There must be (at least) 50 ways to extend

Odysseus

the event processing system

Loosely based on Paul Simon

https://en.wikipedia.org/wiki/Paul_Simon#/media/File:Paul_Simon_at_the_9-30_Club_(b).jpg

Marco Grawunder
https://uol.de/marco.grawunder

Bamberg, 1.10.2018

Remark

- This tutorial is work in progress ;-)
- I designed it the last three weeks (in every free minute)
- There might be errors and some things might be misleading
- There will be questions ;-)
  - You should ask ☺

Extending Odysseus

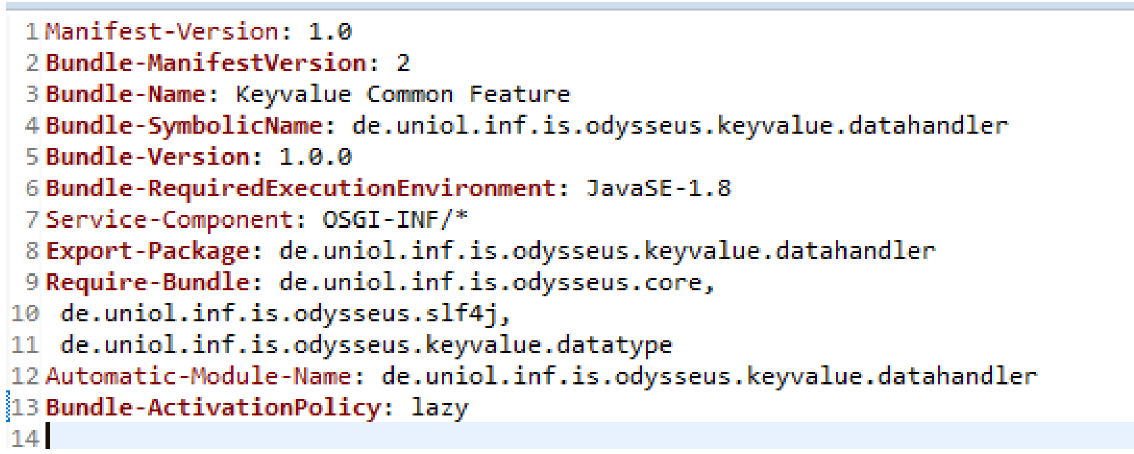
- Nearly everything in Odysseus is designed to be replaced or extended
- ... but some aspects are more challenging ;-)
- … here are the more typical ones ☺

Extensions

- Language extensions
  - Create a new language (not only for queries, could be a DSL for anything)
  - Create a new Odysseus Script Commands (#....)
  - Create a new Logical/PQL Operator
- Processing function extensions
  - Create new datatypes
  - Create new stream object types
    - Data Handler
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates
  - Create new aggregation functions
  - Create new operators
- Create new schedulers and scheduling strategies
- Create new meta data

Some OSGi/Eclipse-Basics

- Bundles
  - Aka plugin: An eclipse project
  - Each bundle has its own class loader
  - MANIFEST.MF: meta data for that bundle (name, version, imports, exports…)
  - In Odysseus: a module that encapsulates functions
- Fragment
  - A special bundle that will not exist allone but together with a host bundle
  - Same class loader as host bundle
  - Used to extend host bundle
  - We do not use this anymore, better approach is declarative services
- Declarative Services
  - OSGi way of dependency injection
  - Defined by so called components
  - Can provide functions by interfaces or use (bind/unbind) implementations by interfaces → examples later



```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Keyvalue Common Feature
Bundle-SymbolicName: de.uniol.inf.is.odysseus.keyvalue.datahandler
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: JavaSE-1.8
Service-Component: OSGI-INF/*
Export-Package: de.uniol.inf.is.odysseus.keyvalue.datahandler
Require-Bundle: de.uniol.inf.is.odysseus.core,
 de.uniol.inf.is.odysseus.slf4j,
 de.uniol.inf.is.odysseus.keyvalue.datatype
Automatic-Module-Name: de.uniol.inf.is.odysseus.keyvalue.datahandler
Bundle-ActivationPolicy: lazy

```

Some OSGi/Eclipse-Basics

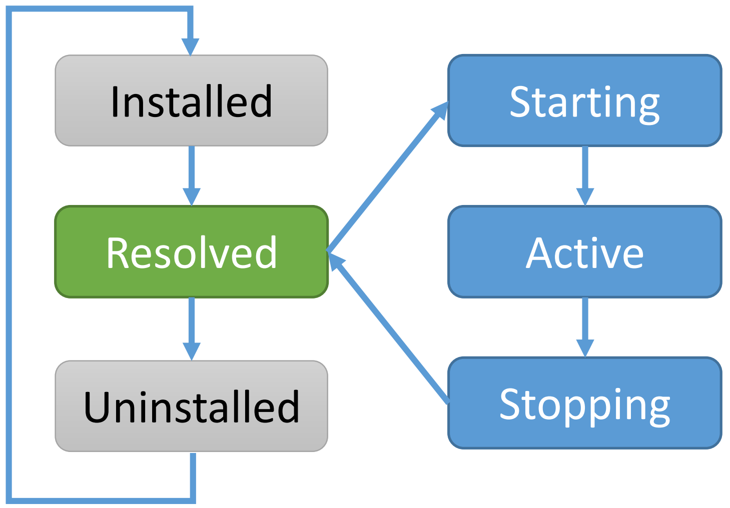
- Feature: a feature is a collection of bundles
  - Define a set of bundles, that belong together and builds some functionality (e.g. each wrapper has its own feature)
  - An update site provides features
  - A bundle can be part of many features
- Update-site:
  - A collection of features that can be installed in Odysseus
  - On the same way, as in Eclipse („Install new software“)
  - Via command on ther server
  - Via Odysseus-Script
- Product:
  - A product is a runnable software (with an application)
  - Can be defined by bundles or features
  - We only use features to define products

OSGi Life cycle

- Each bundle has a life cycle
- Installed: A bundle (with correct Metadata is installed)
- Resolved: All dependencies (MANIFEST) are found
- Uninstalled: removed from runtime

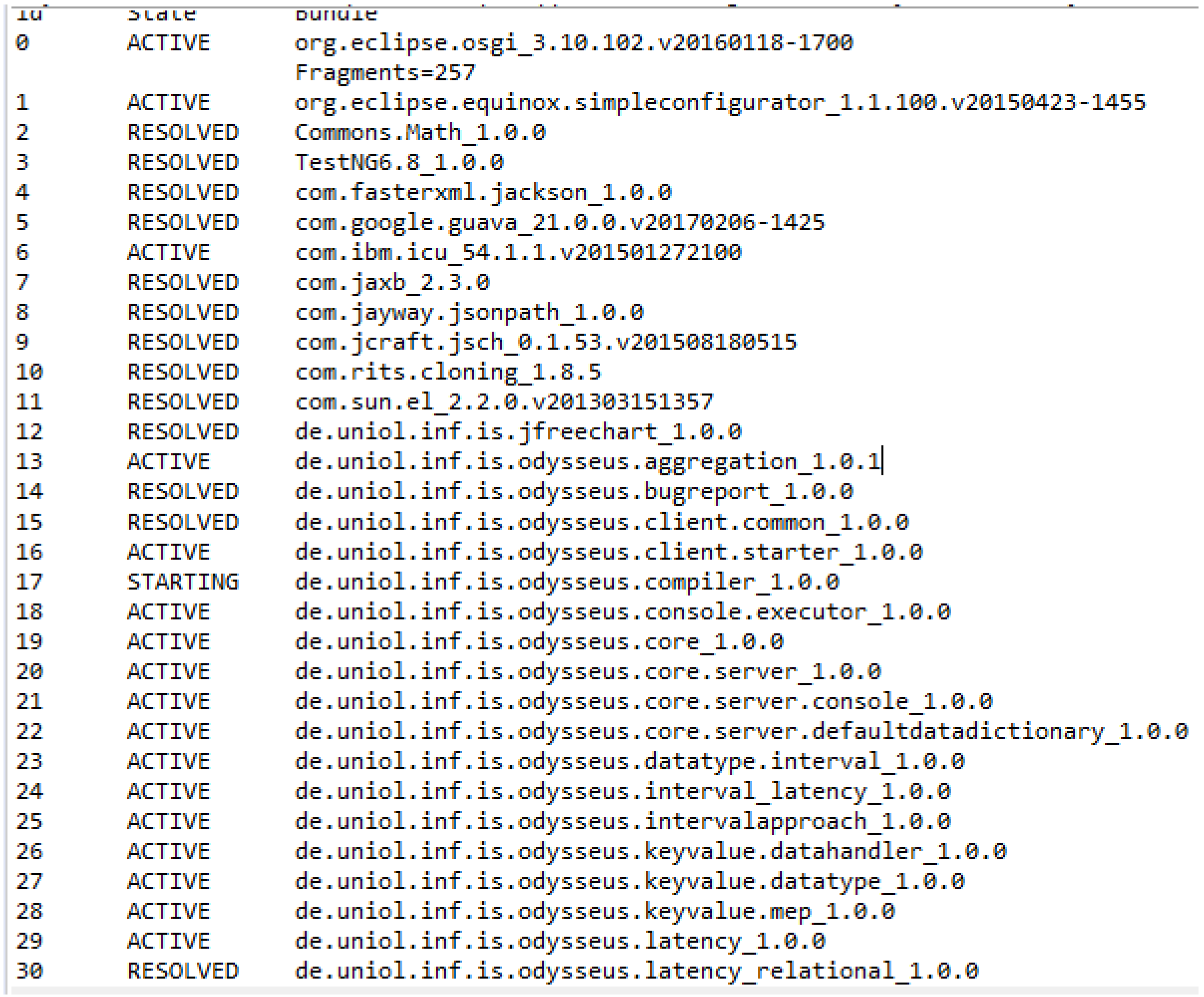
- Active: a bundle is activated
  - E.g. call of bundle activator
  - Remark: there is no need to start a bundle, if the bundle should only provide classes (as a library)

- Eclipse tries to resolve dependencies lazy, if this cannot be done, the bundle stays installed → bundle cannot be used



OSGi debugging

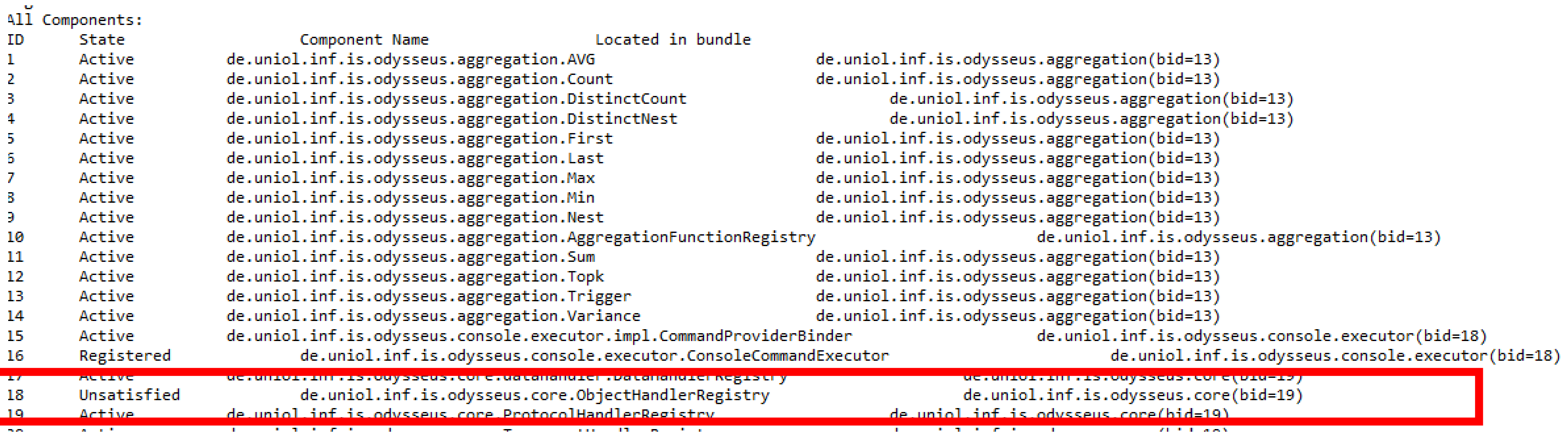
- When an application (product) gets started with `-console` (as in Odysseus always), there is a console available
- `ss` shows all currently available bundles and their current life cycle state
- Sometimes, there are problems because dependencies are missing (INSTALLED)
- Typical problem: The dependency defined in the MANIFEST.MF was not added to any feature
- `diag <id>` shows the missing dependencies
- Again: Resolved is no "problem" ;-)



```
Id      State       Bundle
0       ACTIVE      org.eclipse.osgi_3.10.102.v20160118-1700
                    Fragments=257
1       ACTIVE      org.eclipse.equinox.simpleconfigurator_1.1.100.v20150423-1455
2       RESOLVED    Commons.Math_1.0.0
3       RESOLVED    TestNG6.8_1.0.0
4       RESOLVED    com.fasterxml.jackson_1.0.0
5       RESOLVED    com.google.guava_21.0.0.v20170206-1425
6       ACTIVE      com.ibm.icu_54.1.1.v201501272100
7       RESOLVED    com.jaxb_2.3.0
8       RESOLVED    com.jayway.jsonpath_1.0.0
9       RESOLVED    com.jcraft.jsch_0.1.53.v201508180515
10      RESOLVED    com.rits.cloning_1.8.5
11      RESOLVED    com.sun.el_2.2.0.v201303151357
12      RESOLVED    de.uniol.inf.is.jfreechart_1.0.0
13      ACTIVE      de.uniol.inf.is.odysseus.aggregation_1.0.1
14      RESOLVED    de.uniol.inf.is.odysseus.bugreport_1.0.0
15      RESOLVED    de.uniol.inf.is.odysseus.client.common_1.0.0
16      ACTIVE      de.uniol.inf.is.odysseus.client.starter_1.0.0
17      STARTING    de.uniol.inf.is.odysseus.compiler_1.0.0
18      ACTIVE      de.uniol.inf.is.odysseus.console.executor_1.0.0
19      ACTIVE      de.uniol.inf.is.odysseus.core_1.0.0
20      ACTIVE      de.uniol.inf.is.odysseus.core.server_1.0.0
21      ACTIVE      de.uniol.inf.is.odysseus.core.server.console_1.0.0
22      ACTIVE      de.uniol.inf.is.odysseus.core.server.defaultdatadictionary_1.0.0
23      ACTIVE      de.uniol.inf.is.odysseus.datatype.interval_1.0.0
24      ACTIVE      de.uniol.inf.is.odysseus.interval_latency_1.0.0
25      ACTIVE      de.uniol.inf.is.odysseus.intervalapproach_1.0.0
26      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.datahandler_1.0.0
27      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.datatype_1.0.0
28      ACTIVE      de.uniol.inf.is.odysseus.keyvalue.mep_1.0.0
29      ACTIVE      de.uniol.inf.is.odysseus.latency_1.0.0
30      RESOLVED    de.uniol.inf.is.odysseus.latency_relational_1.0.0
```

