OCP – Preparation

Inhoud

[Chapter 1 (Welcome to Java) 2](#_Toc107517198)

[Major Components 2](#_Toc107517199)

[Identifying Benefits of Java 2](#_Toc107517200)

[Understanding the Java Class Structure 3](#_Toc107517201)

[Understanding package declarations and imports 4](#_Toc107517202)

[Chapter 2 (Java Building Blocks) 4](#_Toc107517203)

[Chapter 3 (Operators) 4](#_Toc107517204)

[Chapter 4 (Making Decisions) 4](#_Toc107517205)

[Chapter 5 (Core Java APIs) 4](#_Toc107517206)

[Chapter 6 (Lambdas and Functional Interfaces) 4](#_Toc107517207)

[Chapter 7 (Methods and Encapsulation) 4](#_Toc107517208)

[Chapter 8 (Class design) 4](#_Toc107517209)

[Chapter 9 (Advanced Class Design) 4](#_Toc107517210)

[Chapter 10 (Exceptions) 4](#_Toc107517211)

[Chapter 11 (Modules) 4](#_Toc107517212)

[Chapter 12 (Java Fundamentals) 4](#_Toc107517213)

[Chapter 13 (Annotations) 4](#_Toc107517214)

[Chapter 14 (Generics and Collections) 4](#_Toc107517215)

[Chapter 15 (Functional Programming) 4](#_Toc107517216)

[Chapter 16 (Exceptions, Assertions and Localizations) 4](#_Toc107517217)

[Chapter 17 (Modular Applications) 5](#_Toc107517218)

[Chapter 18 (Concurrency) 5](#_Toc107517219)

[Chapter 19 (I/O) 5](#_Toc107517220)

[Chapter 20 (NIO.2) 5](#_Toc107517221)

[Chapter 21 (JDBC) 5](#_Toc107517222)

[Chapter 22 (Security) 5](#_Toc107517223)

# Chapter 1 (Welcome to Java)

## Major Components

* JDK – (Java Development Kit) Contains the minimum software you need to do Java Development.
* Compiler – (javac) Converts .java files to .class files, and the launcher java, which creates the virtual machine and executes the program.
* JDK also contains other tools including the archiver (jar) command, which can package files together, and the API documentation (Javadoc) command for generating documentation.
* Javac program generates instructions in a special format that the java command can run called bytecode, then java launches the Java Virtual Machine (JVM) before running the code.
* The JVM knows how to run your .class files and knows how to run bytecode.
* Java comes with a large suite of **application programming interfaces (APIs)** that you can use from the start, such as a StringBuilder class to create large String and a method in Collections

## Identifying Benefits of Java

**Object Oriented –** Java is an object-oriented language. Means all code is defined in classes, and most of those classes can be instantiated into objects.

**Encapsulation –** Java supports access modifiers to protect data from unintended access and modification.

**Platform Independent –** Java is an interpreted language that gets compiled to bytecode. A key benefit is that Java code gets compiled once rather than needing to be recompiled for different operating systems. This is known as “write once, run everywhere.”

**Robust –** Java prevents memory leaks. Java manages memory on its own and does garbage collection automatically.

**Simple –** Java was intended to be simpler to understand than C++ removal of pointers and operator overloading.

**Secure –** Java code runs inside the JVM. This creates a sandbox that makes it hard for Java code to do evil things to the computer it is running on.

**Multithreaded –** Java is designed to allow multiple pieces of code to run at the same time. There are also many APIs to facilitate this task.

**Backward Compatibility –** The Java Language architects pay careful attention to making sure old programs will work with later versions of Java. By using the Deprecation technique, they accomplish this where code is flagged to indicate it shouldn’t be used.

## Understanding the Java Class Structure

**Object** – is a runtime instance of a class in memory, also referred to as an instance since it represents a single representation of the class.

**State of the program** – all the various objects of all the different classes.

**Reference** – is a variable that points to an object.

**Fields and Methods**

Java classes have two primary elements: **methods** – often called functions or procedures in other languages, and **fields**, more generally known as variables. Together they are called the members of the class.

* **Variables hold** the state of the program,
* and **methods operate** on that state.

If a change is important to remember, a variable store that change.

**Classes vs. Files**

Most of the time Java classes are defined in their own .java file. Generally, they are public meaning any other class can call the class. Interestingly, Java does not require that the class be public. You can put two or more classes in a single .java file BUT only one class may be public, and it must have the same name as the file name else it will not compile!

**Writing a main() method**

* each file can contain only one public class but can contain multiple classes.
* The filename must match the class name, including case, and have a .java extension.
* The main() method signature is:
  + public static void main(String[] args){}
* java file is compiled with fist with command javac filename.java
* secondly its then run with the command java filename (omitting the .class JDK knows what file to use)
* if you want to pass parameters as arguments then you can add it after the filename example:
  + java filename 1 2 3 4
    - (parameters 1 2 3 4 is 4 different STRING INPUT PARAMETERS
    - These parameters are always of type String []

**Running a program in one line**

* Adding the .java to the filename using the java command gives us this function since Java 11 ONLY for single .java file programs. When you have more than one .java file you still need to use the javac filename.java -> java filename process to launch your program.
* This helps when creating small one file applications ideal for testing small single file programs.

## Understanding package declarations and imports

# Chapter 2 (Java Building Blocks)

# Chapter 3 (Operators)

# Chapter 4 (Making Decisions)

# Chapter 5 (Core Java APIs)

# Chapter 6 (Lambdas and Functional Interfaces)

# Chapter 7 (Methods and Encapsulation)

# Chapter 8 (Class design)

# Chapter 9 (Advanced Class Design)

# Chapter 10 (Exceptions)

# Chapter 11 (Modules)

# Chapter 12 (Java Fundamentals)

# Chapter 13 (Annotations)

# Chapter 14 (Generics and Collections)

# Chapter 15 (Functional Programming)

# Chapter 16 (Exceptions, Assertions and Localizations)

# Chapter 17 (Modular Applications)

# Chapter 18 (Concurrency)

# Chapter 19 (I/O)

# Chapter 20 (NIO.2)

# Chapter 21 (JDBC)

# Chapter 22 (Security)