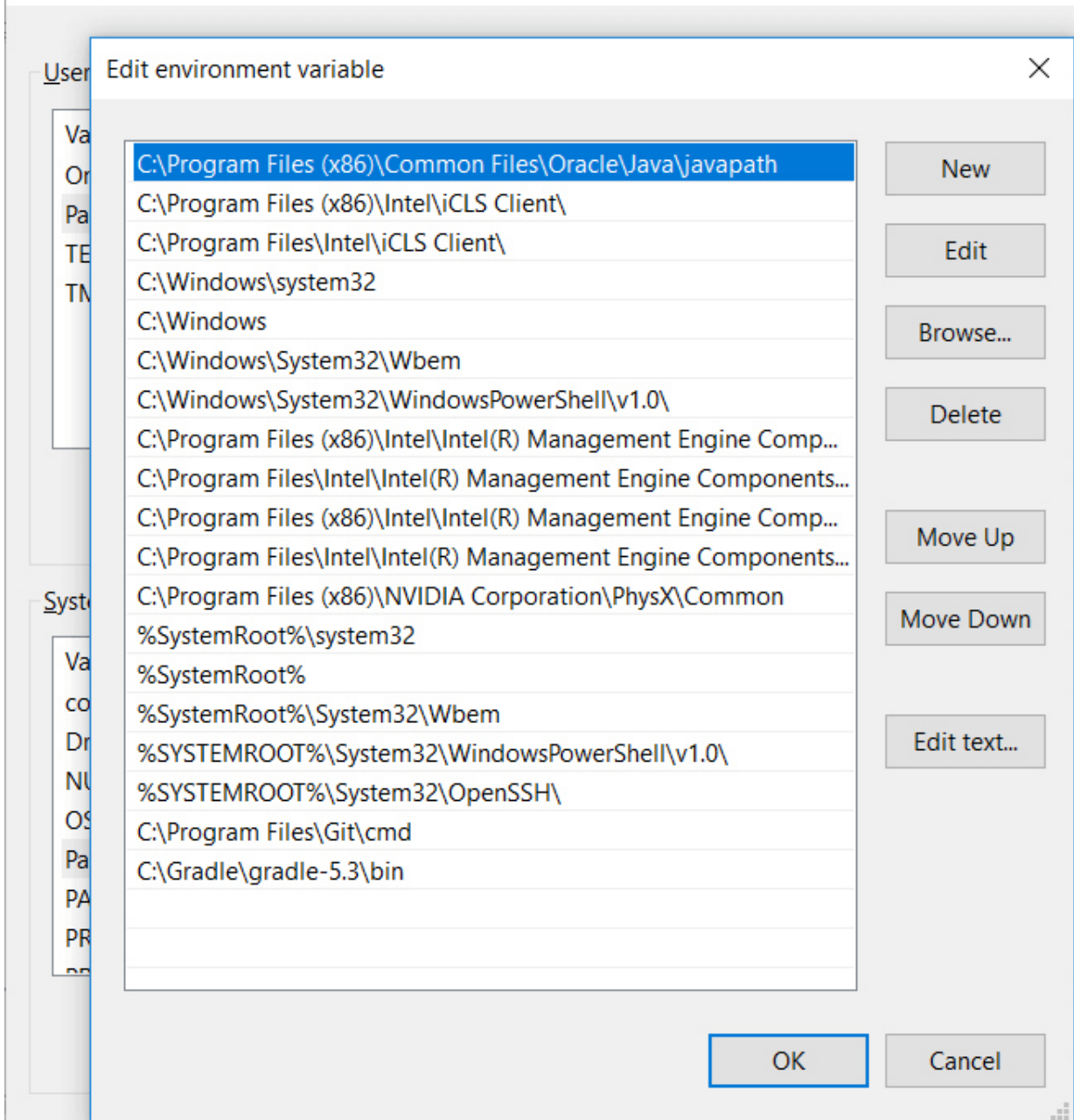
Edit...

Delete

OK

Cancel

Step 7.) Now add the path of your bin file present in the JDK file to the Path variable.



The set up Java environment is complete.

For development, you can use any IDE such as IntelliJ IDEA, Eclipse or NetBeans. Eclipse and NetBeans are free but IDEs but IntelliJ IDEA is paid IDE.