OSGi debugging

- `ls`: shows all currently installed (declarative) services

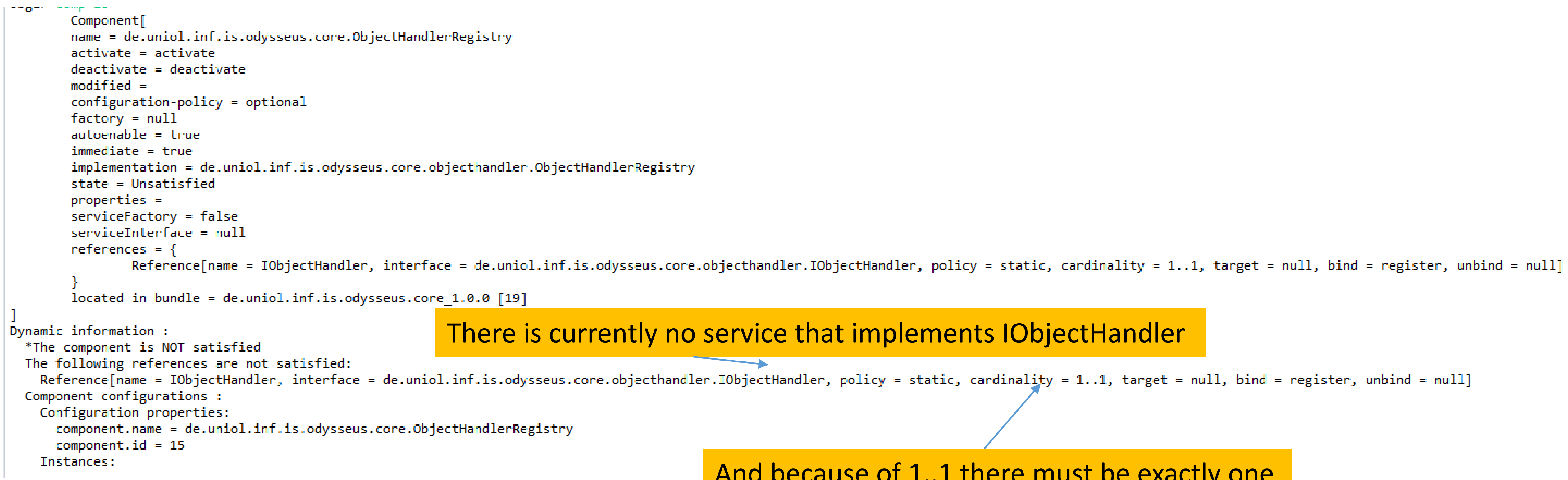


```
All Components:
ID      State           Component Name                  Located in bundle
1       Active          de.uniol.inf.is.odysseus.aggregation.AVG                 de.uniol.inf.is.odysseus.aggregation(bid=13)
2       Active          de.uniol.inf.is.odysseus.aggregation.Count               de.uniol.inf.is.odysseus.aggregation(bid=13)
3       Active          de.uniol.inf.is.odysseus.aggregation.DistinctCount               de.uniol.inf.is.odysseus.aggregation(bid=13)
4       Active          de.uniol.inf.is.odysseus.aggregation.DistinctNest                de.uniol.inf.is.odysseus.aggregation(bid=13)
5       Active          de.uniol.inf.is.odysseus.aggregation.First               de.uniol.inf.is.odysseus.aggregation(bid=13)
6       Active          de.uniol.inf.is.odysseus.aggregation.Last                de.uniol.inf.is.odysseus.aggregation(bid=13)
7       Active          de.uniol.inf.is.odysseus.aggregation.Max                 de.uniol.inf.is.odysseus.aggregation(bid=13)
8       Active          de.uniol.inf.is.odysseus.aggregation.Min                 de.uniol.inf.is.odysseus.aggregation(bid=13)
9       Active          de.uniol.inf.is.odysseus.aggregation.Nest                de.uniol.inf.is.odysseus.aggregation(bid=13)
10      Active          de.uniol.inf.is.odysseus.aggregation.AggregationFunctionRegistry                 de.uniol.inf.is.odysseus.aggregation(bid=13)
11      Active          de.uniol.inf.is.odysseus.aggregation.Sum                 de.uniol.inf.is.odysseus.aggregation(bid=13)
12      Active          de.uniol.inf.is.odysseus.aggregation.Topk                de.uniol.inf.is.odysseus.aggregation(bid=13)
13      Active          de.uniol.inf.is.odysseus.aggregation.Trigger             de.uniol.inf.is.odysseus.aggregation(bid=13)
14      Active          de.uniol.inf.is.odysseus.aggregation.Variance            de.uniol.inf.is.odysseus.aggregation(bid=13)
15      Active          de.uniol.inf.is.odysseus.console.executor.impl.CommandProviderBinder             de.uniol.inf.is.odysseus.console.executor(bid=18)
16      Registered              de.uniol.inf.is.odysseus.console.executor.ConsoleCommandExecutor                 de.uniol.inf.is.odysseus.console.executor(bid=18)
17      Active          de.uniol.inf.is.odysseus.core.datahandler.DataHandlerRegistry            de.uniol.inf.is.odysseus.core(bid=19)
18      Unsatisfied             de.uniol.inf.is.odysseus.core.ObjectHandlerRegistry              de.uniol.inf.is.odysseus.core(bid=19)
19      Active          de.uniol.inf.is.odysseus.core.ProtocolHandlerRegistry            de.uniol.inf.is.odysseus.core(bid=19)
```

- Unsatisfied: Some dependencies cannot be found
- Use comp <id> to determine missing dependencies

OSGi debugging for components (comp)

osgi> comp 18



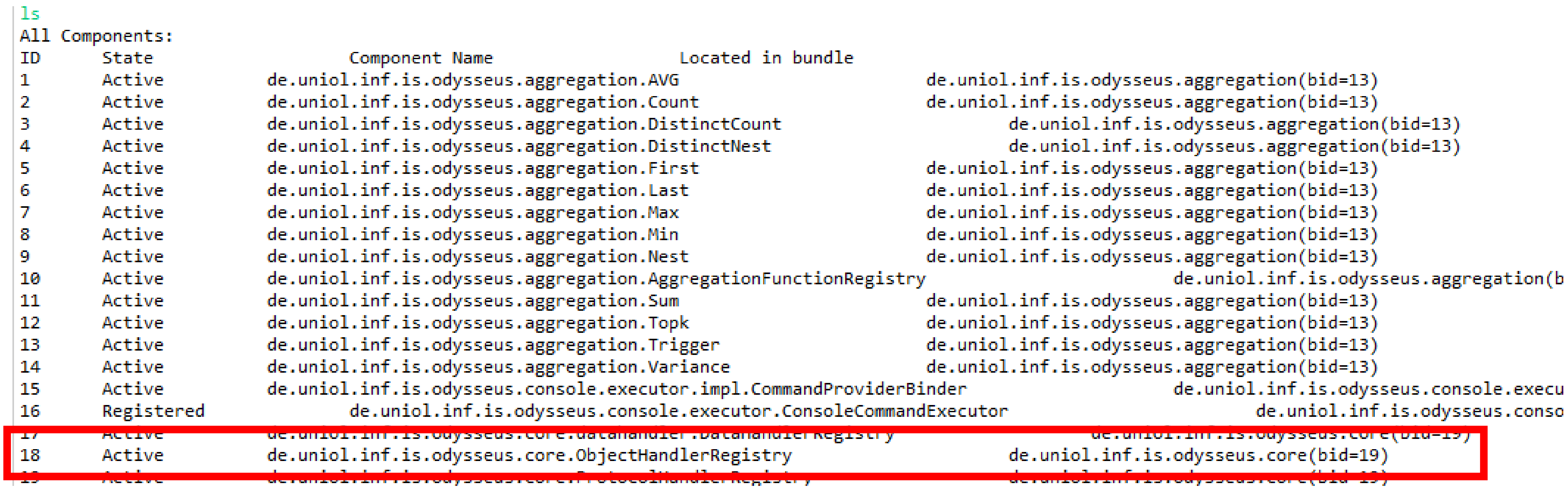
```
        Component[
        name = de.uniol.inf.is.odysseus.core.ObjectHandlerRegistry
        activate = activate
        deactivate = deactivate
        modified =
        configuration-policy = optional
        factory = null
        autoenable = true
        immediate = true
        implementation = de.uniol.inf.is.odysseus.core.objecthandler.ObjectHandlerRegistry
        state = Unsatisfied
        properties =
        serviceFactory = false
        serviceInterface = null
        references = {
                Reference[name = IObjectHandler, interface = de.uniol.inf.is.odysseus.core.objecthandler.IObjectHandler, policy = static, cardinality = 1..1, target = null, bind = register, unbind = null]
        }
        located in bundle = de.uniol.inf.is.odysseus.core_1.0.0 [19]
]
Dynamic information :
  *The component is NOT satisfied
  The following references are not satisfied:
    Reference[name = IObjectHandler, interface = de.uniol.inf.is.odysseus.core.objecthandler.IObjectHandler, policy = static, cardinality = 1..1, target = null, bind = register, unbind = null]
  Component configurations :
    Configuration properties:
      component.name = de.uniol.inf.is.odysseus.core.ObjectHandlerRegistry
      component.id = 15
    Instances:
```

