The Java™ Tutorials

Trail: Bonus
Lesson: Preparation for Java Programmer Language Certification

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

Java SE 8 Upgrade Exam

This page maps sections in the Java Tutorials to topics covered in the Upgrade to Java SE 8 OCP (Oracle Certified Professional) (Java SE 6 and all prior versions) and Upgrade Java SE 7 to Java SE 8 OCP exams. These exams are associated with the Oracle Certified Professional, Java SE 8 Programmer certificate.

Upgrade to Java SE 8 OCP (Java SE 6 and all prior versions)

The topics covered in this exam are:

1. Language Enhancements
2. Concurrency
3. Localization
4. Java File I/O (NIO.2)
5. Lambda
6. Java Collections
7. Java Streams

Section 1: Language Enhancements

Item 1: Develop code that uses `String` objects in the `switch` statement, binary literals, and numeric literals, including underscores in literals

- The switch Statement
- Primitive Data Types
- Primitive Data Types

Item 2: Develop code that uses `try`-with-resources statements, including using classes that implement the `AutoCloseable` interface

- The try-with-resources Statement

Item 3: Develop code that handles multiple `Exception` types in a single `catch` block

- The catch Blocks

Item 4: Use static and default methods of an interface including inheritance rules for a default method

- Default Methods

Section 2: Concurrency

Item 1: Use classes from the `java.util.concurrent` package including `CyclicBarrier` and `CopyOnWriteArrayList` with a focus on the advantages over and differences from the traditional `java.util` collections

Item 2: Use `Lock`, `ReadWriteLock`, and `ReentrantLock` classes in the `java.util.concurrent.locks` and `java.util.concurrent.atomic` packages to support lock-free thread-safe programming on single variables

- Atomic Variables

Item 3: Use `Executor`, `ExecutorService`, `Executors`, `Callable`, and `Future` to execute tasks using thread pools

- Executor Interfaces

Item 4: Use the parallel Fork/Join Framework

- Fork/Join

Section 3: Localization

Item 1: Describe the advantages of localizing an application and developing code that defines, reads, and sets the locale with a `Locale` object

- Introduction
- Creating a Locale

Item 2: Build a resource bundle for a locale and call a resource bundle from an application

- Isolating Locale-Specific Data

Item 3: Create and manage date- and time-based events by using `LocalDate`, `LocalTime`, `LocalDateTime`, `Instant`, `Period`, and `Duration`, including a combination of date and time in a single object

- Date Classes
- Date and Time Classes
- Instant Class
- Period and Duration

Item 4: Format dates, numbers, and currency values for localization with the `NumberFormat` and `DateFormat` classes, including number and date format patterns

- Numbers and Currencies
- Dates and Times

Item 5: Work with dates and times across time zones and manage changes resulting from daylight savings

- Time Zone and Offset Classes

Section 4: Java File I/O (NIO.2)

Item 1: Operate on file and directory paths by using the `java.nio.Path` class

- Path Operations

Item 2: Check, delete, copy, or move a file or directory by using the `java.nio.Files` class

- Checking a File or Directory
- Deleting a File or Directory
- Copying a File or Directory
- Moving a File or Directory

Item 3: Recursively access a directory tree by using the `DirectoryStream` and `FileVisitor` interfaces

- Creating and Reading Directories
- Walking the File Tree

Item 4: Find a file by using the PathMatcher interface, and use Java SE 8 I/O improvements, including `Files.find`, `Files.walk`, and `Files.lines methods`

- Finding Files
- Walking the File Tree

Item 5: Observe the changes in a directory by using the `WatchService` interface

- Watching a Directory for Changes

Section 5: Lambda

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Define and write functional interfaces and describe the interfaces of the `java.util.function` package

Item 2: Describe a lambda expression; refactor the code that uses an anonymous inner class to use a lambda expression; describe type inference and target typing

Item 3: Develop code that uses the built-in interfaces included in the `java.util.function` package, such as `Function`, `Consumer`, `Supplier`, `UnaryOperator`, `Predicate`, and `Optional` APIs, including the primitive and binary variations of the interfaces

Item 4: Develop code that uses a method reference, including refactoring a lambda expression to a method reference

Section 6: Java Collections

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Develop code that uses diamond with generic declarations

- The Diamond

Item 2: Develop code that iterates a collection, filters a collection, and sorts a collection by using lambda expressions

Item 3: Search for data by using methods, such as `findFirst`, `findAny`, `anyMatch`, `allMatch`, and `noneMatch`