Java SE vs Java EE

Java EE refers to Java Enterprise Edition. It is a wrapper around the Java SE providing features for distributed computing, web services, reading and writing from a database in a transactional way. Java EE is a wrapper around Java SE providing certain additional functionalities and features along with that of Java SE.

Java SE	Java EE
Java SE provide basic functionalities such as defining types and objects.	Java EE provides APIs for running large scale applications.
SE is a standard Java specification	EE is built upon Java SE. It provides functionalities like web applications, servlets, etc.
It consists of class libraries, virtual machines, deployment environment programming.	Java EE is a structured application with a separate client, business, and Enterprise layers.
It is mostly used to develop APIs for Desktop Applications like antivirus software, game, etc.	It is mainly used for developing web applications.
It is suitable for beginning Java developers.	It is suitable for experienced Java developers who build enterprise-wide applications.
User authentication functionality is not provided with Java SE.	User authentication is provided by Java EE.

Future of Java SE

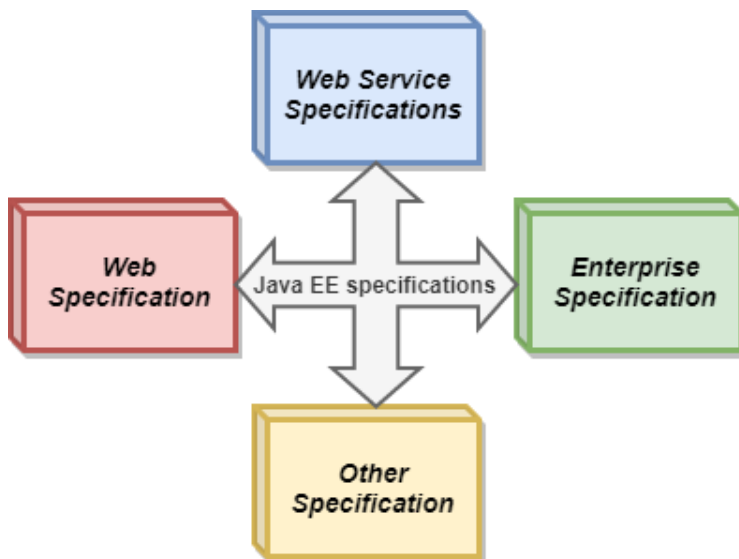
Java SE seems to be losing its charm as it does not provides many useful functionalities and is used for only basic features of Java programming language. The ongoing trend shows decreasing use of Java SE, and more people use other programming platforms such as Java EE, Java ME, and Python. Java SE has come up with features such as Application data-class Sharing, parallel full GC, garbage collector interface, local variable type interface which makes it stand strong with other programming platforms but still it is not up to the mark. Java is currently focusing on data management and machine learning ecosystems.

Java EE

The **Java EE** stands for **Java Enterprise Edition**, which was earlier known as J2EE and is currently known as Jakarta EE. It is a set of specifications wrapping around Java SE (Standard Edition). The Java EE provides a platform for developers with enterprise features such as distributed computing and web services. Java EE applications are usually run on reference run times such as **microservers** or **application servers**. Examples of some contexts where Java EE is used are e-commerce, accounting, banking information systems.

Specifications of Java EE

Java EE has several specifications which are useful in making web pages, reading and writing from database in a transactional way, managing distributed queues. The Java EE contains several APIs which have the functionalities of base Java SE APIs such as Enterprise JavaBeans, connectors, Servlets, Java Server Pages and several web service technologies.



1. Web Specifications of Java EE

- Servlet- This specification defines how you can manage HTTP requests either in a synchronous or asynchronous way. It is low level, and other specifications depend on it
- WebSocket- WebSocket is a computer communication protocol, and this API provides a set of APIs to facilitate WebSocket connections.
- Java Server Faces- It is a service which helps in building GUI out of components.
- Unified Expression Language- It is a simple language which was designed to facilitate web application developers.