There is currently no service that implements IObjectHandler

And because of 1..1 there must be exactly one

Problem here: missing required service, evaluated at runtime

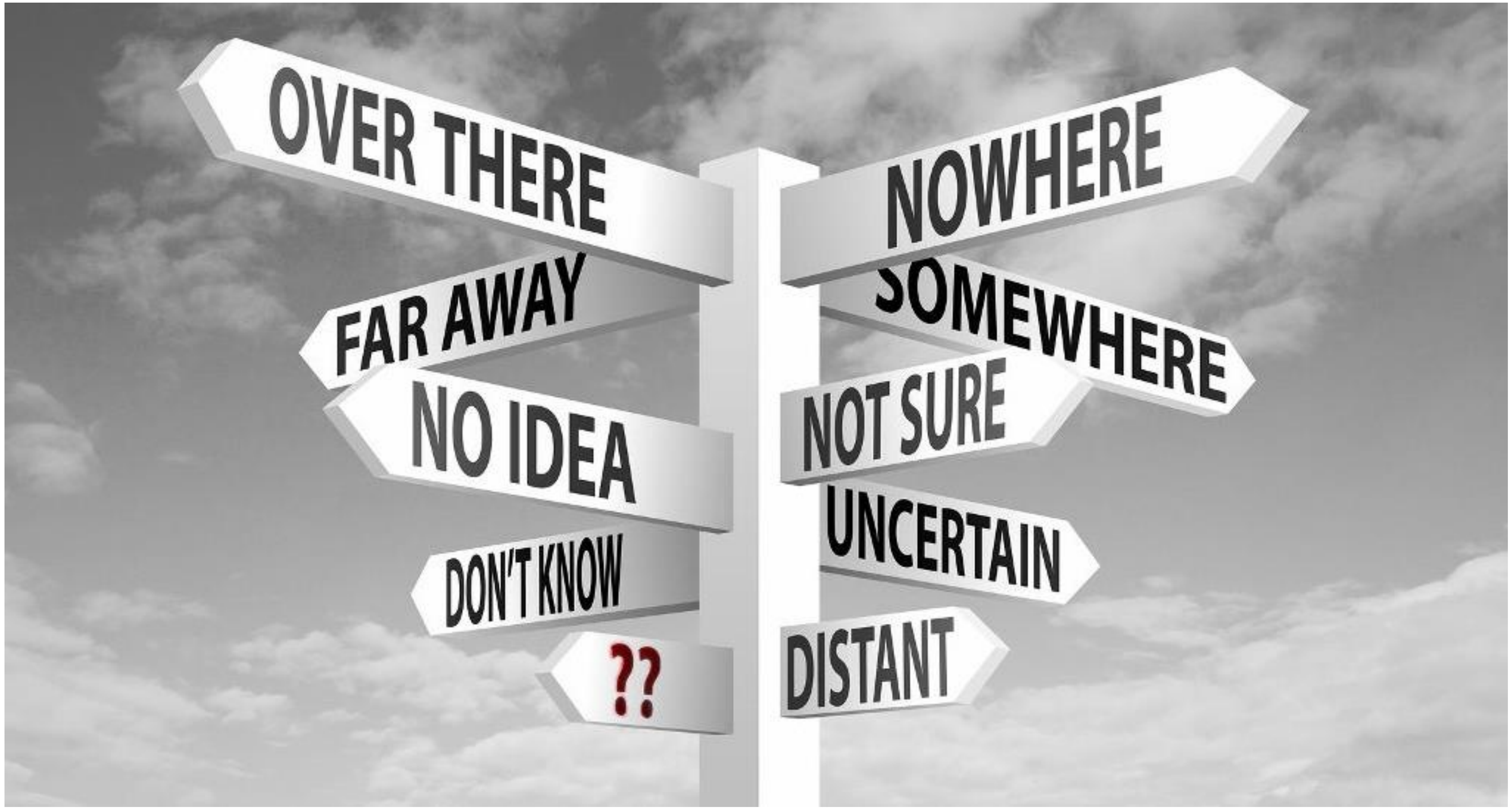
After fixing the problem



```
ls
All Components:
ID      State                   Component Name                          Located in bundle
1       Active          de.uniol.inf.is.odysseus.aggregation.AVG                                de.uniol.inf.is.odysseus.aggregation(bid=13)
2       Active          de.uniol.inf.is.odysseus.aggregation.Count                              de.uniol.inf.is.odysseus.aggregation(bid=13)
3       Active          de.uniol.inf.is.odysseus.aggregation.DistinctCount                              de.uniol.inf.is.odysseus.aggregation(bid=13)
4       Active          de.uniol.inf.is.odysseus.aggregation.DistinctNest                               de.uniol.inf.is.odysseus.aggregation(bid=13)
5       Active          de.uniol.inf.is.odysseus.aggregation.First                              de.uniol.inf.is.odysseus.aggregation(bid=13)
6       Active          de.uniol.inf.is.odysseus.aggregation.Last                               de.uniol.inf.is.odysseus.aggregation(bid=13)
7       Active          de.uniol.inf.is.odysseus.aggregation.Max                                de.uniol.inf.is.odysseus.aggregation(bid=13)
8       Active          de.uniol.inf.is.odysseus.aggregation.Min                                de.uniol.inf.is.odysseus.aggregation(bid=13)
9       Active          de.uniol.inf.is.odysseus.aggregation.Nest                               de.uniol.inf.is.odysseus.aggregation(bid=13)
10      Active          de.uniol.inf.is.odysseus.aggregation.AggregationFunctionRegistry                        de.uniol.inf.is.odysseus.aggregation(b
11      Active          de.uniol.inf.is.odysseus.aggregation.Sum                                de.uniol.inf.is.odysseus.aggregation(bid=13)
12      Active          de.uniol.inf.is.odysseus.aggregation.Topk                               de.uniol.inf.is.odysseus.aggregation(bid=13)
13      Active          de.uniol.inf.is.odysseus.aggregation.Trigger                            de.uniol.inf.is.odysseus.aggregation(bid=13)
14      Active          de.uniol.inf.is.odysseus.aggregation.Variance                           de.uniol.inf.is.odysseus.aggregation(bid=13)
15      Active          de.uniol.inf.is.odysseus.console.executor.impl.CommandProviderBinder                    de.uniol.inf.is.odysseus.console.execu
16      Registered              de.uniol.inf.is.odysseus.console.executor.ConsoleCommandExecutor                        de.uniol.inf.is.odysseus.conso
17      Active          de.uniol.inf.is.odysseus.core.datahandler.DataHandlerRegistry                   de.uniol.inf.is.odysseus.core(bid=19)
18      Active          de.uniol.inf.is.odysseus.core.ObjectHandlerRegistry                     de.uniol.inf.is.odysseus.core(bid=19)
19      Active          de.uniol.inf.is.odysseus.core.ProtocolHandlerRegistry                   de.uniol.inf.is.odysseus.core(bid=19)
```

Odysseus Development Model

- until now: Clone the whole Odysseus source code and get lost ;-)



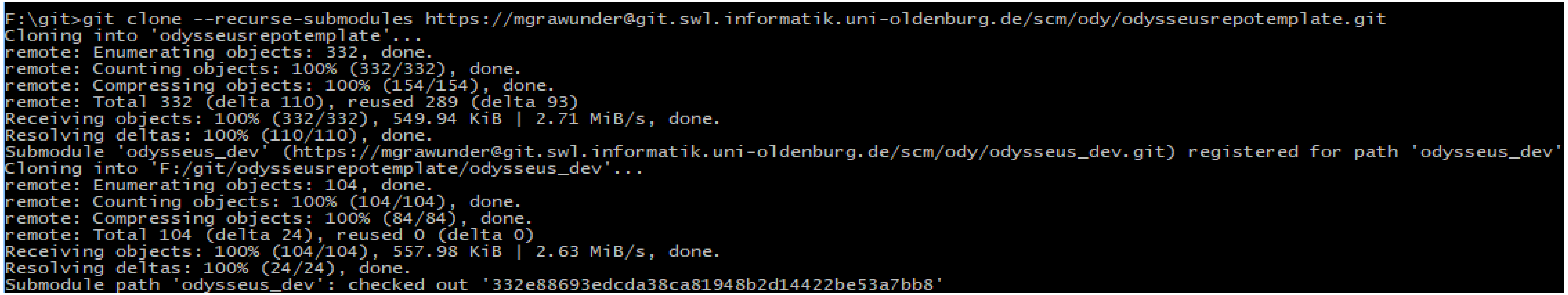
https://liveyourmark.com/feeling-lost-in-life/

Odysseus new development model

- Meanwhile (since a few days …):
  - Definition of submodules
  - Multiple repositories for Odysseus module (in different Bitbucket projects)
  - Each module has now its own update site (there is still some work)
- New development model:
  - Clone/fork a repo template (with submodules!), special template for wrapper
  - Folders for client, server, common, monolithic, resources, wrapper
  - Special submodule with target platform and products for client, server and monolithic
  - Set the target platform (→ see next slides)
  - Create your own bundles and features
  - Copy product files and add features to products
- Do not extend target platform → could be no longer compatible with the core build system
- If there are problems:
  - `odysseus_dev`-submodule could have updates
  - You need to manually update git submodules
- Drawback: Information is now spread about multiple repositories, but core stays together

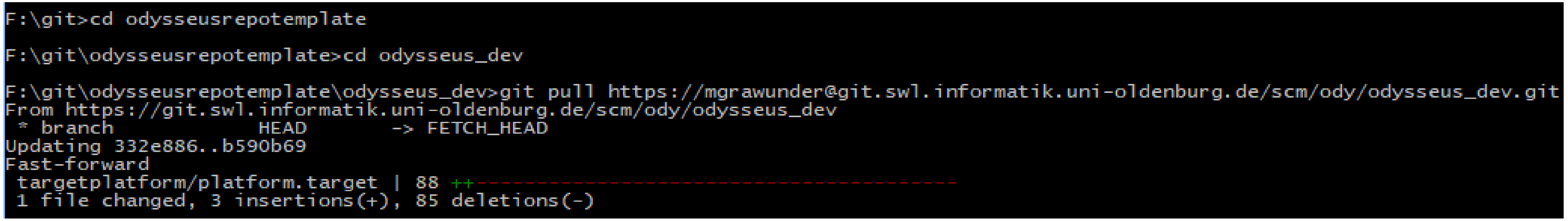
Example

- Clone template



```
F:\git>git clone --recurse-submodules https://mgrawunder@git.swl.informatik.uni-oldenburg.de/scm/ody/odysseusrepotemplate.git
Cloning into 'odysseusrepotemplate'...
remote: Enumerating objects: 332, done.
remote: Counting objects: 100% (332/332), done.
remote: Compressing objects: 100% (154/154), done.
remote: Total 332 (delta 110), reused 289 (delta 93)
Receiving objects: 100% (332/332), 549.94 KiB | 2.71 MiB/s, done.
Resolving deltas: 100% (110/110), done.
Submodule 'odysseus_dev' (https://mgrawunder@git.swl.informatik.uni-oldenburg.de/scm/ody/odysseus_dev.git) registered for path 'odysseus_dev'
Cloning into 'F:/git/odysseusrepotemplate/odysseus_dev'...
remote: Enumerating objects: 104, done.
remote: Counting objects: 100% (104/104), done.
remote: Compressing objects: 100% (84/84), done.
remote: Total 104 (delta 24), reused 0 (delta 0)
Receiving objects: 100% (104/104), 557.98 KiB | 2.63 MiB/s, done.
Resolving deltas: 100% (24/24), done.
Submodule path 'odysseus_dev': checked out '332e88693edcda38ca81948b2d14422be53a7bb8'
```

- Update odysseus_dev (if necessary)



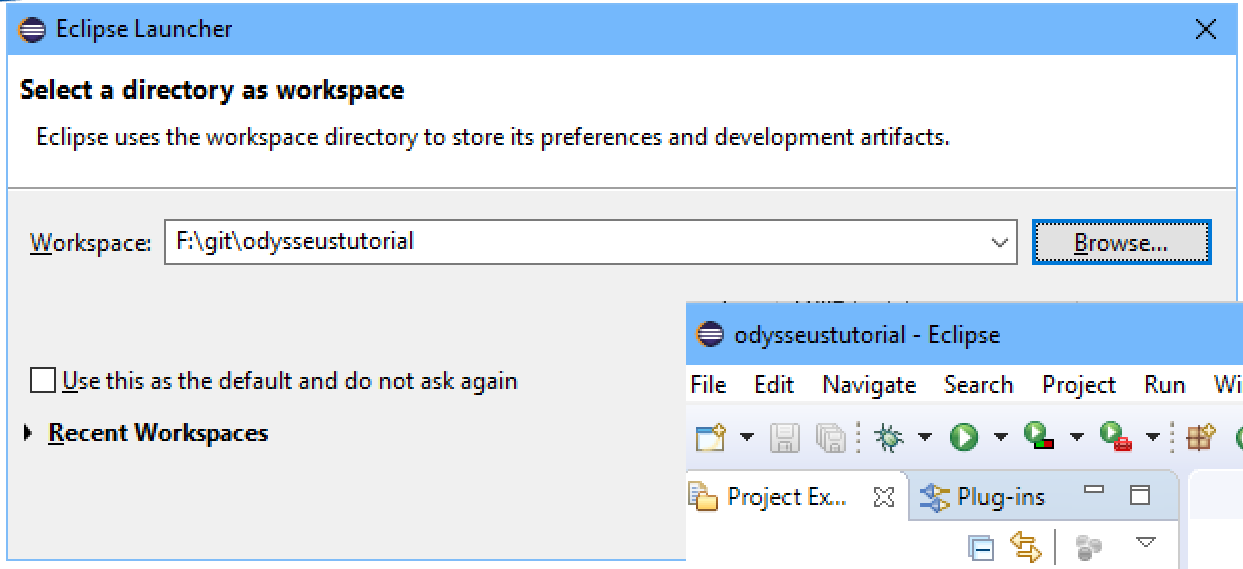
```
F:\git>cd odysseusrepotemplate

F:\git\odysseusrepotemplate>cd odysseus_dev

F:\git\odysseusrepotemplate\odysseus_dev>git pull https://mgrawunder@git.swl.informatik.uni-oldenburg.de/scm/ody/odysseus_dev.git
From https://git.swl.informatik.uni-oldenburg.de/scm/ody/odysseus_dev
 * branch            HEAD       -> FETCH_HEAD
Updating 332e886..b590b69
Fast-forward
 targetplatform/platform.target | 88 ++----------------------------------------------
 1 file changed, 3 insertions(+), 85 deletions(-)
```

Example

Rename folder (e.g. odysseustutorial) and set new origin (new git repo)

```
cd odysseustutorial
git remote set-url origin
        https://mgrawunder@git.swl.informatik.uni-oldenburg.de/scm/ody/odysseustutorial.git
git push -u origin --all
git push origin --tags
```

