**Obsah**

[1. Java basics 3](#_Toc13838039)

[1.1. JDK vs JRE vs JVM 3](#_Toc13838040)

[1.2. Java EE vs SE 3](#_Toc13838041)

[1.3. Collections 4](#_Toc13838042)

[1.4. Abstract class v Interface 4](#_Toc13838043)

[1.5. String – Buffer, Immutable... 4](#_Toc13838044)

[1.6. Keywords 4](#_Toc13838045)

[1.6.1. Static 4](#_Toc13838046)

[2. Technologies around Java 5](#_Toc13838047)

[2.1. Jackson and GSON 5](#_Toc13838048)

[2.2. Maven – scope 5](#_Toc13838049)

[2.3. Kafka 5](#_Toc13838050)

[2.4. Spring 5](#_Toc13838051)

[2.5. Play 5](#_Toc13838052)

[3. Java features 6](#_Toc13838053)

[3.1. Generic programming 6](#_Toc13838054)

[3.2. Java 8 6](#_Toc13838055)

[3.2.1. forEach method 6](#_Toc13838056)

[3.2.2. Nashorn JavaScript Engine 6](#_Toc13838057)

[3.2.3. Optional 6](#_Toc13838058)

[3.2.4. Default and static methods 6](#_Toc13838059)

[3.2.5. Java Stream API for Bulk Data Operations on Collections 6](#_Toc13838060)

[3.2.6. Functional Interfaces and Lambda Expressions 7](#_Toc13838061)

[3.3. Others 7](#_Toc13838062)

[3.3.1. Callback 7](#_Toc13838063)

[3.3.2. Future 7](#_Toc13838064)

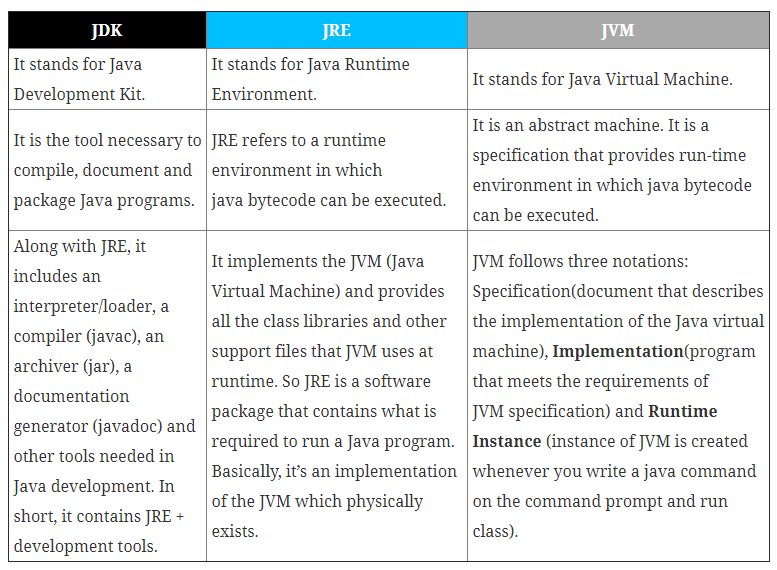
[3.3.3. Promise 7](#_Toc13838065)

[4. REST API (Representational State Transfer) 8](#_Toc13838066)

[4.1. What is 9](#_Toc13838067)

[4.2. Service-oriented architecture (SOA) 9](#_Toc13838068)

1. Java basics
   1. JDK vs JRE vs JVM



* 1. Java EE vs SE

Java SE = Standard Edition. This is the core Java programming platform. It contains all of the libraries and APIs that any Java programmer should learn (java.lang, java.io, java.math, java.net, java.util, etc...).

Java EE = Enterprise Edition. From Wikipedia:

The Java platform (Enterprise Edition) differs from the Java Standard Edition Platform (Java SE) in that it adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.

In other words, if your application demands a very large scale, distributed system, then you should consider using Java EE. Built on top of Java SE, it provides libraries for database access (JDBC, JPA), remote method invocation (RMI), messaging (JMS), web services, XML processing, and defines standard APIs for Enterprise JavaBeans, servlets, portlets, Java Server Pages, etc...

* 1. Collections
  2. Abstract class v Interface
  3. String – Buffer, Immutable...
  4. Keywords
     1. Static

1. Technologies around Java
   1. Jackson and GSON
   2. Maven – scope
   3. Kafka
   4. Spring
   5. Play
2. Java features
   1. Generic programming
   2. Java 8
      1. forEach method

Java provides a new method forEach() to iterate the elements. It is defined in Iterable and Stream interfaces.

It is a default method defined in the Iterable interface. Collection classes which extends Iterable interface can use forEach() method to iterate elements.

