# 1 Java 8 – 17 Features

# 1.1 Lambdas, Streams

# **Cycles Simplification**

How to do make simpler?

```
List<LocalDate> myReports = new ArrayList<>();
for (Report r : reports) {
    if (r.isActive()) {
        if (r.getAuthor().equals(me)) {
            myReports.add(r.getDueTo());
        }
    }
}
Collections.sort(myReports);
```

Nothing wrong, business as usual. Can we do it better?

# **Cycles Simplification**

Is this better/more readable?

```
List<LocalDate> myReports = reports.stream()
   .filter((Report r) -> r.isActive())
   .filter((Report r) -> r.getAuthor().equals(me))
   .map((Report r) -> r.getDueTo())
   .sorted()
   .toList();
```

...and we can continue...

#### **Cycles Simplification**

We can remove types...

```
List<LocalDate> myReports = reports.stream()
    .filter(r -> r.isActive())
    .filter(r -> r.getAuthor().equals(me))
    .map(r -> r.getDueTo())
    .sorted()
    .toList();
```

...and we can continue...

# **Cycles Simplification**

We can use method reference...

```
public static boolean isMyReport(Report r) {
    return r.equals(me);
}
List<LocalDate> myReports = reports.stream()
    .filter(Report::isActive)
    .filter(TestFunctional::isMyReport)
    .map(Report::getDueTo)
    .sorted()
    .toList();
```

...Let's compare it on the next slide!

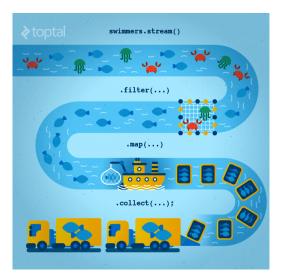


Figure 1: Stream processing visualization. Source: https://www.toptal.com/ java/why-you-need-to-upgrade-to-java-8-already

## **Cycles Simplification**

#### Before

```
List<LocalDate> myReports = new ArrayList<>();
for (Report r : reports) {
    if (r.isActive()) {
        if (r.getAuthor().equals(me)) {
            myReports.add(r.getDueTo());
    } }
Collections.sort(myReports);
```

#### After

```
List<LocalDate> myReports = reports.stream()
    .filter(Report::isActive)
    .filter(TestFunctional::isMyReport)
    .map(Report::getDueTo)
    .sorted()
    .toList();
```

#### Stream

# Lambda for Multithreaded Application

So far it was just a syntax sugar. BUT! How easily can you write multithreaded apps?

```
List<LocalDate> myReports = reports.stream()
    .parallel() // run on multiple threads!
    .filter(RemoteVerification::isValid) // calls outsite service
    .toList();
```

# 1.2 Optional

#### Optional

#### Before

```
Report r = reports.get(0);
Band header = r.getHeaderBand();
if(header!=null) {
    title = header.getTitle();
    if(title==null) {
        title = "Default Title";
    }
}
```

# After

```
String title = Optional.of(reports.get(0))
    .map(Report::getHeaderBand)
    .map(Band::getTitle)
    .orElse("Default Title");
```

# Log4j

# Before – annoying

```
if(log.isDebugEnabled()) {
    log.debug(prepareDataForLog());
}
```

#### Simple – useless overhead if not used

log.debug(prepareDataForLog());

#### Functional – simple and effective

log.debug(() -> prepareDataForLog());

# **Couple of Usefulness**

#### Switch Expression

```
switch (day) {
   case MONDAY, FRIDAY, SUNDAY -> 6;
   case TUESDAY -> 7;
   case THURSDAY, SATURDAY -> 8;
   case WEDNESDAY -> 9;
   default -> throw new IllegalStateException("Invalid day: " + day);
}
```

#### **Records aka Lombok**

record Point(int x, int y) {}

#### ... and Many Others

- Collectors.teeing,
- String.repeat(n) or Stream<String> lines()
- text block
- if(abj instanceof String str) {}
- JMH
- NullPointerException: a.b.c()
- shebang
- Vector API (Incubator)
- Foreign Function and Memory API (Incubator)

# 2 Agile World

#### Waterfall, Model V

- What's wrong with waterfall, model V (e.g. detailed planning before programming)? Everything!
  - Detailed analysis becomes useless immediately after programming starts many assumptions are wrong.
  - Detailed long-time planning is crazy can you say, what you will do on September  $21^{st}$  2021 in the morning? And afternoon?
  - Users tend to change their minds when they see the first version.
  - Programming takes long time and situation changes.
  - Studies have shown that in over 80 % of the investigated and failed software projects, the usage of the Waterfall methodology was one of the key factors of failure. https://www.scrum-institute.org/What\_Makes\_Waterfall\_Fail\_in\_Many\_Ways.php

### Agile Style of Work

- Principles (see agilemanifesto.org)
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- Pair programming
- SCRUM or Canban
- Cooperation is much more important than individual success.
- Frequent and regular increments! Often are shared with customers.