Start eclipse with new workspace and import

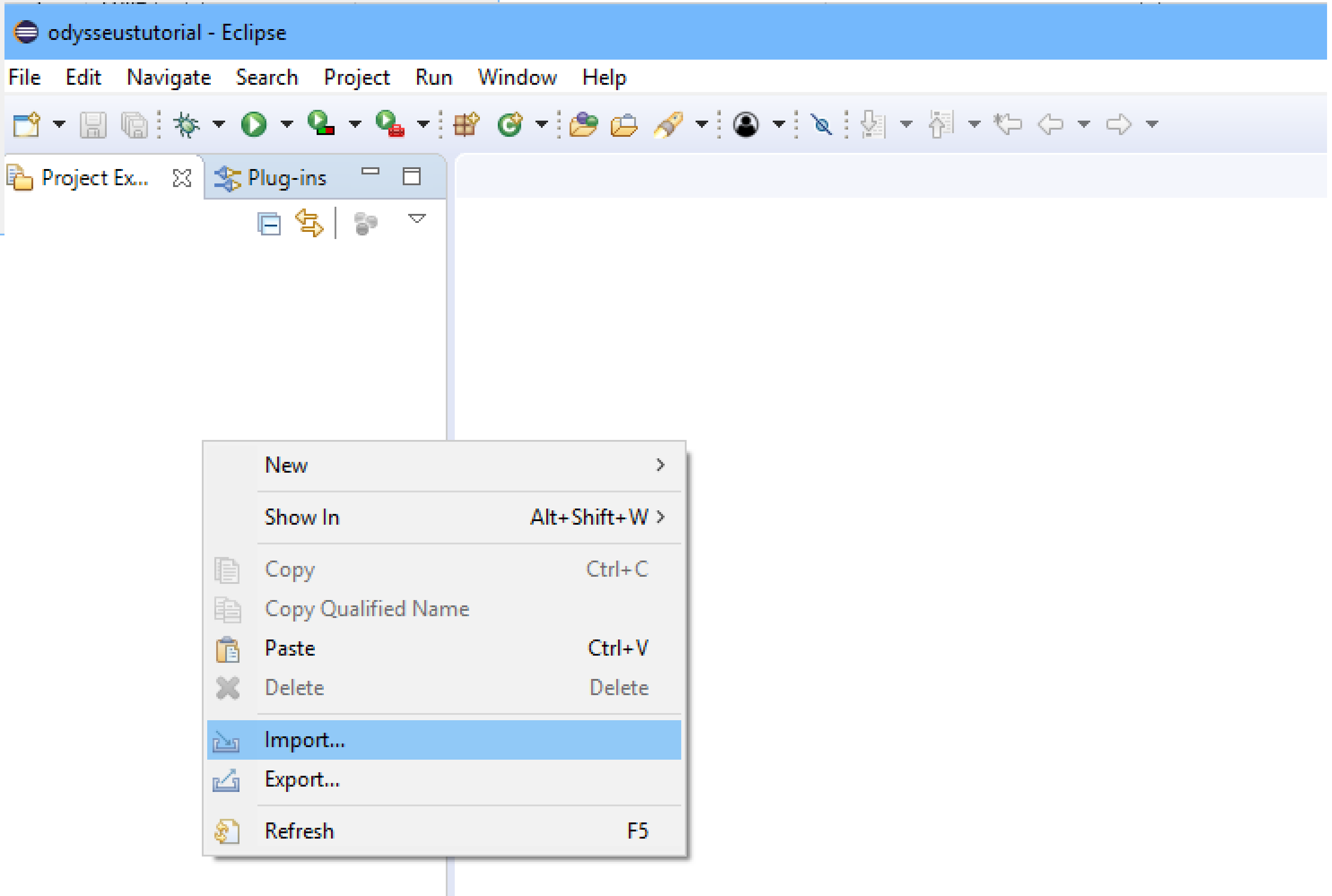


Eclipse Launcher

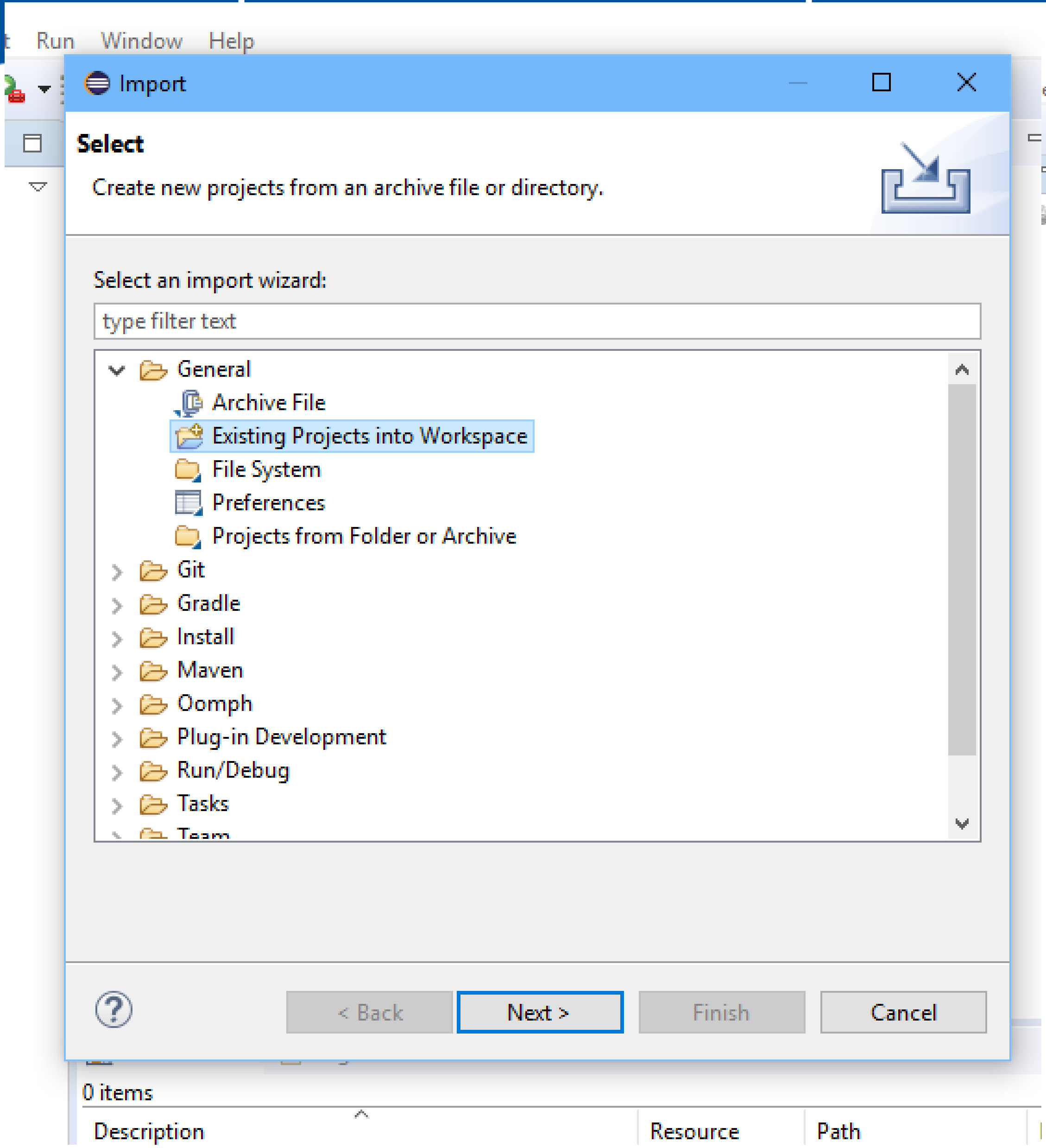
Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: F:\git\odysseustutorial



Start eclipse with new workspace and import



Import

Select

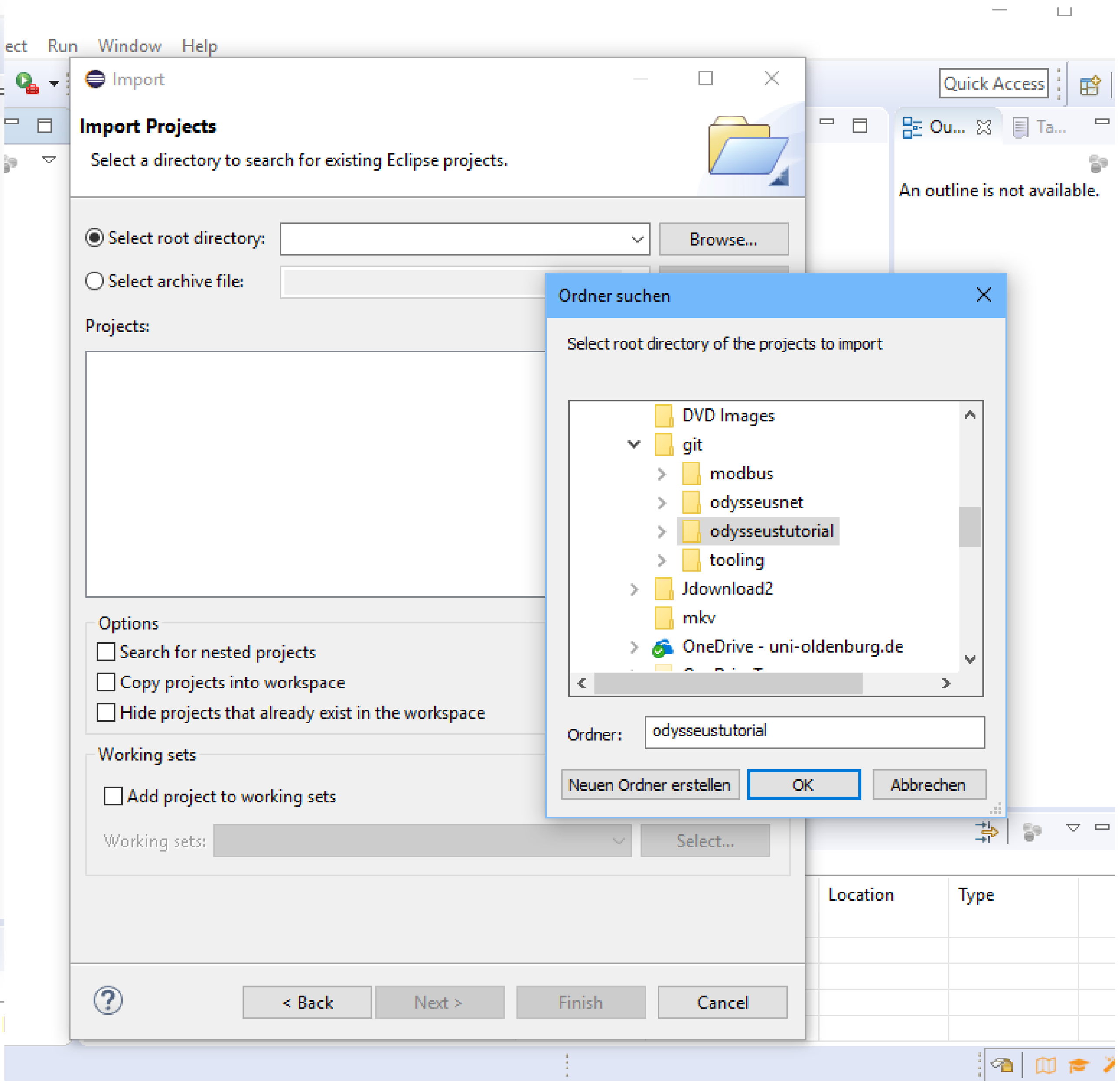
Create new projects from an archive file or directory.

Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Projects from Folder or Archive
- Git
- Gradle
- Install
- Maven
- Oomph
- Plug-in Development
- Run/Debug
- Tasks
- Team

< Back | Next > | Finish | Cancel



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: [ ] Browse...

( ) Select archive file:

Projects:

Options
- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets
- [ ] Add project to working sets

Working sets: Select...

