## **Java latest updates**

Every six months, the version number of Java gets incremented but **Every three years, Oracle has a long-term support (LTS) release (now LTS version will be for every 2 years)**

|  |  |
| --- | --- |
| 2027 | java 29 |
| 2025 | java 25 |
| 2023 | java 21 released |
| 2021 | Java 17 released in 2021 (so here also no more java EE – its is jakarta EE)  ex:- virtual threads |
| 2020 | as part of JEE 9 API namespace move from javax to jakarta in 2020-12-08  <https://en.wikipedia.org/wiki/Jakarta_EE>  ex:- all the javax anno like @PostConstruct have been migrated to sep package and separate jar “**jakarta.annotation-api**” “jakarta.annotation.PostConstruct”  In 2020, Oracle handed over stewardship of Java EE to an open-source foundation |
| 2018 | java 11 - Java 11 came out in September 2018. |
|  | java 10 – onwards jaxb api has been removed,  server has to provide these libraries here after , if developers need it we should manually add it |
| 2014 | java 8 |
| 2006 | java 6 |

## **List.of** vs  **Arrays.asList**

1. List.of can be best used when data set is less and unchanged, while Arrays.asList can be used best in case of large and dynamic data set.
2. List.of take very less overhead space because it has field-based implementation and consume less heap space, both in terms of fixed overhead and on a per-element basis. while Arrays.asList take more overhead space because while initialization it creates more objects in heap.
3. Collection returned by List.of is immutable and hence thread-safe while Collection returned by Arrays.asList is mutable and not thread safe. (Immutable collection instances generally consume much less memory than their mutable counterparts.)
4. List.of doesn't allow null elements while Arrays.asList allows null elements.

Other way of creating object

Class.forName(“com.nt.Employee”).newInstance();

1. In try block if I place a return statement will finally gets exceuted or not?-yes it will executed
2. By default src is on build path, if u want to check what is on classpath

then use system.getProperty(“java.class.path”);

"java.home","java.version","path.separator","user.home"

## Load JKS keystore file and cacert truststore file

//Load keystore means loading jks file

KeyStore keyStore = KeyStore.getInstance("JKS");

keyStore.load(new java.io.FileInputStream(keystorePath), keystorePassword.toCharArray());

// Retrieve the key

Key key = keystore.getKey(alias, keyPassword.toCharArray());

if (key instanceof PrivateKey) {

PrivateKey privateKey = (PrivateKey) key;

// Optional: Get certificate

Certificate cert = keystore.getCertificate(alias);

// Print details

System.out.println("Private Key: " + privateKey);

System.out.println("Certificate: " + cert);

// Load TrustStore means cacert file

KeyStore trustStore = KeyStore.getInstance("JKS");

trustStore.load(new java.io.FileInputStream(truststorePath), truststorePassword.toCharArray());

what is serialization

only serialized objects can be travelled in network ex:- any class implementing serializable interface

## jaxb unmarshal