Contents

[2 Basics of Java 3](#_Toc138189727)

[2.1 Chapter 1 History of java 3](#_Toc138189728)

[2.2 Versions of Java 3](#_Toc138189729)

[2.3 Installation and Environmental setup 6](#_Toc138189730)

[2.4 Packages in Java 6](#_Toc138189731)

[2.4.1 Built-in packages 7](#_Toc138189732)

[2.4.2 User-defined packages 8](#_Toc138189733)

[2.5 Access Modifiers 8](#_Toc138189734)

# 

# Basics of Java

## 

## Chapter 1 History of java

Java is Object Oriented programming g language used to develop mobile web and desktop based application developed by James Gosling in the early 1990s. The team initiated this project to develop a language for digital devices such as set-top boxes, television, etc. Originally C++ was considered to be used in the project but the idea was rejected for several reasons (For instance C++ required more memory). Gosling endeavored to alter and expand C++ however before long surrendered that for making another stage called Green. James Gosling and his team called their project “Green talk” and its file extension was .gt and later became to known as “OAK”. Why “Oak”? The name Oak was used by Gosling after an oak tree that remained outside his office. Also, Oak is an image of solidarity and picked as a national tree of numerous nations like the U.S.A., France, Germany, Romania, etc. But they had to later rename it as “JAVA” as it was already a trademark by Oak Technologies. “JAVA” Gosling and his team did a brainstorm session and after the session, they came up with several names such as JAVA, DNA, SILK, RUBY, etc. Java name was decided after much discussion since it was so unique. The name Java originates from a sort of espresso bean, Java. Gosling came up with this name while having a coffee near his office. Java was created on the principles like Robust, Portable, Platform Independent, High Performance, Multithread, etc. and was called one of the Ten Best Products of 1995 by the TIME MAGAZINE. Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. The Java language has experienced a few changes since JDK 1.0 just as various augmentations of classes and packages to the standard library. In Addition to the language changes, considerably more sensational changes have been made to the Java Class Library throughout the years, which has developed from a couple of hundred classes in JDK

