1 Java 8 – 21 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!

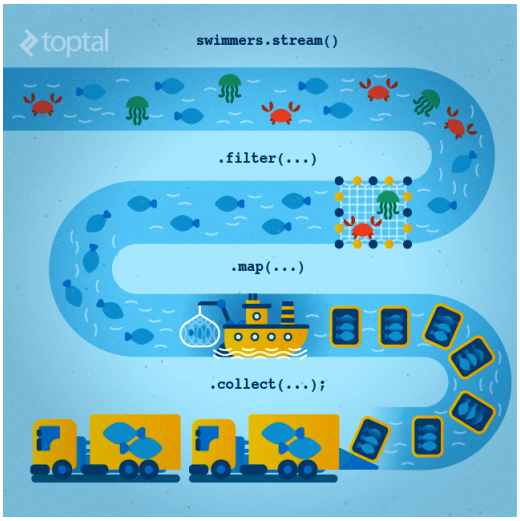


Figure 1: Stream processing visualization. Source: https://www.toptal.com/java/why-you-need-to-upgrade-to-java-8-already

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

```
title = reports.get(0).getTitle(); // I want this!

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)
      .map(reports -> 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

```
int value = 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) {}
```

Couple of Usefulness

Text Blocks

```
public static final String SAMPLE_ENTRY
      = """
      R 6 (#70c710)
      D 5 (#0dc571)
      L 2 (#5713f0)
      """;
```

Virtual Threads

```
Thread.startVirtualThread(() -> {
            // do asynch job
        });
```

...and Many Others

- `Collectors.teeing`,
- `String.repeat(n)` or `Stream<String> lines()`
- `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 October 21^{st} next year 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.
- ... e.g. you have a release several times a day – you need CI.

2.1 Continuous Integration & Deployment

Continuous Integration

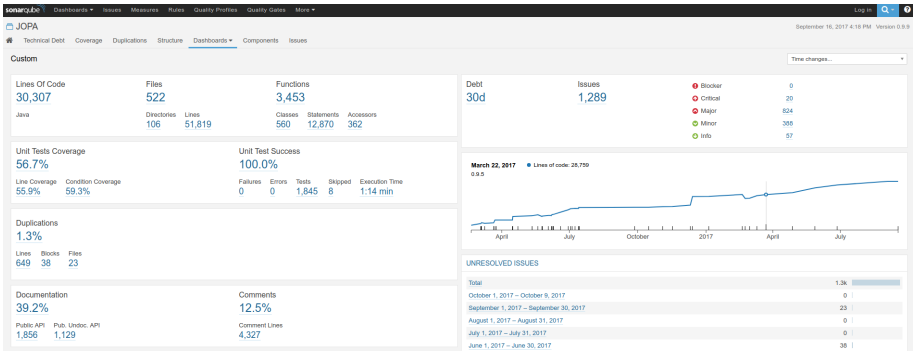
- After every commit, build is verified – including unit tests
- At least once a day, the whole product is deployed – including functional tests
- UI tests are done frequently (can take hours)
- ... all automated.
- Quick detection of errors, cheaper fixes, fewer integration issues.

CI Tools

Version Control

- **Git**, others: CSV, Subversion, Bazaar, Mercurial, Bitkeeper, RTC...
- Pro GIT (Chapter 1 Introduction, Chapter 2 Basics, Chapter 3 Branches)

CI Servers



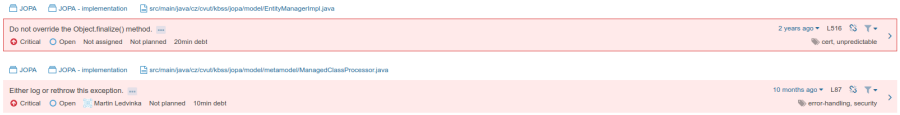
- **Jenkins**
  - Open-source, easy to setup,
  - Highly configurable, lots of plugins.
- **TeamCity**
  - Free for 3 agents and 20 build configurations,
  - Developed by JetBrains,
  - More suitable for enterprises – beast.
- **Github/Gitlab CI**
- Today's servers concentrate on whole process including deployment to cloud.

Static Code Analysis

- Analyze code structure or flows, don't run it.
- Full-featured IDEs contain some sort of SCA.
- **Checkstyle** – checks just formatting.
- **SpotBugs[1](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

[1] https://jenkins.payara.fish/view/All/job/Spotbugs-Payara6-Build/



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[2] 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

[2] db, messaging server

  - 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.

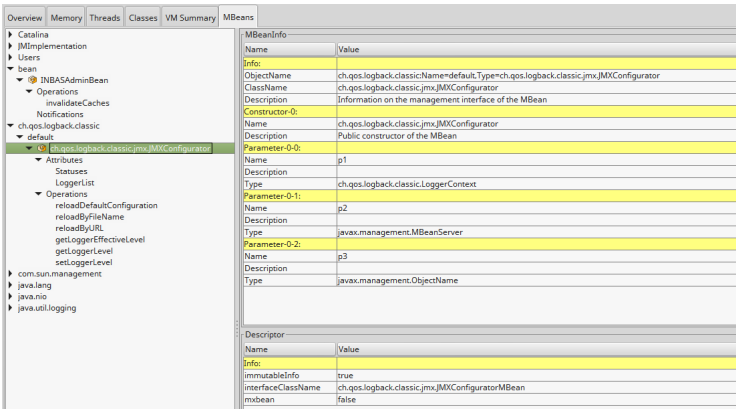
Application Management via JMX

- 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
- 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, github, gitlab
- CI: Jenkins, investigating Gitlab CI
- Code analyzis: Spotbugs
- 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