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 Programmer II Exam

This page maps sections in the Java Tutorials to topics covered in the Java SE 8 Programmer II exam. This exam is associated with the Oracle Certified Professional, Java SE 8 Programmer certificate. The topics covered in this exam are:

1. Java Class Design
2. Advanced Class Design
3. Generics and Collections
4. Lambda Built-In Functional Interfaces
5. Java Stream API
6. Exceptions and Assertions
7. Use Java SE 8 Date/Time API
8. Java I/O Fundamentals
9. Java File I/O (NIO.2)
10. Concurrency
11. Building Database Applications with JDBC
12. Localization

Section 1: Java Class Design

Item 1: Implement encapsulation.

- What Is an Object?

Item 2: Implement inheritance including visibility modifiers and composition.

- Inheritance
- Overriding and Hiding Methods

Item 3: Implement polymorphism.

- Polymorphism

Item 4: Override `hasCode`, `equals`, and `toString` methods from `Object` class.

- Object as a Superclass

Item 5: Create and use singleton classes and immutable classes.

- The Singleton Design Pattern
- A Strategy for Defining Immutable Objects

Item 6: Develop code that uses the `static` keyword on initialize blocks, variables, methods, and classes.

- Understanding Class Members
- Initializing Fields
- Overriding and Hiding Methods
- Default Methods

Section 2: Advanced Class Design

Item 1: Develop code that uses abstract classes and methods.

- Abstract Methods and Classes

Item 2: Develop code that uses the `final`.

- Variables

- Understanding Class Members

Item 3: Create inner classes including static inner classes, local classes, nested classes, and anonymous innter classes.

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

Item 4: Use enumerated types including methods, and constructors in an `enum` type

- Enum Types
- Default Methods
- Enumerated Types

Item 5: Develop code that declares, implements and/or extends interfaces and use the `@Override` annotation.

- Predefined Annotation Types

Item 6: Create and use lambda expressions.

- Lambda Expressions

Section 3: Generics and Collections

The Generics (Updated) lesson, the Collections trail and, in particular, the specified pages.

Item 1: Create and use a generic class.

- Generic Types

Item 2: Create and use `ArrayList`, `TreeSet`, `TreeMap`, and `ArrayDeque` objects.

- The List Interface
- The Set Interface
- The Map Interface
- The Deque Interface

Item 3: Use `java.util.Comparator` and `java.lang.Comparable` interfaces.

- Object Ordering

Item 4: Collections, streams, and filters.

- Aggregate Operations

Item 5: Iterate using `forEach` methods of `Streams` and `List`.

- The Collection Interface
- Aggregate Operations

Item 6: Describe the `Stream` interface and the `Stream` pipeline.

- Aggregate Operations

Item 7: Filter a collection by using lambda expressions.

- The Collection Interface
- Aggregate Operations

Item 8: Use method references with streams.

- Method References

Section 4: Lambda Built-In Functional Interfaces

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

Item 1: Use the built-in interfaces included in the `java.util.function` package such as `Predicate`, `Consumer`, `Function`, and `Supplier`.

Item 2: Develop code that uses primitive versions of functional interfaces.

Item 3: Develop code that uses binary versions of functional interfaces.

Item 4: Develop code that uses the `UnaryOperator` interface.

Section 5: Java Stream API

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

Item 1: Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method.

Item 2: Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch.

Item 3: Develop code that uses the Optional class.

Item 4: Develop code that uses Stream data methods and calculation methods.

Item 5: Sort a collection using Stream API.

Item 6: Save results to a collection using the collect method and group/partition data using the Collectors class.

Item 7: Use flatMap() methods in the Stream API.

Section 6: Exceptions and Assertions

Item 1: Use `try-catch` and `throws` statements.

- Specifying the Exceptions Thrown by a Method
- How to Throw Exceptions

Item 2: Use `catch`, `multi-catch`, and `finally` clauses.

- Catching and Handling Exceptions
- The try Block
- The catch Blocks
- The finally Block
- Putting It All Together

Item 3: Use autoclose resources with a `try`-with-resources statement.

- The try-with-resources Statement

Item 4: Create custom exceptions and autocloseable resources.

- Creating Exception Classes

Item 5: Test invariants by using assertions.

- Questions and Exercises: Classes (assertion example)

Section 7: Use Java SE 8 Date/Time API

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

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

Item 2: Work with dates and times across timezones and manage changes resulting from daylight savings including Format date and times values.