< Back | Next > | Finish | Cancel

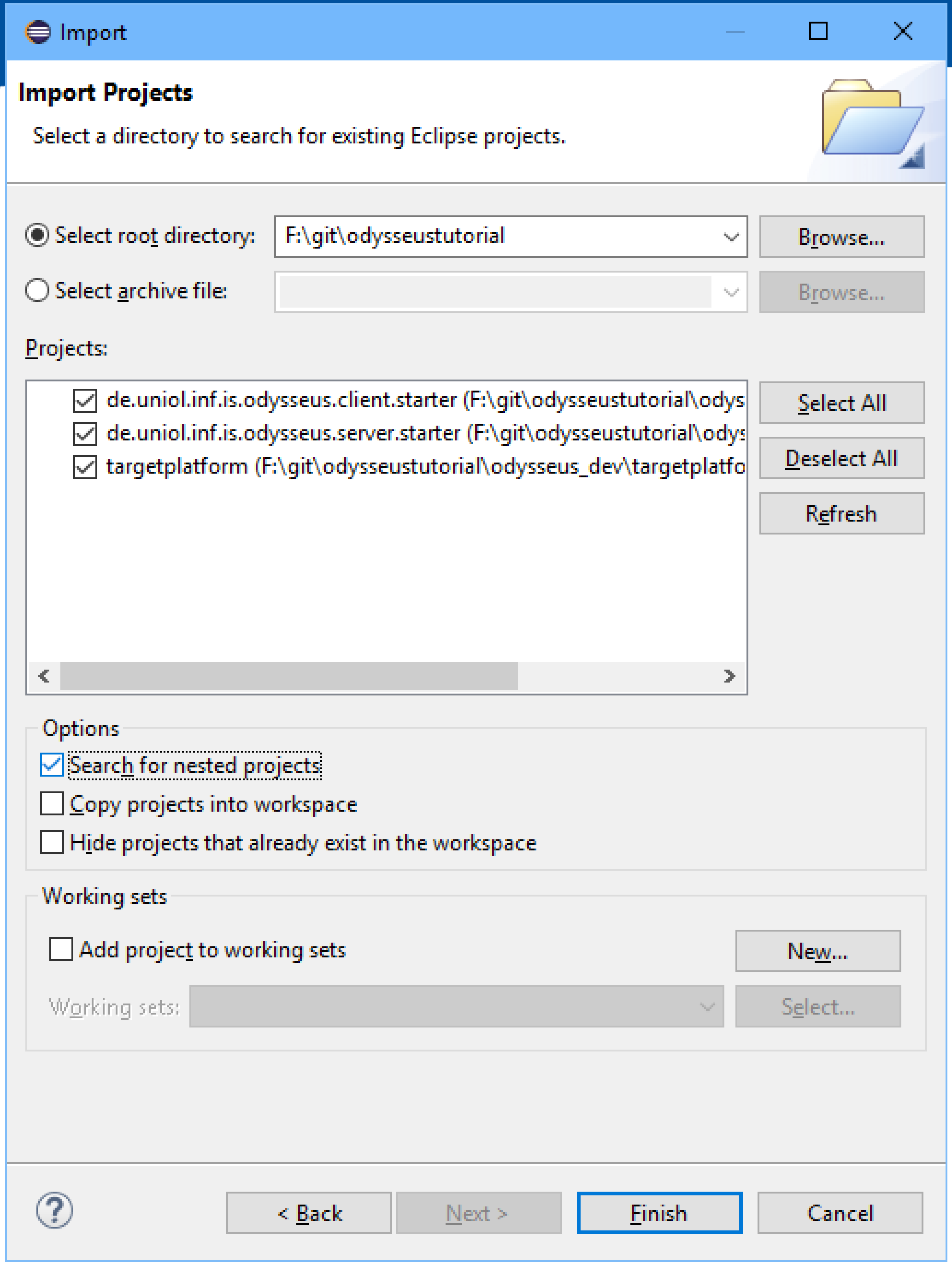
Ordner suchen

Select root directory of the projects to import

- DVD Images
- git
  - modbus
  - odysseusnet
  - odysseustutorial
  - tooling
- Jdownload2
- mkv
- OneDrive - uni-oldenburg.de

Ordner: odysseustutorial

Neuen Ordner erstellen | OK | Abbrechen


Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: F:\git\odysseustutorial

Select archive file:

Projects:

- [✓] de.uniol.inf.is.odysseus.client.starter (F:\git\odysseustutorial\odys
- [✓] de.uniol.inf.is.odysseus.server.starter (F:\git\odysseustutorial\odys
- [✓] targetplatform (F:\git\odysseustutorial\odysseus_dev\targetplatfo

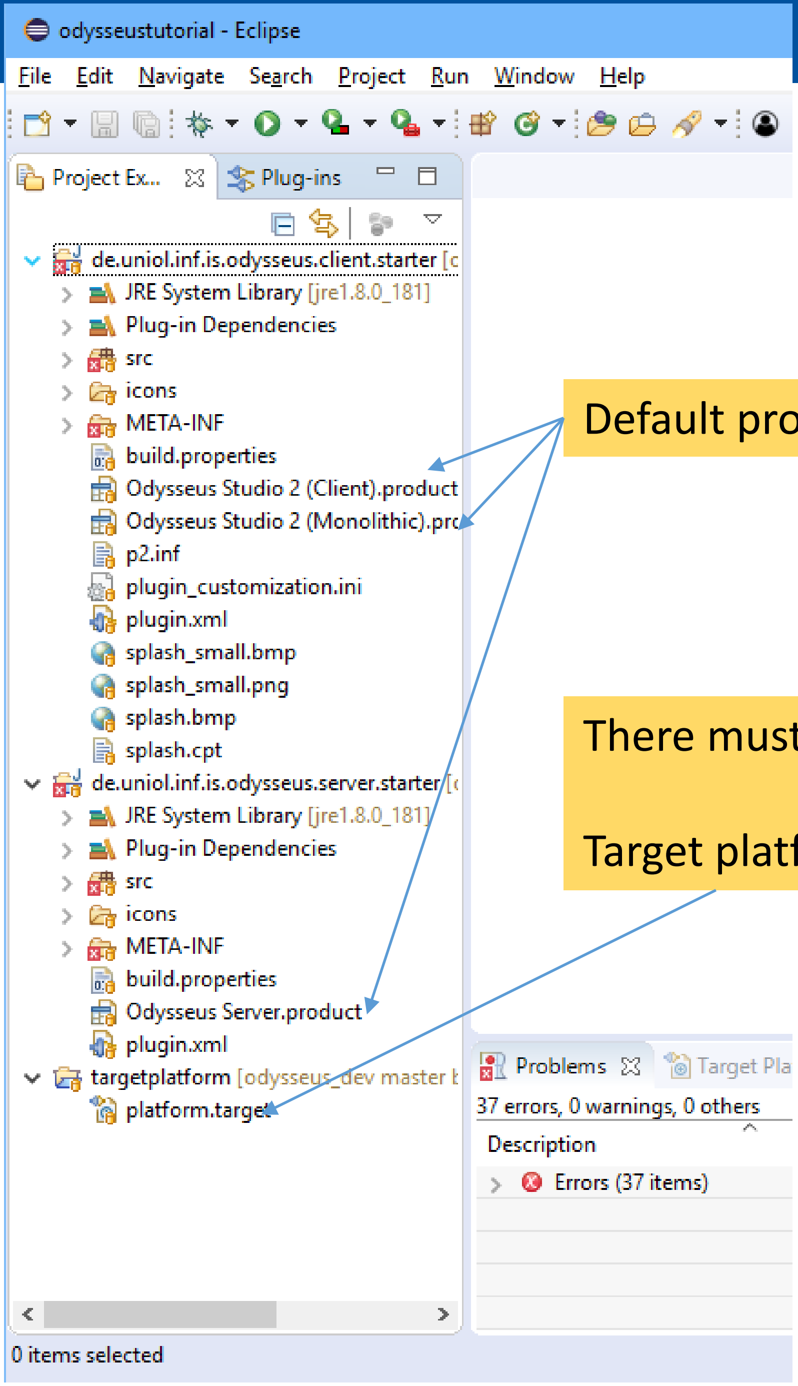
Options

- [✓] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:


odysseustutorial - Eclipse

```
de.uniol.inf.is.odysseus.client.starter [c
  JRE System Library [jre1.8.0_181]
  Plug-in Dependencies
  src
  icons
  META-INF
  build.properties
  Odysseus Studio 2 (Client).product
  Odysseus Studio 2 (Monolithic).pro
  p2.inf
  plugin_customization.ini
  plugin.xml
  splash_small.bmp
  splash_small.png
  splash.bmp
  splash.cpt
de.uniol.inf.is.odysseus.server.starter [
  JRE System Library [jre1.8.0_181]
  Plug-in Dependencies
  src
  icons
  META-INF
  build.properties
  Odysseus Server.product
  plugin.xml
targetplatform [odysseus_dev master
  platform.target
```

Problems | Target Pla

37 errors, 0 warnings, 0 others

Description

Errors (37 items)

0 items selected

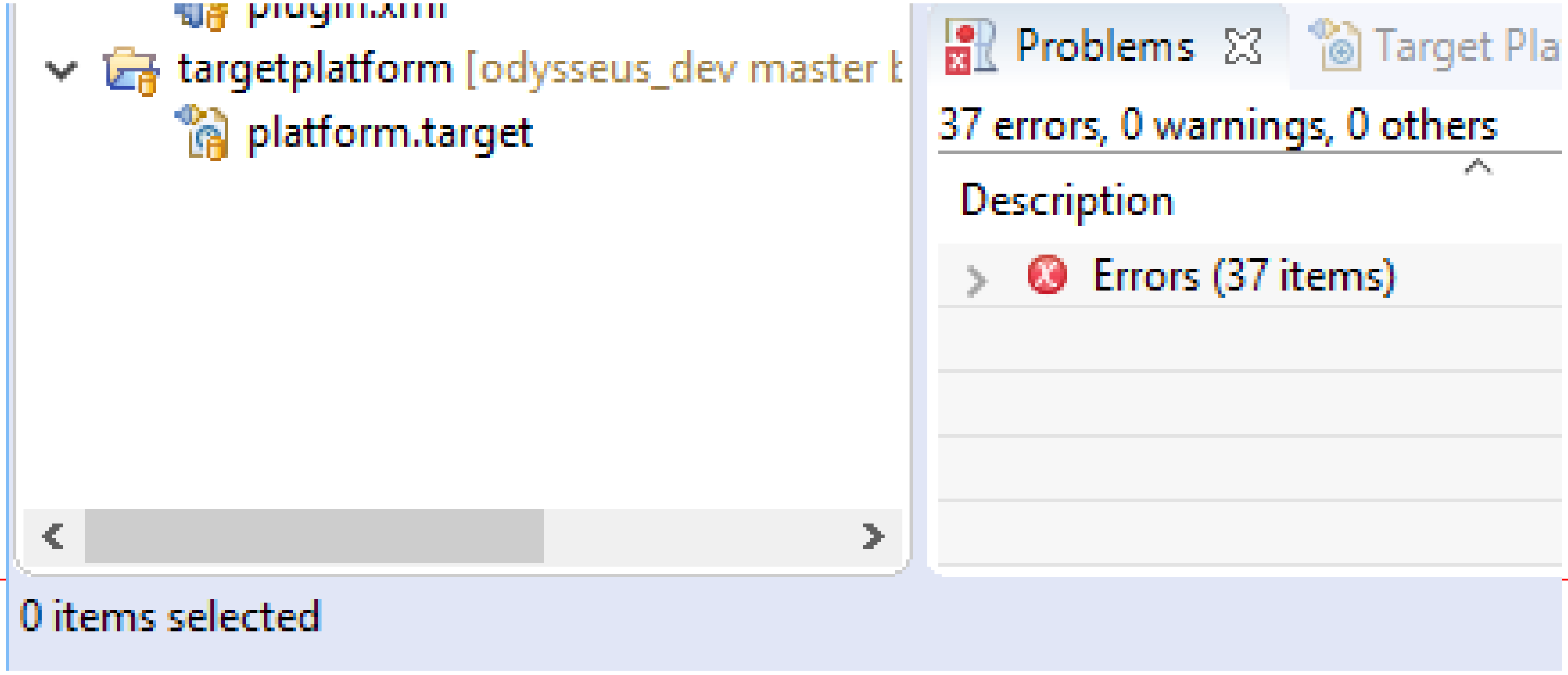
Default products

There must be errors ;-)

Target platform is not set!