This method takes a single parameter which is a functional interface. So, you can pass lambda expression as an argument.

* + 1. Nashorn JavaScript Engine

Nashorn is a JavaScript **engine**. It is used to execute JavaScript code dynamically at JVM (Java Virtual Machine). Java provides a command-line tool jjs which is used to execute JavaScript code.

You can execute JavaScript code by two ways:

1. Using jjs command-line tool, and
2. By embedding into Java source code.
   * 1. Optional

Java introduced a new class Optional in Java 8. It is a public final class which is used to deal with NullPointerException in Java application. We must import java.util package to use this class. It provides methods to check the presence of value for particular variable.

* + 1. Default and static methods

We can use default and static keyword to create interfaces with method implementation.

* + 1. Java Stream API for Bulk Data Operations on Collections

A new java.util.stream has been added in Java 8 to perform filter/map/reduce like operations with the collection. Stream API will allow sequential as well as parallel execution. This is one of the best feature for me because I work a lot with Collections and usually with Big Data, we need to filter out them based on some conditions.

* + 1. Functional Interfaces and Lambda Expressions

@FunctionalInterface annotation. Functional interfaces are new concept introduced in Java 8. An interface with exactly one abstract method becomes Functional Interface. We don’t need to use @FunctionalInterface annotation to mark an interface as Functional Interface. @FunctionalInterface annotation is a facility to avoid accidental addition of abstract methods in the functional interfaces.

So lambda expressions are means to create anonymous classes of functional interfaces easily. There are no runtime benefits of using lambda expressions, so I will use it cautiously because I don’t mind writing few extra lines of code.

(n) -> n\*n

Arrow operator ‚->‘, The left side specifies the parameters required by the expression, which could also be empty if no parameters are required.

The right side is the lambda body which specifies the actions of the lambda expression. It might be helpful to think about this operator as “becomes”. For example, “n becomes n\*n”, or “n becomes n squared”.

* 1. Others
     1. Callback

A callback function is a function you provide to another piece of code, allowing it to be called by that code. A callback is a function that is to be executed after another function has finished executing — hence the name ‘call back’.

In JavaScript, functions are objects. Because of this, functions can take functions as arguments, and can be returned by other functions. Functions that do this are called higher-order functions. Any function that is passed as an argument is called a callback function.

* + 1. Future

Simply put, the Future class represents a future result of an asynchronous computation – a result that will eventually appear in the Future after the processing is complete.

* + 1. Promise

The Promise object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value.

1. REST API (Representational State Transfer)

* Representational State Transfer (REST) is a software architectural style that defines a **set of constraints to be used for creating Web services.**
* Web services that conform to the REST architectural style, called **RESTful Web services** (RWS), **provide interoperability between computer systems on the Internet.**
* RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations.
* Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations.

"Web resources" were first defined on the World Wide Web as documents or files identified by their URLs. However, today they have a much more generic and abstract definition that encompasses every thing or entity that can be identified, named, addressed, or handled, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the stored resource, and the response can provide hypertext links to other related resources or collections of resources. When HTTP is used, as is most common, the operations (HTTP methods) available are:

1. GET
2. HEAD - The HTTP HEAD method requests the headers that are returned if the specified resource would be requested with an HTTP GET method. Such a request can be done before deciding to download a large resource to save bandwidth, for example.
3. POST
4. PUT
5. PATCH - HTTP PATCH requests are to make partial update on a resource. If you see PUT requests also modify a resource entity so to make more clear – PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you’re replacing a resource in its entirety.
6. DELETE
7. CONNECT
8. OPTIONS
9. TRACE.

**Idempotent methods**: The term idempotent is used more comprehensively to describe an operation that will produce the same results if executed once or multiple times. GET, HEAD, PUT and DELETE are declared idempotent methods

By using a stateless protocol and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running.

* 1. What is
* Resource-oriented architecture (ROA)
* Resource-oriented computing (ROC)
* Web-oriented architecture (WOA)
* Atomicity, Consistenct, isolation, durability (ACID)
  1. Service-oriented architecture (SOA)

Service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service-oriented architecture are independent of vendors, products and technologies.

**Service:** is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online. A service has four properties according to one of many definitions of SOA:

* It logically represents a business activity with a specified outcome.
* It is self-contained.
* It is a black box for its consumers.
* It may consist of other underlying services.[3]

Different services can be used in conjunction to provide the functionality of a large software application, a principle SOA shares with modular programming. Service-oriented architecture integrates distributed, separately-maintained and -deployed software components. It is enabled by technologies and standards that facilitate components' communication and cooperation over a network, especially over an IP network.Frameworks.