2. Web Service Specifications of Java EE

- Java API for RESTful Web Services- It helps in providing services having Representational State Transfer schema.
- Java API for JSON Processing- It is a set of specifications to manage the information provided in JSON format.
- Java API for JSON Binding- It is a set of specifications provide for binding or parsing a JSON file into Java classes.
- Java Architecture for XML Binding- It allows binding of xml into Java objects.
- Java API for XML Web Services- SOAP is an xml based protocol to access web services over http. This API allows you to create SOAP web services.

3. Enterprise Specifications of Java EE

- Contexts and Dependency Injection- It provides a container to inject dependencies as in Swing.
- Enterprise JavaBean- It is a set of lightweight APIs that an object container possesses in order to provide transactions, remote procedure calls, and concurrency control.
- Java Persistence API- These are the specifications of object-relational mapping between relational database tables and Java classes.
- Java Transaction API- It contains the interfaces and annotations to establish interaction between transaction support offered by Java EE. The APIs in this abstract from low-level details and the interfaces are also considered low-level.
- Java Message Service- It provides a common way to Java program to create, send and read enterprise messaging system's messages.

4. Other Specifications of Java EE

- Validation- This package contains various interfaces and annotations for declarative validation support offered by Bean Validation API.
- Batch applications- It provides the means to run long running background tasks which involve a large volume of data and which need to be periodically executed.
- Java EE Connector Architecture- This is a Java-based technological solution for connecting Java servers to Enterprise Information System.

Setting up Java EE

Requirements

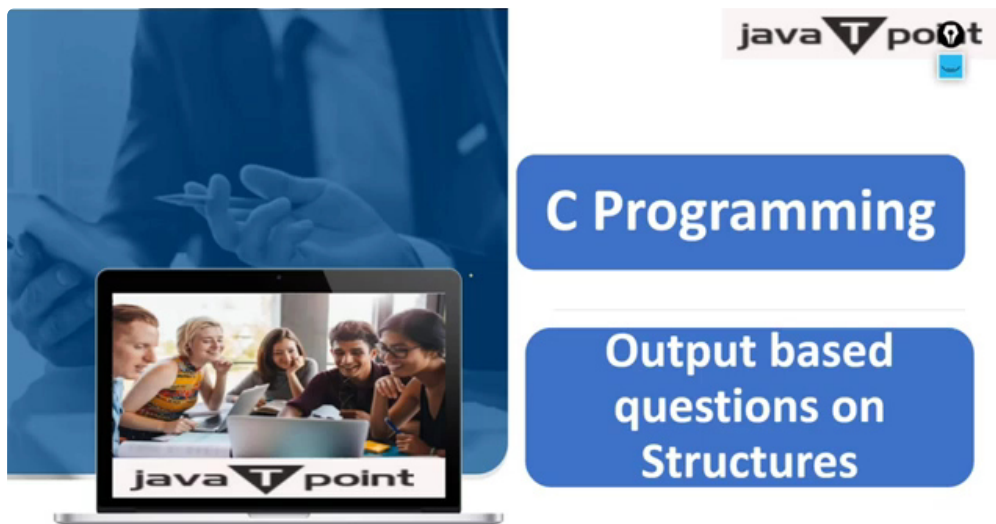
For the installation of latest SDK of Java EE which is Java EE 6 SDK on windows, you require to have a minimum memory of 1GB, minimum Disk space of 250MB free and JVM Java SE 6. For setting up Java EE, you require to have a JDK and then have an IDE preferably Eclipse as it is free.

Install a Java Development Kit

1. Browse to [Oracle's Java SE Development Kit downloads](#)
2. In the section titled **Java SE Development Kit 9.0.1**, read the license and, if you agree, click **Accept License Agreement**
3. Still, in that section, click on **JDK-9.0.1_windows-x64_bin.exe** (or the right download for your OS)
4. Run the downloaded JDK installer, using **Run As Administrator**
5. Add the Windows (or Linux) Environment Variable **JAVA_HOME**. Set it to the root folder of your newly-installed JDK, which looks like C:\Program Files\Java\jdk1.8.0_51.

Install Eclipse for Java EE

1. Browse to [Eclipse Downloads](#)



2. Click on the Download button under **Get Eclipse**.
3. On the resulting page, click on the Download button.

Note: The version of Eclipse (32-bit or 64-bit) which you download should match the version of your JDK. You installed JDK-9.0.1_windows-x64 above, so download the 64-bit Eclipse.

