howest
hogeschool

Object Oriented Architectures and Secure Development

01-03 Logging

Arne Debou

Mattias De Wael

Frédéric Vlummens

What is logging?

- Keeping track of errors, status changes, … in your application

- In the most simple form: writing towards a text file

- Various logging frameworks/mechanisms exist

- Today: java.util.logging → built-in logging mechanism

howest
hogeschool

Note: do not use System.out.println() for logging purposes!

- System.out.println() is used to create command line (terminal) applications

- Using (or rather: abusing) System.out.println() for logging is a major code smell

- https://rules.sonarsource.com/java/RSPEC-106

- We should use actual loggers!

howest
hogeschool

Creating an instance of java.util.logging.Logger

```java
package be.howest.ti.shop;

public class Shop {

    private static final Logger LOGGER = Logger.getLogger(Shop.class.getName());

    ...

}
```

howest
hogeschool

Creating an instance of java.util.logging.Logger

```java
package be.howest.ti.shop;

public class Shop {

    private static final Logger LOGGER = Logger.getLogger("be.howest.ti.shop.Shop");

    ...

}
```

**howest**
hogeschool

Why we should use java.util.logging.Logger

- We decided to use java.util.logging.Logger.

- Other logging frameworks exist, this is just a choice for this course.

- Why static? Because there is no need to keep an instance per object, one suffices.

- Why final? There is no need to change the logger at runtime...

- Why give it the (full)name of the class? This will allow us to configure the log output per class.

- You can pick any name you like, but this is a common practice...

How to use a logger?
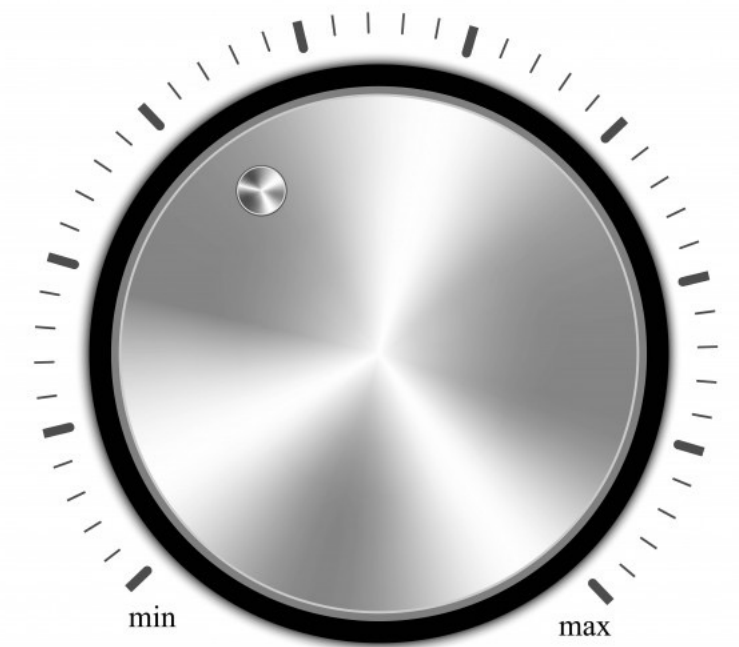
- We can use a logger to log a message at a certain level.

- You can configure your logger to change its *"volume level"*.

