DeepL
Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

Object-Oriented Programming

8. Lecture
LS 2025/2026
Author: Juraj Petrík

IO

- FileWriter
- write()
- close()
- Everything can throw IOException!

- File - represents the name and path of a file (folder)

IO - Buffer

- BufferedWriter
- Why?
  - The disk is slower than memory
  - We work with "chunks"
- flush() – when we want to write before filling the buffer

io, nio, nio2

- With NIO.2, we usually talk about two packages:
  - Java.nio.file
    - Path interface
    - Paths class
    - Files class
  - Java.nio.file.attribute – metadata

- Try-with-resources:
  - implements Autocloseable
  - multiple IO resources closed in opposite order

Documentation (JavaDoc)

- Imagine your program as a library that someone wants to use
- Good API documentation is critical for (re)using code
- Javadoc are "special comments": /** **/
- Standardized, HTML like Java Standard Library
- Forces you to think
- E.g. "before" a class, interface, method, attribute
- https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

JavaDoc what (not) to do

- <code> for keywords and names
- Overuse:
  - In-line links ({@link})
- Use third person
- This instead of the when referring to an object from the class we are describing
- The best names are self-descriptive, so don't "document" them unnecessarily

JavaDoc Tags (order matters)

- @author (classes and interfaces only, required)
- @version (classes and interfaces only, required. See footnote 1)
- @param (methods and constructors only)
- @return (methods only)
- @exception (@throws is a synonym added in Javadoc 1.2)
- @see
- @since
- @serial (or @serialField or @serialData)
- @deprecated (see How and When To Deprecate APIs)

JavaDoc tags

- {@link package.class#member label}
- @see:
  - @see Java Documentation
  - @see "This method performs some function."
  - @see String#toLowerCase() convertToLowerCase
- {@inheritDoc}

JavaDoc Generation

- CLI – javadoc –d docs Class.java
- IDE
- Maven – mvn javadoc:javadoc
- Gradle – gradle javadoc

- Custom tags

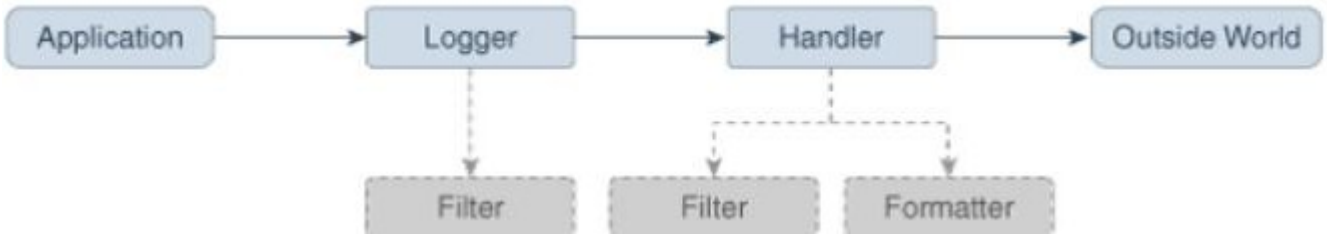
Logging

- Why?
  - Debugging assistance
  - Audit logs
  - Monitoring
  - Observability
- Multiple levels (LogLevel)
  - E.g.: Fatal, Error, Warn, Info, Debug, Trace, (Off)
- How?
  - Timestamp
  - Class name
  - Log Level
  - Thread ID

- java.util.logging
- Log4j
- logback
- Log4j2
- SLF4J - The Simple Logging **Facade** for Java

- Standard combo:
  - SLF4J + logback
  - SLF4J + log4j2

Java.util.logging



SLF4J

- Simple Logging **Facade** for Java
- Abstraction over various logging frameworks – plug and play

- import org.slf4j.Logger;
- import org.slf4j.LoggerFactory;

Some things to consider

- if (logger.isEnabled(Level.INFO))

  logger.info(String.format("The result is %d.", superExpensiveMethod()));
- Hot Path
- Asynchronous logging
- Sensitive data
- Include Stack Trace when logging exceptions
- Machine-processable logs

log4j2

- Appenders (handlers)
  - log4j2.xml

Asynchronous logging

- Higher throughput
- Lower latency

- -- Error handling
- -- Beware of mutable messages

Quiz time