4. Run the downloaded installer using **Run as Administrator**.
5. Choose the version of Eclipse you wish to install. **Eclipse IDE for Java EE developers** is preferable for Java work.
6. If the installation fails, try again with real-time virus scanning temporarily turned off. Remember to turn it on again when it's done.

Java SE vs Java EE

Java SE refers to standard edition and contains basic functionalities and packages required by a beginner or intermediate-level programmer. Java EE is an enhanced platform and a wrapper around Java SE. It has the edge over Java SE and also has a variety of aspects in which it outshines other features.

Java SE	Java EE
Java SE provide basic functionalities such as defining types and objects.	Java EE facilitates development of large scale applications.
SE is a normal Java specification	EE is built upon Java SE. It provides functionalities like web applications, and Servlets.
It has features like class libraries, deployment environments, etc.	Java EE is a structured application with a separate client, business, and Enterprise layers.
It is mostly used to develop APIs for Desktop Applications like antivirus software, game, etc.	It is mainly used for developing web applications.
Suitable for beginning Java developers.	Suitable for experienced Java developers who build enterprise-wide applications.
It does not provide user authentication.	It provides user authentication.

What is Java ME?

The Java ME stands for **Java Micro Edition**. It is a development and deployment platform of portable code for embedded and mobile devices (sensors, gateways, mobile phones, printers, TV set-top boxes). It is based on **object-oriented Java**. The Java ME has a robust user interface, great security, built-in network protocols, and support for applications that can be downloaded dynamically. Applications which are developed on Java ME are portable and can run across various devices and can also leverage the native capabilities of the device.

Java ME SDK

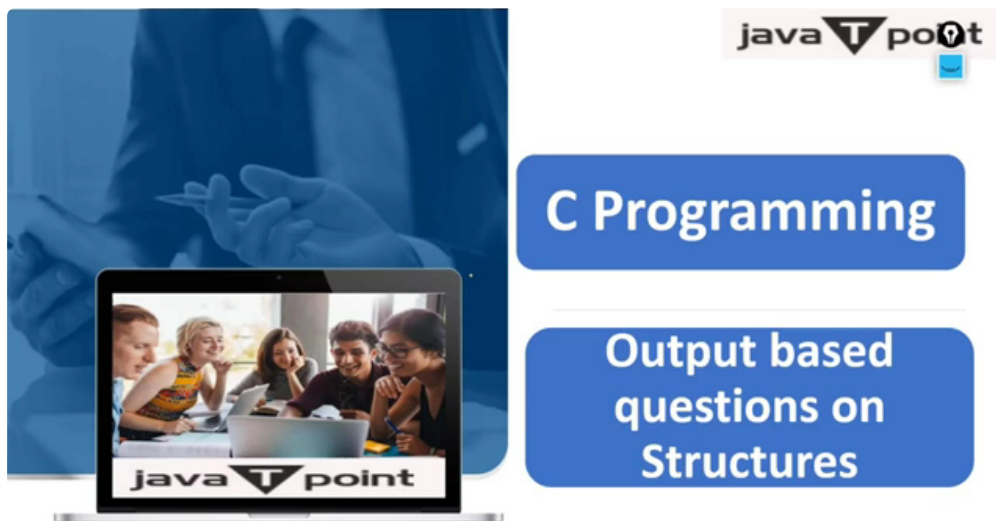
Java ME Software Development Kit (SDK) provides the standalone runtime environment and various utilities required for development Java ME applications. It combines **the Connected Limited Device Configuration (CLDC)** and **the Connected Device Configuration (CDC)** into one single environment.

Java ME Embedded

Java ME embedded is a run time platform that leverages the Java ME technologies that are deployed to billions of devices across the Internet of Things. It is designed by keeping in mind that the applications developed can be portable to various devices while being resource-efficient and keeping the demands from the underlying platform low.

How Java ME is organized

The generic computing devices usually consist of hardware such as display, permanent storage, keyboard, etc. but the small computing devices are not like this. Some of them don't have permanent storage, and some don't even have a permanent display. As Java ME target a variety of small computing devices, this problem is handled by it by using a two-fold approach.