# 2.1 Continuous Integration & Deployment

## **Continuous Integration**

- After every commit, build is verified including unit tests
- At least once a day, the whole product is deployed including functional tests
- UI tests are done frequently (can take hours)
- ... all automated.
- Quick detection of errors, cheaper fixes, fewer integration issues.

# CI Tools

### Version Control

• Git, others: CSV, Subversion, Bazaar, Mercurial, Bitkeeper, RTC...

### **CI** Servers

- Jenkins
  - Open-source, easy to setup,
  - Highly configurable, lots of plugins.

### • TeamCity

- Free for 3 agents and 20 build configurations,
- Developed by JetBrains,
- More suitable for enterprises beast.
- Gitlab CI
- Today's servers concentrate on whole process including deployment to cloud.

Arcobe <sup>®</sup> Dashboards • Issues Mo JOPA Technical Debt Coverage Duplicati	nsures Rules Quality Prefi					Log In Q = September 16, 2017 4:16 PM Version
ustom		_				Time changes
Lines Of Code 30,307 Java	Files 522 Directories Lines 106 51,819	Functions 3,453 Classe Battements Accessors 560 12,870 362	Debt 30d	lssues 1,289	Blocker     Critical     Major     Minor     Inflo	0 20 824 385 67
Unit Tests Coverage 56.7% Line Coverage Condition Coverage 55.9% 59.3%		Unit Test Success           100.0%           Failures         Errors           0         1,845           8         1:14 min	March 22, 2017 0.9.5	Lines of code: 28,750		
Duplications 1.3% Lines Blocks Files 649 38 23			UNRESOLVED ISS	July October UES	2017	April July
Documentation 39.2% Nublic API Pub. Undoc. API 1,856 1,129		Comments 12.5% Comment Lines 4.327	Total October 1, 2017 – C September 1, 2017 – Au July 1, 2017 – Au July 1, 2017 – July 3	- September 30, 2017 gust 31, 2017		1.3k 0 23 0
JOPA JOPA - implementation	0		June 1, 2017 – June	30, 2017		38
Critical Open Not assigned 1	ethod. 🚥	ayoparmooevennywanagemipi jarva				2 years ago ▼ L516 S ▼ ▼
JOPA JOPA - Implementation Ether log or rethrow this exception. Critical O Open Martin Ledvi		si jopa/model/metamodel/ManagedClassProcessor.java				10 months ago = L87 S T =

# **Static Code Analysis**

- Analyze code structure or flows, don't run it.
- Full-featured IDEs contain some sort of SCA.
- Checkstyle checks just formatting.
- **FindBugs** simple and pretty fast check, can find adding to String inside cycle, impossible equals, bad null handling...
- **Sonarqube** server-side analysis, long, discovers data flow from database to servlet (e.g. finds XSS)

# Sonarqube

### 2.2 12 Factor App

## 12 Factor App 1/2

- https://12factor.net/
- I. Codebase
  - One codebase tracked in revision control, many deploys
- II. Dependencies
  - Explicitly declare and isolate dependencies
- III. Config
  - Store config in the environment

- IV. Backing services
  - Treat backing services as attached resources
- V. Build, release, run
  - Strictly separate build and run stages
- VI. Processes
  - Execute the app as one or more stateless processes

# 12 Factor App 2/2

- VII. Port binding
  - Export services via port binding
- VIII. Concurrency
  - Scale out via the process model
- IX. Disposability
  - Maximize robustness with fast startup and graceful shutdown
- X. Dev/prod parity
  - Keep development, staging, and production as similar as possible
- XI. Logs
  - Treat logs as event streams
- XII. Admin processes
  - Run admin/management tasks as one-off processes

# **3** Application Monitoring and Administration

# 3.1 JMX

# Java Management Extensions (JMX)

- Allow management of resources in an application,
- Standard part of the Java platform,
- Resources represented by Managed Beans (MBeans), registered in an MBean server,
- Accessible via JMX connectors.