- Time Zone and Offset Classes

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

- Instant Class
- Period and Duration
- The Temporal Package

Section 8: Java I/O Fundamentals

Item 1: Read and write data from the console.

The I/O Streams lesson and, in particular, the following pages:

- Byte Streams
- I/O from the Command Line

Item 2: Use `BufferedReader`, `BufferedWriter`, `File`, `FileReader`, `FileWriter`, `FileInputStream`, `FileOutputStream`, `ObjectOutputStream`, `ObjectInputStream`, and `PrintWriter` in the `java.io package`.

The File I/O (Featuring NIO.2) lesson, and in particular, the following pages:

- Reading, Writing, and Creating Files

- Creating and Reading Directories
- Random Access Files

Section 9: Java File I/O (NIO.2)

Item 1: Use the `Path` interface to operate on file and directory paths.

- What Is a Path? (And Other File System Facts)
- Path Operations

Item 2: Use the `Files` class to check, read, delete, copy, move, and manage metadata a file or directory.

- File Operations
- Checking a File or Directory
- Deleting a File or Directory
- Copying a File or Directory
- Moving a File or Directory
- Managing Metadata (File and File Store Attributes)
- Walking the File Tree
- Finding Files
- What is a Glob?
- Watching a Directory for Changes

Item 3: Use Stream API with NIO.2.

Section 3: Object-Oriented Design Principles

The Java Tutorials do not cover Design Patterns topics. The following references cover design patterns using the Java programming language:

- *Head First Design Patterns* by Elizabeth Freeman, et al.
- *Java Design Pattern Essentials* by Tony Bevis

Item 1: Write code that declares, implements and/or extends interfaces.

- Defining an Interface
- Interfaces
- Implementing an Interface

Item 2: Choose between interface inheritance and class inheritance.

Item 3: Develop code that implements "is-a" and/or "has-a" relationships.

Item 4: Apply object composition principles.

Item 5: Design a class using the Singleton design pattern.

Item 6: Write code to implement the DAO pattern.

Item 7: Design and create objects using a factory, and use factories from the API.

Section 5: String Processing

Item 1: Search, parse and build strings.

- Strings
- Converting Between Numbers and Strings
- Comparing Strings and Portions of Strings
- Manipulating Characters in a String

Item 2: Search, parse, and replace strings by using regular expressions.

- Methods of the Pattern Class
- Methods of the Matcher Class

Item 3: Use string formatting.

- Strings
- Formatting Numeric Print Output

Section 10: Concurrency

Item 1: Create worker threads using `Runnable`, `Callable` and use an `ExecutorService` to concurrently execute tasks.

- Executors
- Executor Interfaces
- Thread Pools

Item 2: Identify potential threading problems among deadlock, starvation, livelock, and race conditions.

- Memory Consistency Errors
- Deadlock

Item 3: Use synchronized keyword and java.util.concurrent.atomic package to control the order of thread execution.

- Atomic Variables

Item 4: Use java.util.concurrent collections and classes including CyclicBarrier and CopyOnWriteArrayList.

- Concurrent Collections

Item 5: Use parallel Fork/Join Framework.

- Fork/Join

Item 6: Use parallel Streams including reduction, decomposition, merging processes, pipelines and performance.

Section 11: Building Database Applicatons with JDBC

Item 1: Describe the interfaces that make up the core of the JDBC API including the `Driver`, `Connection`, `Statement`, and `ResultSet` interfaces and their relationship to provider implementations.

- JDBC Basics: Getting Started

Item 2: Identify the components required to connect to a database using the DriverManager class including the JDBC URL.

- Establishing a Connection
- Connecting with DataSource Objects

Item 3: Submit queries and read results from the database including creating statements, returning result sets, iterating through the results, and properly closing result sets, statements, and connections.

- Processing SQL Statements with JDBC
- Using Transactions
- Using RowSet Objects
- Using JdbcRowSet Objects

Section 12: Localization

Item 1: Read and set the locale by using the `Locale` object..

- Setting the Locale
- Creating a Locale

Item 2: Create and read a `Properties` file.

- About the ResourceBundle Class
- Backing a ResourceBundle with Properties Files

Item 3: Build a resource bundle for each locale and load a resource bundle in an application.

- Customizing Resource Bundle Loading

Previous page: Java SE 8 Programmer I Exam
Next page: Java SE 8 Upgrade Exam