Target Platform

- A target platform defines the versions of plugins that should be used in one project
- The given update site is used in our central jenkins modules build
- If you change the target platform, it is no longer compatible … but if you do not want to use the central cite you can change the target platform as you like
- Loading of target platform just by opening the file in the project target platform



targetplatform [odysseus_dev master
platform.target

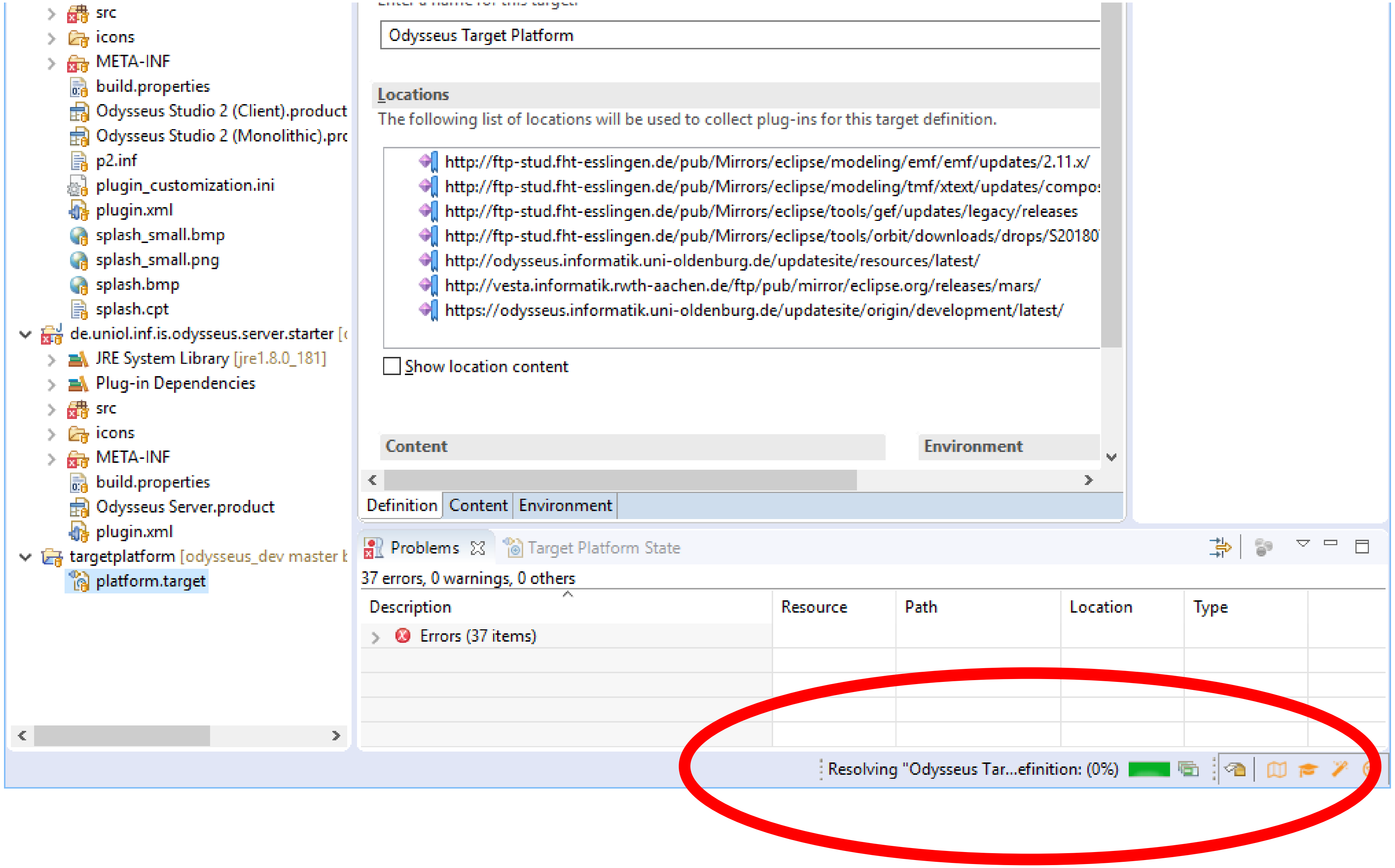
Problems | Target Pla

37 errors, 0 warnings, 0 others

Description
Errors (37 items)

0 items selected

Target Platform … be patient



src
icons
META-INF
build.properties
Odysseus Studio 2 (Client).product
Odysseus Studio 2 (Monolithic).prc
p2.inf
plugin_customization.ini
plugin.xml
splash_small.bmp
splash_small.png
splash.bmp
splash.cpt
de.uniol.inf.is.odysseus.server.starter [
JRE System Library [jre1.8.0_181]
Plug-in Dependencies
src
icons
META-INF
build.properties
Odysseus Server.product
plugin.xml
targetplatform [odysseus_dev master
platform.target

Odysseus Target Platform

Locations

The following list of locations will be used to collect plug-ins for this target definition.

http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/emf/emf/updates/2.11.x/
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/tmf/xtext/updates/compo
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/gef/updates/legacy/releases
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/orbit/downloads/drops/S20180
http://odysseus.informatik.uni-oldenburg.de/updatesite/resources/latest/
http://vesta.informatik.rwth-aachen.de/ftp/pub/mirror/eclipse.org/releases/mars/
https://odysseus.informatik.uni-oldenburg.de/updatesite/origin/development/latest/

☐ Show location content

Content **Environment**

Definition | Content | Environment

Problems | Target Platform State

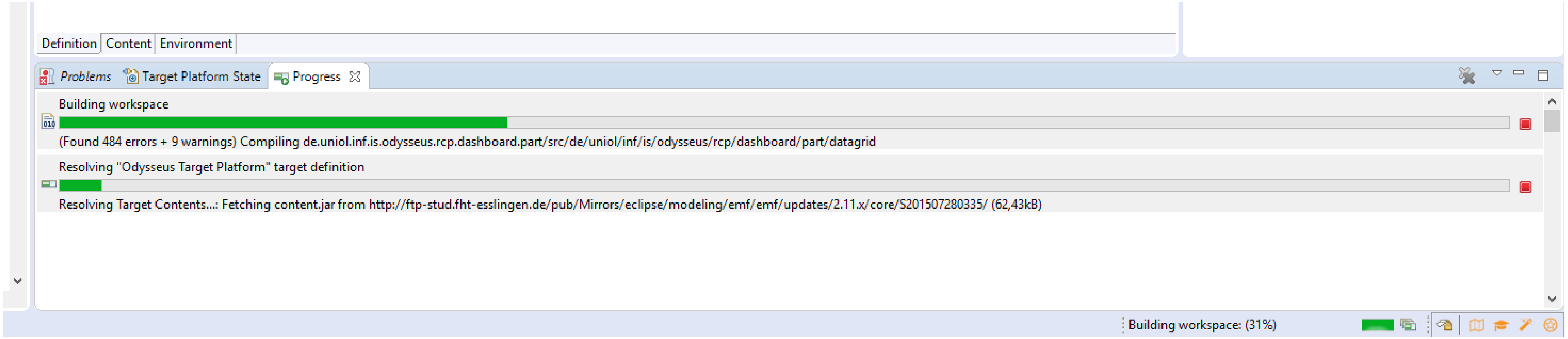
37 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (37 items)				

Resolving "Odysseus Tar...efinition: (0%)

Target Platform …

- Setting target platform takes time …
  - be patient!
  - Do not use eclipse while target platform gets updated



Building workspace

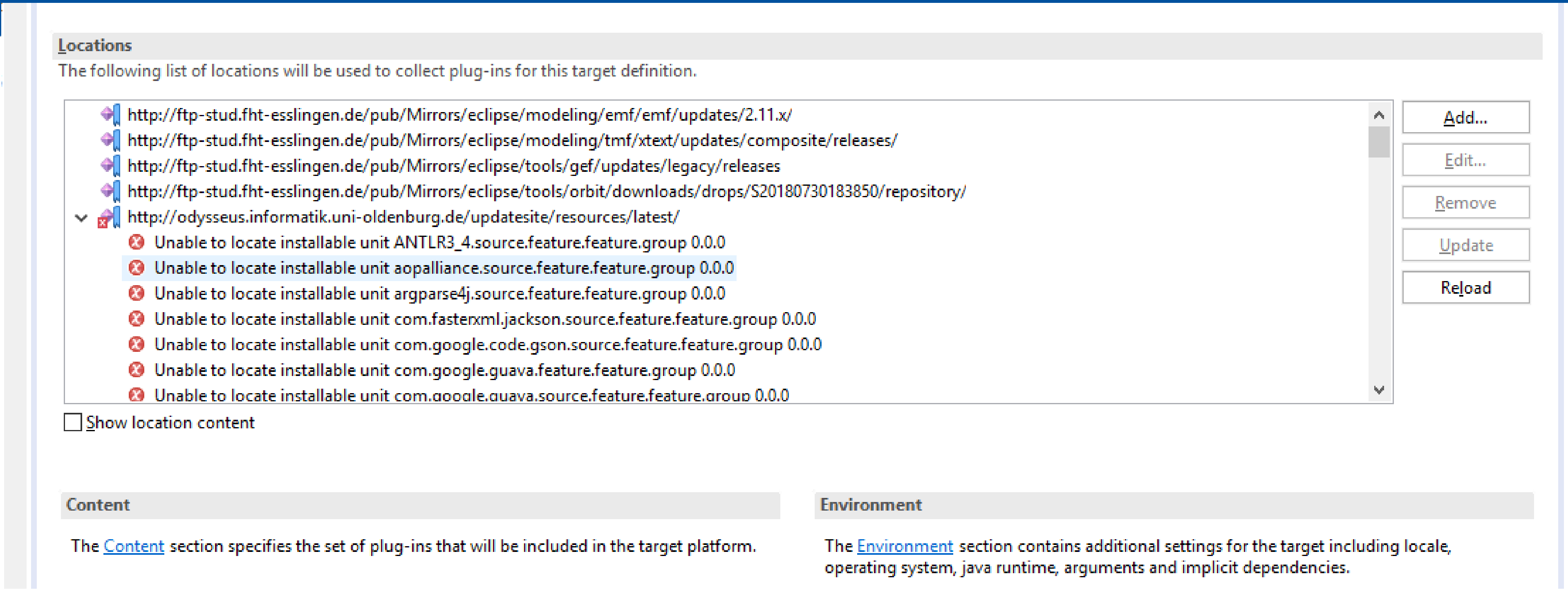
(Found 484 errors + 9 warnings) Compiling de.uniol.inf.is.odysseus.rcp.dashboard.part/src/de/uniol/inf/is/odysseus/rcp/dashboard/part/datagrid

Resolving "Odysseus Target Platform" target definition

Resolving Target Contents...: Fetching content.jar from http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/emf/emf/updates/2.11.x/core/S201507280335/ (62,43kB)

Building workspace: (31%)

Target Platform … Sometimes:



The following list of locations will be used to collect plug-ins for this target definition.

```
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/emf/emf/updates/2.11.x/
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/tmf/xtext/updates/composite/releases/
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/gef/updates/legacy/releases
http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/orbit/downloads/drops/S20180730183850/repository/
http://odysseus.informatik.uni-oldenburg.de/updatesite/resources/latest/
    Unable to locate installable unit ANTLR3_4.source.feature.feature.group 0.0.0
    Unable to locate installable unit aopalliance.source.feature.feature.group 0.0.0
    Unable to locate installable unit argparse4j.source.feature.feature.group 0.0.0
    Unable to locate installable unit com.fasterxml.jackson.source.feature.feature.group 0.0.0
    Unable to locate installable unit com.google.code.gson.source.feature.feature.group 0.0.0
    Unable to locate installable unit com.google.guava.feature.feature.group 0.0.0
    Unable to locate installable unit com.google.guava.source.feature.feature.group 0.0.0
```

☐ Show location content

The Content section specifies the set of plug-ins that will be included in the target platform.

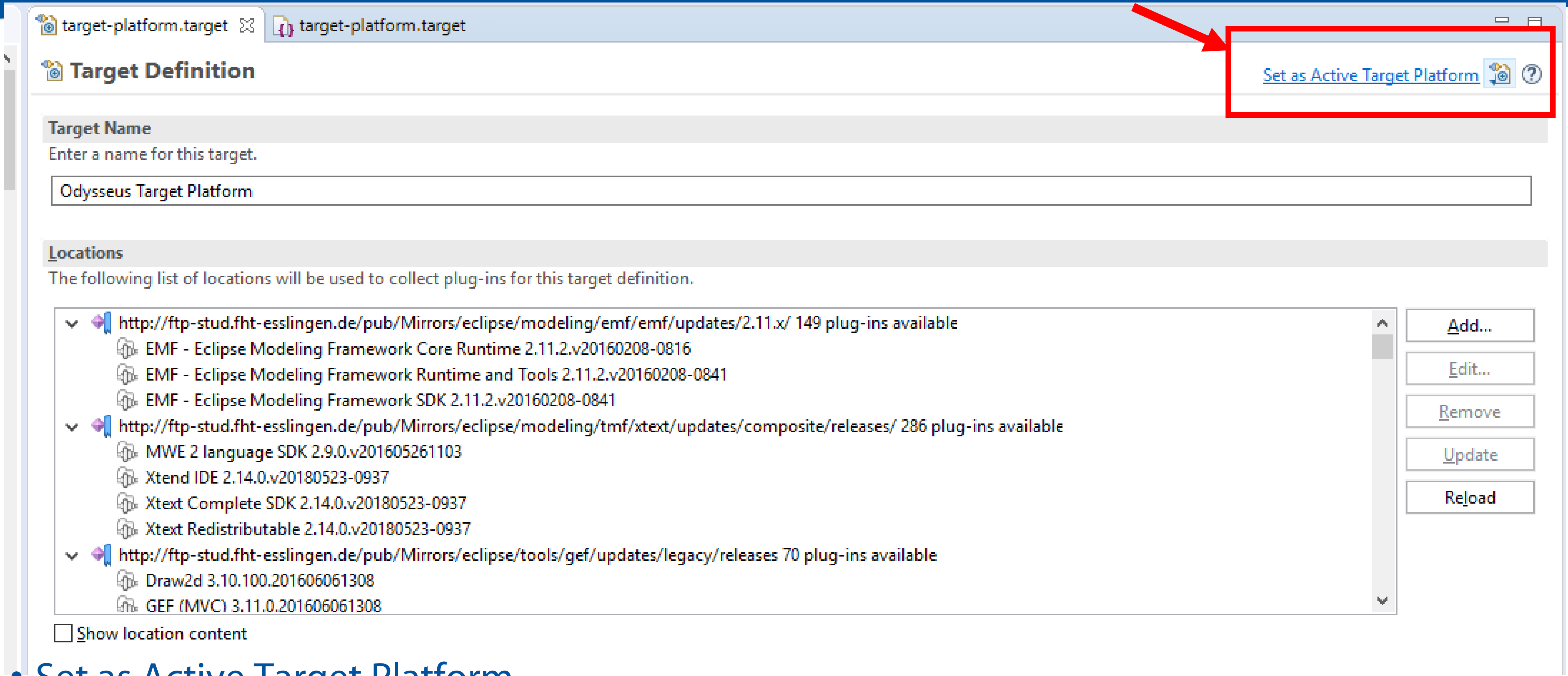
The Environment section contains additional settings for the target including locale, operating system, java runtime, arguments and implicit dependencies.