Item 4: Perform calculations on Java streams by using `count`, `max`, `min`, `average`, and `sum` methods and save results to a collection by using the `collect` method and `Collector` class, including the `averagingDouble`, `groupingBy`, `joining`, and `partitioningBy` methods

Item 5: Develop code that uses Java SE 8 collection improvements, including the `Collection.removeIf`, `List.replaceAll`, `Map.computeIfAbsent`, and `Map.computeIfPresent` methods

Item 6: Develop code that uses the `merge`, `flatMap`, and `map` methods on Java streams

Section 7: Java Streams

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Describe the `Stream` interface and pipelines; create a stream by using the `Arrays.stream` and `IntStream.range` methods; identify the lambda operations that are lazy

Item 2: Develop code that uses parallel streams, including decomposition operation and reduction operation in streams

Upgrade Java SE 7 to Java SE 8 OCP Programmer

The topics covered in this exam are:

1. Lambda Expressions
2. Using Built-in Lambda Types
3. Java Collections and Streams with Lambdas
4. Collection Operations with Lambda
5. Parallel Streams
6. Lambda Cookbook
7. Method Enhancements
8. Use Java SE 8 Date/Time API

Section 1: Lambda Expressions

Item 1: Describe and develop code that uses Java inner classes, including nested class, static class, local class, and anonymous classes

- Nested Classes
- Local Classes
- Anonymous Classes
- When to Use Nested Classes, Local Classes, Anonymous Classes, and Lambda Expressions

Item 2: Describe and write functional interfaces

Lambda Expressions

Item 3: Describe a lambda expression; refactor the code that uses an anonymous inner class to use a lambda expression; describe type inference and target typing

Lambda Expressions

Section 2: Using Built-in Lambda Types

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Describe the interfaces of the java.util.function package

Item 2: Develop code that uses the Function interface

Item 3: Develop code that uses the Consumer interface

Item 4: Develop code that uses the Supplier interface

Item 5: Develop code that uses the UnaryOperator interface

Item 6: Develop code that uses the Predicate interface

Item 7: Develop code that uses the primitive and binary variations of the base interfaces of the java.util.function package

Item 8: Develop code that uses a method reference, including refactoring a lambda expression to a method reference

Section 3: Java Collections and Streams with Lambdas

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Develop code that iterates a collection by using the forEach() method and method chaining

Item 2: Describe the Stream interface and pipelines

Item 3: Filter a collection by using lambda expressions

Item 4: Identify the operations, on stream, that are lazy

Section 4: Collection Operations with Lambda

The sections Lambda Expressions and Aggregate Operations cover some of the following items:

Item 1: Develop code to extract data from an object by using the map() method

Item 2: Search for data by using methods such as findFirst(), findAny(), anyMatch(), allMatch(), and noneMatch()

Item 3: Describe the unique characteristics of the Optional class

Item 4: Perform calculations by using Java Stream methods, such as count(), max(), min(), average(), and sum()

Item 5: Sort a collection by using lambda expressions

Item 6: Develop code that uses the Stream.collect() method and Collectors class methods, such as averagingDouble(), groupingBy(), joining(), and partitioningBy()

Section 5: Parallel Streams

Item 1: Develop code that uses parallel streams

Parallelism

Item 2: Implement decomposition and reduction in streams

Reduction

Section 6: Lambda Cookbook

Item 1: Develop code that uses Java SE 8 collection improvements, including Collection.removeIf, List.replaceAll, Map.computeIfAbsent, and Map.computeIfPresent methods

Item 2: Develop code that uses Java SE 8 I/O improvements, including Files.find, Files.walk, and Files.lines methods

Item 3: Use flatMap() methods in the Stream API

Item 4: Develop code that creates a stream by using the Arrays.stream() and IntStream.range() methods

Section 7: Method Enhancements

Item 1: Add static methods to interfaces

Static Methods

Item 2: Define and use a default method of an interface and describe the inheritance rules for the default method

Default Methods Overriding and Hiding Methods

Section 8: Use Java SE 8 Date/Time API

Item 1: Create and manage date- and time-based events, including a combination of date and time in a single object, by using LocalDate, LocalTime, LocalDateTime, Instant, Period, and Duration

- Date Classes
- Date and Time Classes

Item 2: Work with dates and times across time zones and manage changes resulting from daylight savings, including Format date and times values

- Time Zone and Offset Classes

Item 3: Define, create, and manage date- and time-based events using Instant, Period, Duration, and TemporalUnit

- Instant Class
- The Temporal Package
- Period and Duration

Previous page: Java SE 8 Programmer II Exam
Next page: End of Trail