## Versions of Java

| **Version** | **Release Date** | **Major changes** |
| --- | --- | --- |
| JDK Beta | 1995 |  |
| JDK 1.0 | January 1996 | The Very first version was released on January 23, 1996. The principal stable variant, JDK 1.0.2, is called Java 1. |
| JDK 1.1 | February 1997 | Was released on February 19, 1997. There were many additions in JDK 1.1 as compared to version 1.0 such as   * A broad retooling of the AWT occasion show * Inner classes added to the language * JavaBeans * JDBC * RMI |
| J2SE 1.2 | December 1998 | “Play area” was the codename which was given to this form and was released on 8th December 1998. Its real expansion included: strictfp keyword   * the Swing graphical API was coordinated into the centre classes * Sun’s JVM was outfitted with a JIT compiler out of the blue * Java module * Java IDL, an IDL usage for CORBA interoperability * Collections system |
| J2SE 1.3 | May 2000 | Codename- “KESTREL” Release Date- 8th May 2000 Additions:   * Hotspot JVM included * Java Naming and Directory Interface * JPDA * Java Sound * Synthetic proxy classes |
| J2SE 1.4 | February 2002 | Codename- “Merlin” Release Date- 6th February 2002 Additions: Library improvements   * Regular expressions modelled after Perl regular expressions * The image I/O API for reading and writing images in formats like JPEG and PNG * Integrated XML parser and XSLT processor (JAXP) (specified in JSR 5 and JSR 63)   Public Support and security updates for this version ended in October 2008. |
| J2SE 5.0 | September 2004 | Codename- “Tiger” Release Date- “30th September 2004” Originally numbered as 1.5 which is still used as its internal version. Added several new language features such as:   * for-each loop * Generics * Autoboxing * Var-args |
| JAVA SE 6 | December 2006 | Codename- “Mustang” Released Date- 11th December 2006 Packaged with a database supervisor and encourages the utilization of scripting languages with the JVM. Replaced the name J2SE with java SE and dropped the .0 from the version number. Additions:   * Upgrade of JAXB to version 2.0: Including integration of a StAX parser. * Support for pluggable annotations (JSR 269). * JDBC 4.0 support (JSR 221) |
| JAVA SE 7 | July 2011 | Codename- “Dolphin” Release Date- 7th July 2011 Added small language changes including strings in the switch. The JVM was extended with support for dynamic languages. Additions:   * Compressed 64-bit pointers. * Binary Integer Literals. * Upstream updates to XML and Unicode. |
| JAVA SE 8 | March 2014 | Released Date- 18th March 2014 Language level support for lambda expressions and default methods and a new date and time API inspired by Joda Time. |
| JAVA SE 9 | September 2017 | Release Date: 21st September 2017 Project Jigsaw: designing and implementing a standard, a module system for the Java SE platform, and to apply that system to the platform itself and the JDK. |
| JAVA SE 10 | March 2018 | Released Date- 20th March Addition:   * Additional Unicode language-tag extensions * Root certificates * Thread-local handshakes * Heap allocation on alternative memory devices * Remove the native-header generation tool – javah. * Consolidate the JDK forest into a single repository. |
| JAVA SE 11 | September 2018 | Released Date- 25th September, 2018 Additions-   * Dynamic class-file constants * Epsilon: a no-op garbage collector * The local-variable syntax for lambda parameters * Low-overhead heap profiling * HTTP client (standard) * Transport Layer Security (TLS) 1.3 * Flight recorder |
| JAVA SE 12 | March 2019 | Released Date- 19th March 2019 Additions-   * Shenandoah: A Low-Pause-Time Garbage Collector (Experimental) * Microbenchmark Suite * Switch Expressions (Preview) * JVM Constants API * One AArch64 Port, Not Two * Default CDS Archives |

## Installation and Environmental setup

JDK URL :- <https://www.oracle.com/in/java/technologies/downloads/>

Eclipse URL:- <https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers>

Git URL:- <https://github.com/ChanduGudipati/Java_FS>

## Packages in Java

A Java package is a collection of similar types of sub-packages, interfaces, and classes. In Java, there are two types of packages: built-in packages and user-defined packages. The package keyword is used in Java to create Java packages.

Many in-built packages are available in Java, including util, lang, awt, javax, swing, net, io, sql, etc. We can import all members of a package using package name.\* statement.

Let’s find out why we even need packages in the first place. Say you have a laptop and a bunch of data you want to store. Data includes your favorite movies, songs, and images.

So, do you store them all in a single folder or make a separate category for each one and store them in their corresponding folder?

It is obvious that anyone would like to create separate folders for images, videos, songs, movies, etc. And the reason is the ease of accessibility and manageability.

### Built-in packages

When we install Java into a personal computer or laptop, many packages get installed automatically. They all are unique on their own and have the capabilities to handle multiple tasks.

Due to this, we don’t have to start building everything from scratch. Some of the examples of built-in packages are listed below.

Java.lang

Java.io

Java.util

Java.applet

Java.awt

Java.net

User-defined packages are those that developers create to incorporate different needs of applications. In simple terms, User-defined packages are those that the users define. Inside a package, you can have Java files like classes, interfaces, and a package as well (called a sub-package).

### User-defined packages

In Eclipse we can create a user defined package by new 🡪package

And the (.) will creates the sub folders.

Package should be in small letters and all the characters will are allowed to create the folders in the windows and Linux are elidable to create the package

## Access Modifiers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Modifier** | **Within Class** |  | **Within Package** | **Outside Package by subclass only** | **Outside Package** |
| **Private** | Yes |  | No | No | No |
| **Default** | Yes |  | Yes | No | No |
| **Protected** | Yes |  | Yes | Yes | No |
| **Public** | Yes |  | Yes | Yes | Yes |