- Eclipse is sometimes very … strange
- Different ways to solve this:
  1) Stop eclipse and remove .metadata in workspace and import project again
  2) Restart Eclipse and increase sequence number in target platform

```
<?pde version="3.8"?><target name="Odysseus Target Platform" sequenceNumber="1502128613">
```

  3) Target platform is part of git **submodule** … try to update this

Target Platform … hopefully



Target Definition

Target Name
Enter a name for this target.
Odysseus Target Platform

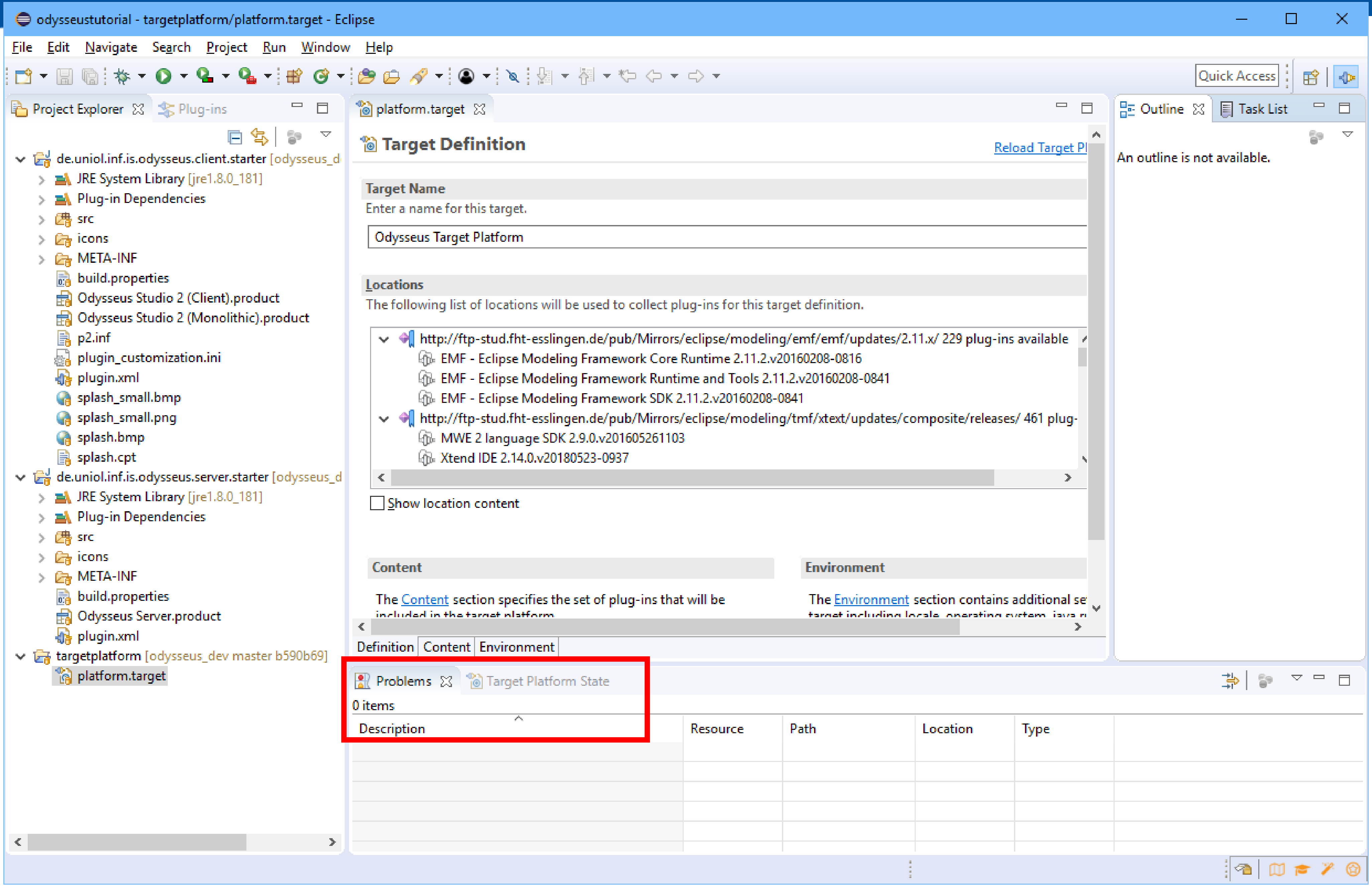
Locations
The following list of locations will be used to collect plug-ins for this target definition.

- http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/emf/emf/updates/2.11.x/ 149 plug-ins available
  - EMF - Eclipse Modeling Framework Core Runtime 2.11.2.v20160208-0816
  - EMF - Eclipse Modeling Framework Runtime and Tools 2.11.2.v20160208-0841
  - EMF - Eclipse Modeling Framework SDK 2.11.2.v20160208-0841
- http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/tmf/xtext/updates/composite/releases/ 286 plug-ins available
  - MWE 2 language SDK 2.9.0.v201605261103
  - Xtend IDE 2.14.0.v20180523-0937
  - Xtext Complete SDK 2.14.0.v20180523-0937
  - Xtext Redistributable 2.14.0.v20180523-0937
- http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/gef/updates/legacy/releases 70 plug-ins available
  - Draw2d 3.10.100.201606061308
  - GEF (MVC) 3.11.0.201606061308

☐ Show location content

- Set as Active Target Platform,
- Remark: Do not set before the target platform is fully resolved!
- Refresh Workspace
- Clean and Build (in future maven/tycho will be an option, too)

And finally …



odysseustutorial - targetplatform/platform.target - Eclipse

- de.uniol.inf.is.odysseus.client.starter [odysseus_d
  - JRE System Library [jre1.8.0_181]
  - Plug-in Dependencies
  - src
  - icons
  - META-INF
  - build.properties
  - Odysseus Studio 2 (Client).product
  - Odysseus Studio 2 (Monolithic).product
  - p2.inf
  - plugin_customization.ini
  - plugin.xml
  - splash_small.bmp
  - splash_small.png
  - splash.bmp
  - splash.cpt
- de.uniol.inf.is.odysseus.server.starter [odysseus_d
  - JRE System Library [jre1.8.0_181]
  - Plug-in Dependencies
  - src
  - icons
  - META-INF
  - build.properties
  - Odysseus Server.product
  - plugin.xml
- targetplatform [odysseus_dev master b590b69]
  - platform.target

Target Definition

Reload Target Pl

Target Name

Enter a name for this target.

Odysseus Target Platform

Locations

The following list of locations will be used to collect plug-ins for this target definition.

- http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/emf/emf/updates/2.11.x/ 229 plug-ins available
  - EMF - Eclipse Modeling Framework Core Runtime 2.11.2.v20160208-0816
  - EMF - Eclipse Modeling Framework Runtime and Tools 2.11.2.v20160208-0841
  - EMF - Eclipse Modeling Framework SDK 2.11.2.v20160208-0841
- http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/modeling/tmf/xtext/updates/composite/releases/ 461 plug-
  - MWE 2 language SDK 2.9.0.v201605261103
  - Xtend IDE 2.14.0.v20180523-0937

Show location content

Content

The Content section specifies the set of plug-ins that will be included in the target platform

Environment

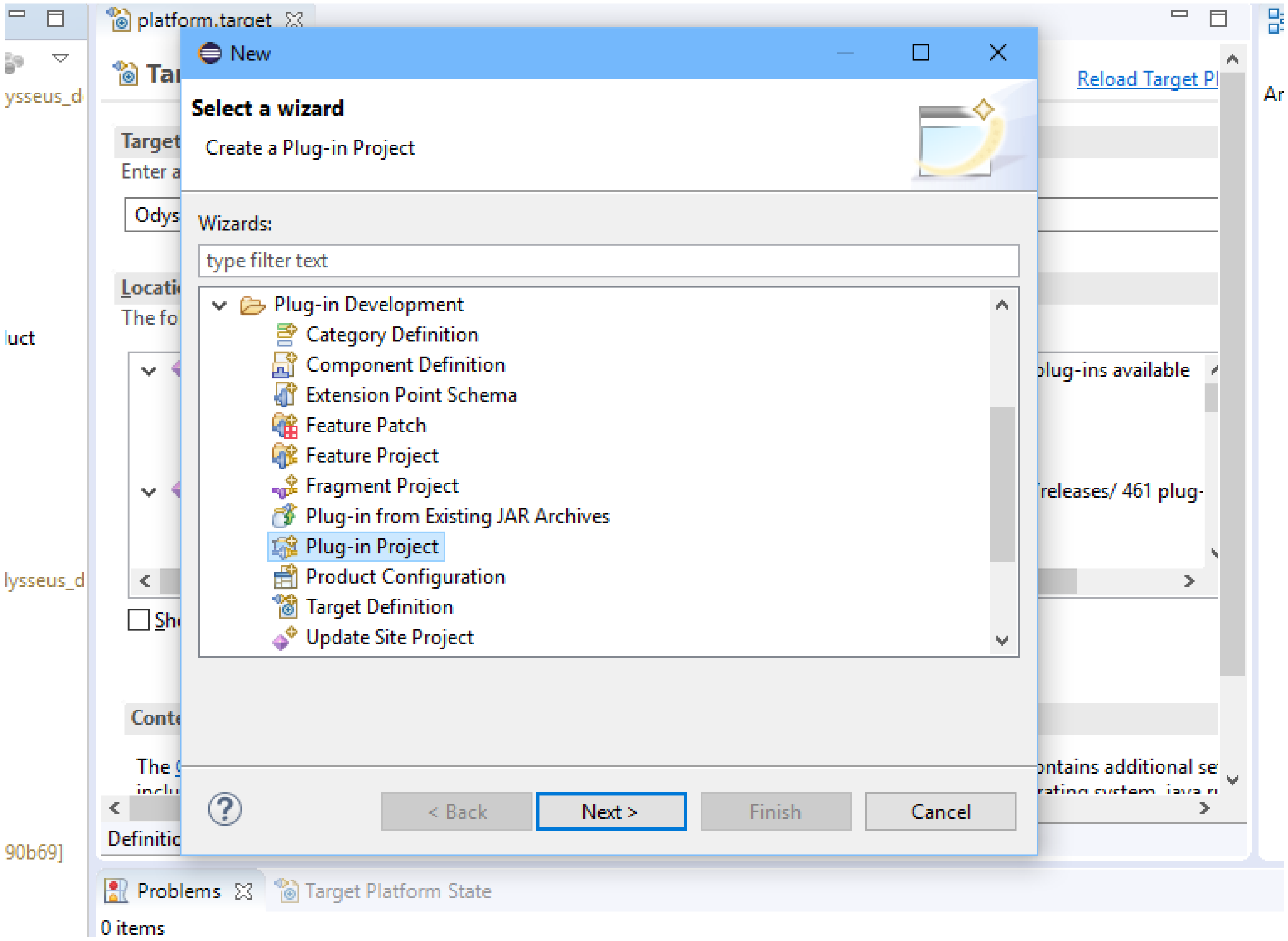
The Environment section contains additional se target including locale, operating system, java r

An outline is not available.