### Managed Beans

- Operations (MBean methods), through which the application can be managed,
- Attributes (getters/setters) for information/configuration.

Overview Memory Threads Classes VM Summary MBe	ans			
▶ Catalina	MBeanInfo			
MImplementation	Name	Value		
Users		Trade,		
▼ bean	Info:			
▼ <sup>®</sup> INBASAdminBean	ObjectName	ch.qos.logback.classic:Name=default.Type=ch.qos.logback.classic.jmx.JMXConfigurator		
✓ Operations	ClassName	ch.qos.logback.classic.jmx.JMXConfigurator		
invalidateCaches	Description	Information on the management interface of the MBean		
Notifications	Constructor-0:			
<ul> <li>ch.gos.logback.classic</li> </ul>	Name	ch.qos.logback.classic.jmx.JMXConfigurator		
▼ default	Description	Public constructor of the MBean		
<ul> <li>Ch.qos.logback.classic.jmx.JMXConfigurator</li> </ul>	Parameter-0-0:			
▼ Attributes	Name	p1		
Statuses	Description			
LoggerList	Type	ch.gos.logback.classic.LoggerContext		
▼ Operations	Parameter-0-1:			
reloadDefaultConfiguration	Name	p2		
reloadByFileName	Description			
reloadByURL	Type	javax.management.MBeanServer		
getLoggerEffectiveLevel	Parameter-0-2:			
getLoggerLevel	Name	p3		
setLoggerLevel	Description			
com.sun.management     iava.lang	Туре	javax.management.ObjectName		
java.nio				
java.util.logging				
	Descriptor			
	Name	Value		
	Info:			
	immutableInfo	true		
	interfaceClassName	ch.gos.logback.classic.jmx.JMXConfiguratorMBean		
	mxbean	false		

# **Application Management via JMX**

- Connect to application with *JConsole*,
- Locate the desired MBean,
  - Invoke managed operations,
  - View/configure attributes,
- MBean server set up in Spring @EnableMBeanExport.

# More Tools

### JDK

- jmap memory-related statistics about a VM, obsolete,
- jcmd send diagnostic commands to JVM, internally used by the GUI tools,
- jstat monitors JVM statistics, lots of options.
- Eclipse MAT advanced memory analyzer,
- Java Mission Control and Java Flight Recorder commercial JVM monitoring tools by Oracle,
- StageMonitor, MoSKito etc. open source alternatives.
- CA Wily very famous and very detailed monitoring of JavaEE

# 4 Database Versioning

# **Database Versioning**

- JPA provides a possibility to create missing tables
- ... useless when table is changed
- Libraries: Liquibase and Flyway
- A list of changes is recorded, keeps current database version
- Application keeps steps to upgrade from one version to the next
- The most reliable way
- Alternatives: direct upgrades from older version (leads to multiple ways hard testing), creating SQL scripts (customers tend to make mistakes during deployment, problematic error handling)
- Example of bad database upgrade: https://docs.gitlab.com/ee/update/ \#upgrade-paths
- Martin Fowler: Evolutionary Database Design

# **5** Production

# **Production Environments**

- As usual supported servers inside client's network (Payara, Glassfish, TomEE, WildFly, WebSphere)
- Hosted our servers in server houses
- Currently investigating Clouds, Docker
  - Problem with acceptance in banks
  - Cloud requires multitenancy application, e.g. there is a big risk of information leak, very rare
  - Docker seem a good choice, pack of all required software, needs just CPU, memory, disk space, TCP/IP ports.

#### What We Actually Use

- Versioning: git, gitlab
- CI: Jenkins, investigating Gitlab CI
- Code analyzis: Findbugs
- IDE: NetBeans :-), Idea (In fact, this doesn't matter.)
- Servers: Payara, TomEE, less Glassfish, WebSphere, WildFly
- Databases: PostgreSQL, MSSQL, Oracle
- Monitoring: JavaMelody
- OS: our systems Linux, clients often Windows, recently Docker

#### The End

# Thank You Petr Aubrecht petr@aubrecht.net

#### Resources

- R. Urma, M. Fusco and A. Mycroft: Java 8 in Action
- http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646. html
- https://martinfowler.com/articles/continuousIntegration.html
- https://www.martinfowler.com/articles/evodb.html
- http://docs.oracle.com/javase/tutorial/jmx/mbeans/index.html
- http://docs.oracle.com/javase/7/docs/technotes/guides/management/jconsole. html
- https://github.com/javamelody/javamelody/wiki