- Firstly, there is a Java Run-time Environment and other core classes that are defined to target specifically the device on which it is operating. This is referred to as configurations.
- Secondly, a profile is defined as a set of similar small computing devices. A profile has several classes within it which are made to implement features found on a related group of small computing devices.

Java ME architecture

The Java ME architecture helps in scaling an application based on the constraints provided by the small computing device. Java ME does not simply replace the operating system, rather it stacks up layers on the native operating system and makes an environment for the application to run. These layers are collectively named as **Connected Limited Device Configuration (CLDC)**.

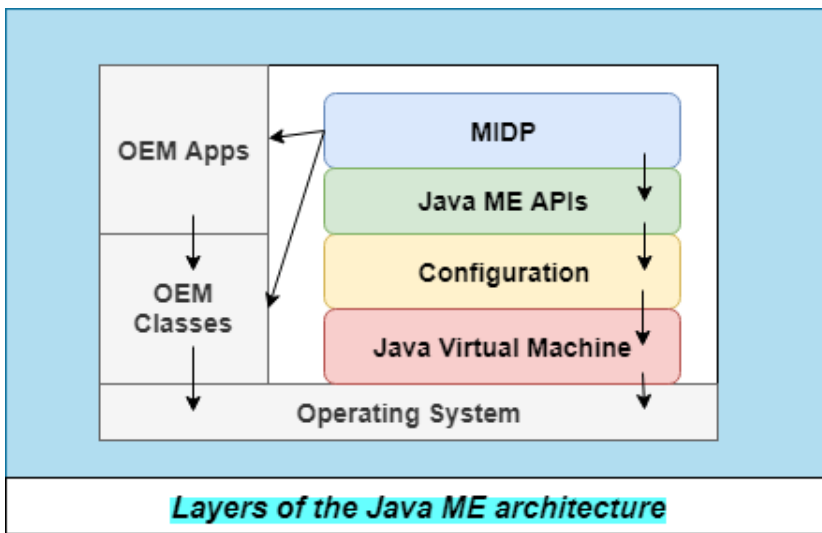
The first layer is the **configuration layer** that includes the Java Virtual Machine. This layer interacts directly with the native operating system and builds the connection between the profile and the JVM.

The second layer is the **profile** which contains the minimum set of APIs for the small computing device. The profile contains a set of classes which are made to implement the features of a related group of small computing devices.

The third layer is the **Mobile Information Device Profile (MIDP)**. The MIDP layer consists of APIs which are for user network connections, persistence storage, and the user interface. It also has access to Connected Language Device Configuration (CLDC) and Mobile Information Device Profile (MIDP) libraries.

A small computing device has two components supplied by the Original Equipment Manufacturer (OEM). They are, namely OEM apps and OEM classes. The MIDP communicates with the OEM classes to gain access to features like sending and receiving messages and accessing device-specific persistent data. OEM applications are small programs such as address book etc.

NOTE: Dependency of MIDP on OEM apps and OEM classes makes the application less portable as OEM feature are different for all manufacturers, and not all of them use the same classes and apps.



Java ME Configurations

Java ME configurations specify a JVM and certain core APIs which are directed towards a certain set of devices. There are two configurations available with Java ME, namely Connected Device Configuration (CDC) and Connected Limited Device Configuration. The Java ME configurations and profiles are based on memory and for small devices based on volatile and non-volatile memory.

Java ME vs. Java EE

Java ME	Java EE
Java ME facilitates the development of applications for small computing devices such as embedded systems, sensors, etc.	Java EE facilitates development of large scale applications.
Java ME is built upon Java SE. Provides functionalities such as networking, communication with native operating systems of mobile devices	EE is also built upon Java SE. It provides functionalities like web applications, servlets, etc.
It has features which make applications portable and which can run on various devices. It deals with many constraints, such as a small battery, small display, etc.	Java EE is a structured application with a separate client, business, and Enterprise layers.
It is mostly used to develop mobile applications.	It is mainly used for developing web applications.
It is suitable for developers targeting diversified operating systems and a variety of devices.	It is suitable for experienced Java developers who build enterprise-wide applications.
It does not provide user authentication.	It provides user authentication.