- The levels are:
  - SEVERE (highest value)
  - WARNING
  - INFO
  - CONFIG
  - FINE
  - FINER
  - FINEST (lowest value)
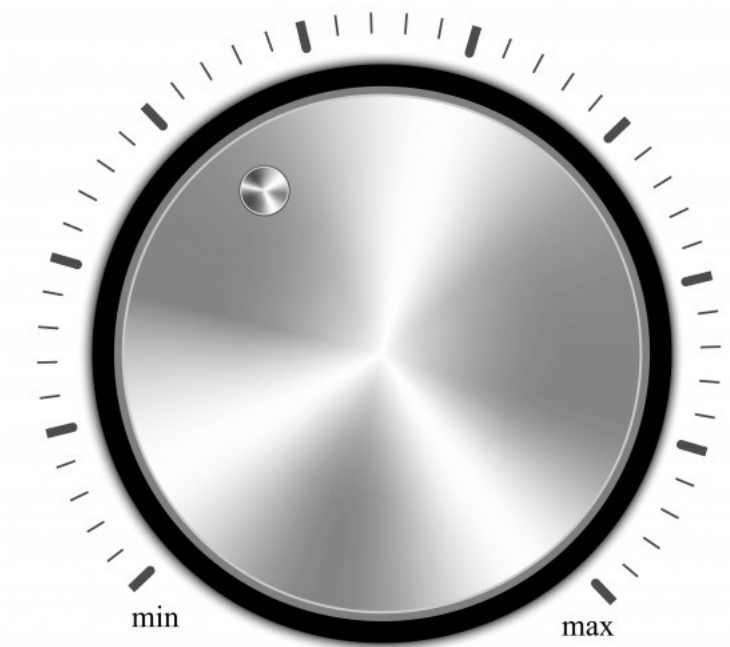
howest
hogeschool

How to use a logger?

- We can use a logger to log a message at a certain level.

- You can configure your logger to change its *"volume level"*.

- The levels are:
    - SEVERE is a message level indicating a serious failure.
    - WARNING is a message level indicating a potential problem.
    - INFO is a message level for informational messages.
    - CONFIG is a message level for static configuration messages.
    - FINE is a message level providing tracing information.
    - FINER indicates a fairly detailed tracing message.
    - FINEST indicates a highly detailed tracing message.

howest
hogeschool

How to use a logger?

- We can use a logger to log a message at a certain level.

- You can configure your logger to change its volume level.

- *LOGGER*.log(Level.*FINER*, "Product created");

- This exists in many flavours …

Simple Message

LOGGER.log(Level.*SEVERE*, "Simple message");

Sep 24, 2020 7:47:59 PM be.howest.ti.shop.Program main

SEVERE: Simple message

Parameterised Message (built-in)

LOGGER.log(Level.*SEVERE*, "Parameterised message: {0} {1} {2}",

　　new Object[]{1, "two", 3}

);

Replaces the {x}'s in the message string by the elements in the array.

Sep 24, 2020 7:47:59 PM be.howest.ti.shop.Program main

SEVERE: Parameterised message: 1 two 3

howest
hogeschool

Delayed Message (with String Supplier)

LOGGER.log(Level.SEVERE, () -> String.format("Delayed message: %s", new Date()));

If you feel more comfortable with String.format, you should use this form.

Just like with assertThrows, the () -> {} delays the computation until (and if) needed.

Sep 24, 2020 7:47:59 PM be.howest.ti.shop.Program main
SEVERE: Delayed message: Thu Sep 24 19:47:59 CEST 2020

howest
hogeschool

Exception Message

```java
public String toString() {
    try {
        return String.format("Product #%d: %s @ %.2f", this.getCode(),
                this.getName(), this.getPrice());
    } catch (IllegalStateException ex) {
        LOGGER.log(Level.SEVERE, "Unable to determine valid price", ex);
        return String.format("Product #%d: %s", this.getCode(), this.getName());
    }
}
```

Log an exception and its stack trace. sep. 21, 2021 3:08:24 P.M. be.howest.ti.shop.Product toString

SEVERE: Unable to determine valid price   java.lang.IllegalStateException: Product has invalid price

at be.howest.ti.shop.Product.getPrice(Product.java:32)

at be.howest.ti.shop.Product.toString(Product.java:50)

howest
hogeschool

Secure Coding Guidelines

- **Guideline 1-1 / DOS-1:** Beware of activities that may use disproportionate resources
    - *Detailed logging of unusual behaviour may result in excessive output to log files.*

- **Guideline 2-2 / CONFIDENTIAL-2:** Do not log highly sensitive information
    - *E.g., do not log paswords or social security numbers or the like, …*

https://www.oracle.com/java/technologies/javase/seccodeguide.html

howest
hogeschool