JAVA 9

the great modularity migration

an exploratory experiment into how many emojis
we can fit into one presentation

contents

- installing tools
- disclaimers
- class path hell/reflection
- module system
- new java services

- old java
  - classpath hell
  - reflection
  - lack of emojis 😭
- new java
  - module system
  - improved services
  - emojis 😍

disclaimers

- Oracle is charging money for support and security updates
- otherwise use the openJDK
- new garbage collector in V10 ♻
- JAVA 9 HAS EMOJI SUPPORT 💯🏆😍

classpath hell 😈

- when a class is initialised the JVM looks for a class name by going down a list of all classes
- uses the first hit
- becomes a problem with multiple versions of the same JAR; when similar class names are used

reflection 🔍

- you can use reflection to make any private matter… less private…
- AKA, everyone can use everything
- welcome to the wild west 🤠

```
public static void main(String[] args) throws ClassNotFoundException,
    Class c = Class.forName("AmazingClassName").getClass();
    Object o = c.newInstance();
    Method m = c.getDeclaredMethod( name: "amazingMethod");
    m.setAccessible(true);
    m.invoke(o, ...args: null);
}
```

modules 

- extra layer around a set of packages
- explicitly states what it imports/exports
- public is no longer application ‘global’ but module bound
- deals with the problems of reflection and class path hell

- modules deal in readability not visibility
- modules are defined in “module-info.java” file
- “exports” for PACKAGES
  - only public classes can be exported
- “requires” for external modules
- “requires transitive” forces import

```
module madness.modularity.migration {
    // exports
    exports madness.geocoder.output;
    exports madness.geocoder.input;
    exports madness.geocoder.process;

    //imports
    requires google.maps.services;
    requires java.sql;
}
```

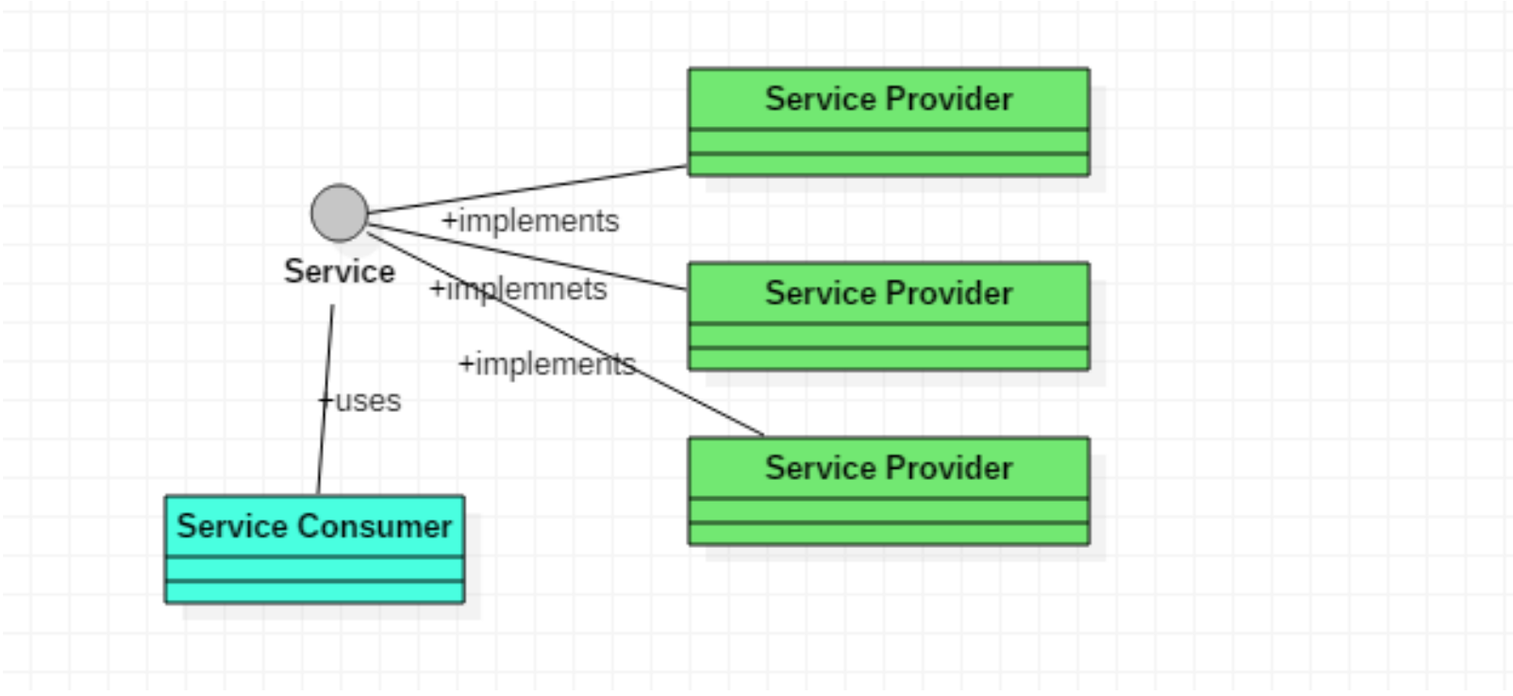
GeoCoder

1. fork and clone the git repo
2. implement class diagram
3. export/require the right modules
4. install the module to your local maven repository (mvn install) (good luck with that… ☘)
5. open user project and import the right modules
6. make the user project use the GeoCoder app to get coordinates for Fontys FHTenL

services

- services are a way to make your application “plug-and-play”
- you do not need an implementation of a service on compile time only at run time
- Without services you would need to export/import all implementations and users and recompile all stake holding JAR’s
- With services you only recompile the service providing JAR
- Java 9 synergy!

- service admin
- service provider
- service implementation
- service annotation



vehicle ORM

1. commit to your own git repo and first checkout the “transportorm” project
2. have a look around the internet for some examples
3. Good luck!
4. No but seriously ask us questions when you get stuck, we are/have the solution(s)