0 items

Description	Resource	Path	Location	Type

Now create your own bundles and feature



New

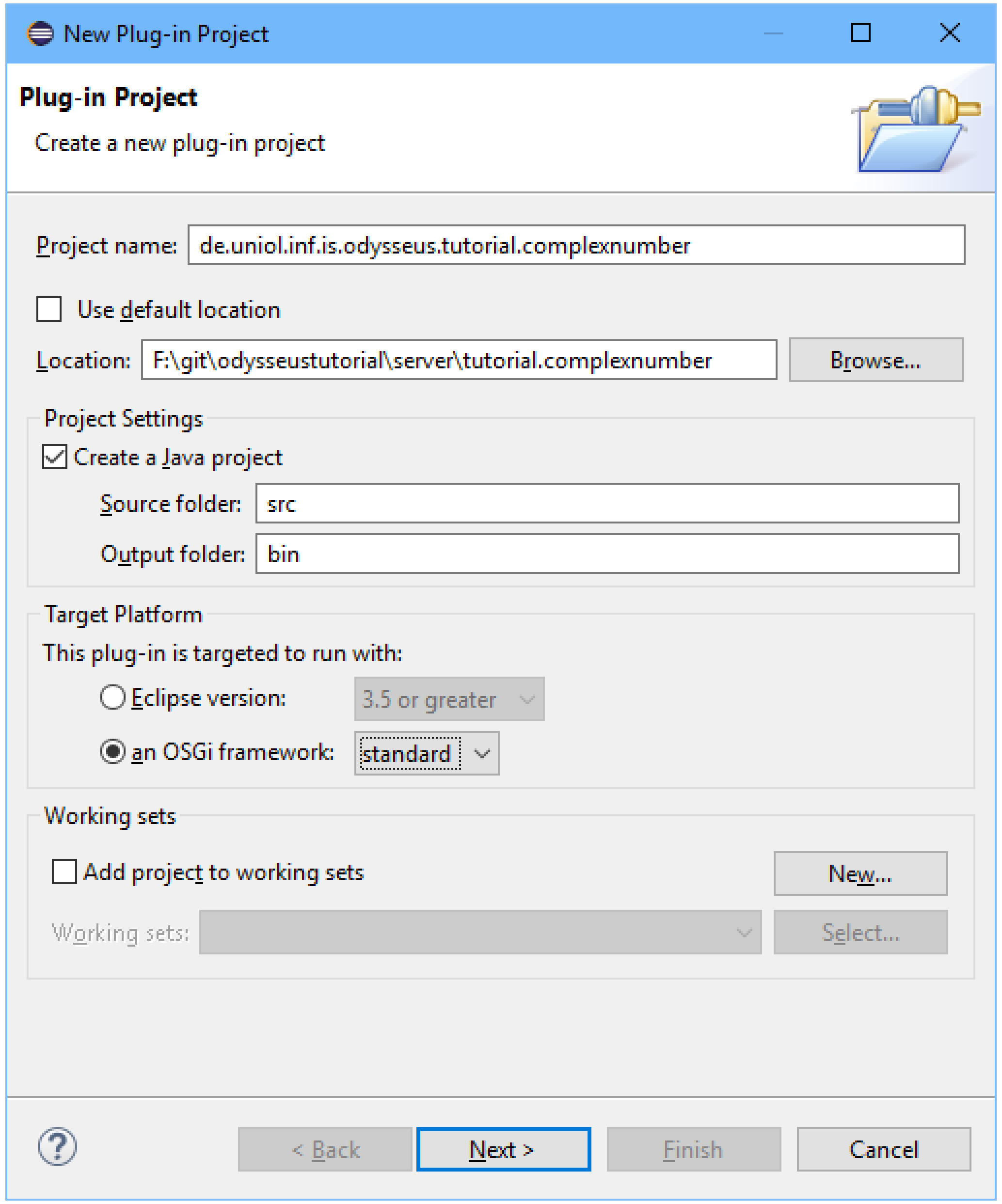
Select a wizard

Create a Plug-in Project

Wizards:

type filter text

- Plug-in Development
  - Category Definition
  - Component Definition
  - Extension Point Schema
  - Feature Patch
  - Feature Project
  - Fragment Project
  - Plug-in from Existing JAR Archives
  - Plug-in Project
  - Product Configuration
  - Target Definition
  - Update Site Project


New Plug-in Project

Plug-in Project

Create a new plug-in project

Project name: de.uniol.inf.is.odysseus.tutorial.complexnumber

☐ Use default location

Location: F:\git\odysseustutorial\server\tutorial.complexnumber

Project Settings

☑ Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

○ Eclipse version: 3.5 or greater

● an OSGi framework: standard

Working sets

☐ Add project to working sets

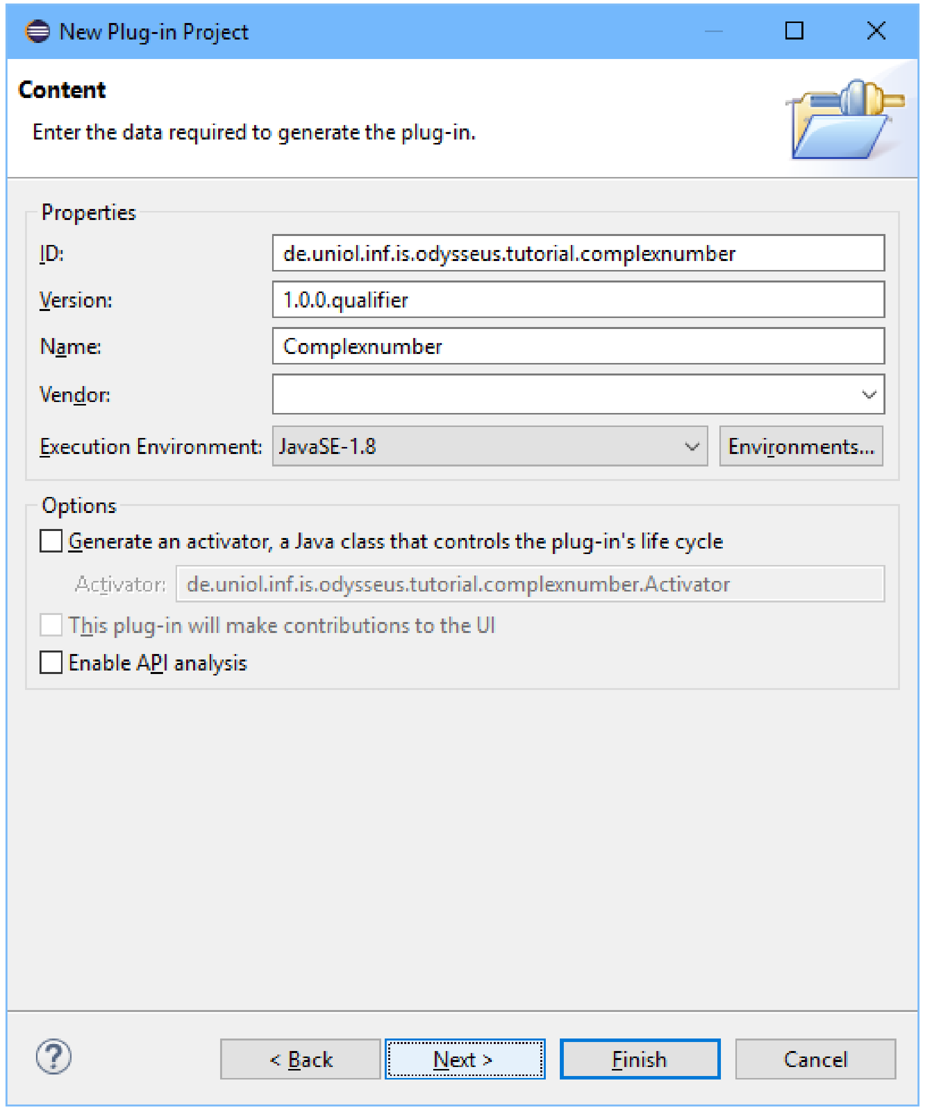
Working sets:

Create name (must be unique inside an Odysseus installation!)

Recommendation: Place bundle in one of the given subfolders:

- common: For shared classes between client and server
- client: Only for client, dependency only to common & resources
- monolithic: client and server together (no remote working)
- server: server components, dep. only to common & resources
- wrapper: special server component, dependencies to common, resources, server allowed
- resources: place resources not available on target platform here

Use standard OSGi framework


New Plug-in Project

Content

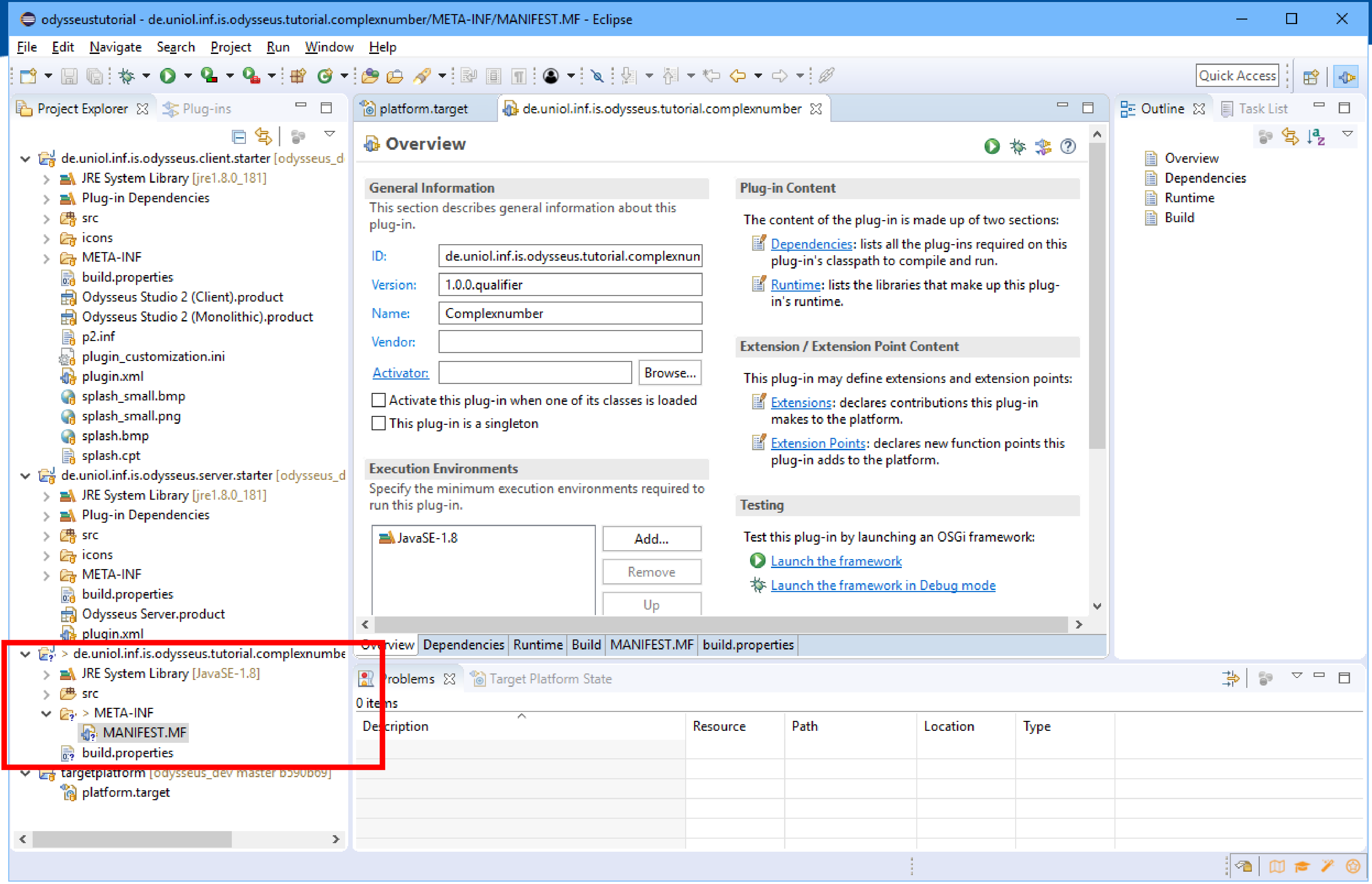
Enter the data required to generate the plug-in.

Properties

ID:	de.uniol.inf.is.odysseus.tutorial.complexnumber
Version:	1.0.0.qualifier
Name:	Complexnumber
Vendor:	
Execution Environment:	JavaSE-1.8

Options

- [ ] Generate an activator, a Java class that controls the plug-in's life cycle
  - Activator: de.uniol.inf.is.odysseus.tutorial.complexnumber.Activator
- [ ] This plug-in will make contributions to the UI
- [ ] Enable API analysis



odysseustutorial - de.uniol.inf.is.odysseus.tutorial.complexnumber/META-INF/MANIFEST.MF - Eclipse

File Edit Navigate Search Project Run Window Help

Project Explorer | Plug-ins

- de.uniol.inf.is.odysseus.client.starter [odysseus_d
  - JRE System Library [jre1.8.0_181]
  - Plug-in Dependencies
  - src
  - icons
  - META-INF
  - build.properties
  - Odysseus Studio 2 (Client).product
  - Odysseus Studio 2 (Monolithic).product
  - p2.inf
  - plugin_customization.ini
  - plugin.xml
  - splash_small.bmp
  - splash_small.png
  - splash.bmp
  - splash.cpt
- de.uniol.inf.is.odysseus.server.starter [odysseus_d
  - JRE System Library [jre1.8.0_181]
  - Plug-in Dependencies
  - src
  - icons
  - META-INF
  - build.properties
  - Odysseus Server.product
  - plugin.xml
- > de.uniol.inf.is.odysseus.tutorial.complexnumbe
  - JRE System Library [JavaSE-1.8]
  - src
  - > META-INF
    - MANIFEST.MF
  - build.properties
- targetplatform [odysseus_dev master b590b69]
  - platform.target

platform.target | de.uniol.inf.is.odysseus.tutorial.complexnumber

Overview

General Information

This section describes general information about this plug-in.

ID:	de.uniol.inf.is.odysseus.tutorial.complexnun
Version:	1.0.0.qualifier
Name:	Complexnumber
Vendor:	
Activator:	

Browse...

- [ ] Activate this plug-in when one of its classes is loaded
- [ ] This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.8

Add... Remove Up

Plug-in Content

The content of the plug-in is made up of two sections:

- Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.
- Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points:

- Extensions: declares contributions this plug-in makes to the platform.
- Extension Points: declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching an OSGi framework:

- Launch the framework
- Launch the framework in Debug mode

Overview | Dependencies | Runtime | Build | MANIFEST.MF | build.properties

Outline | Task List

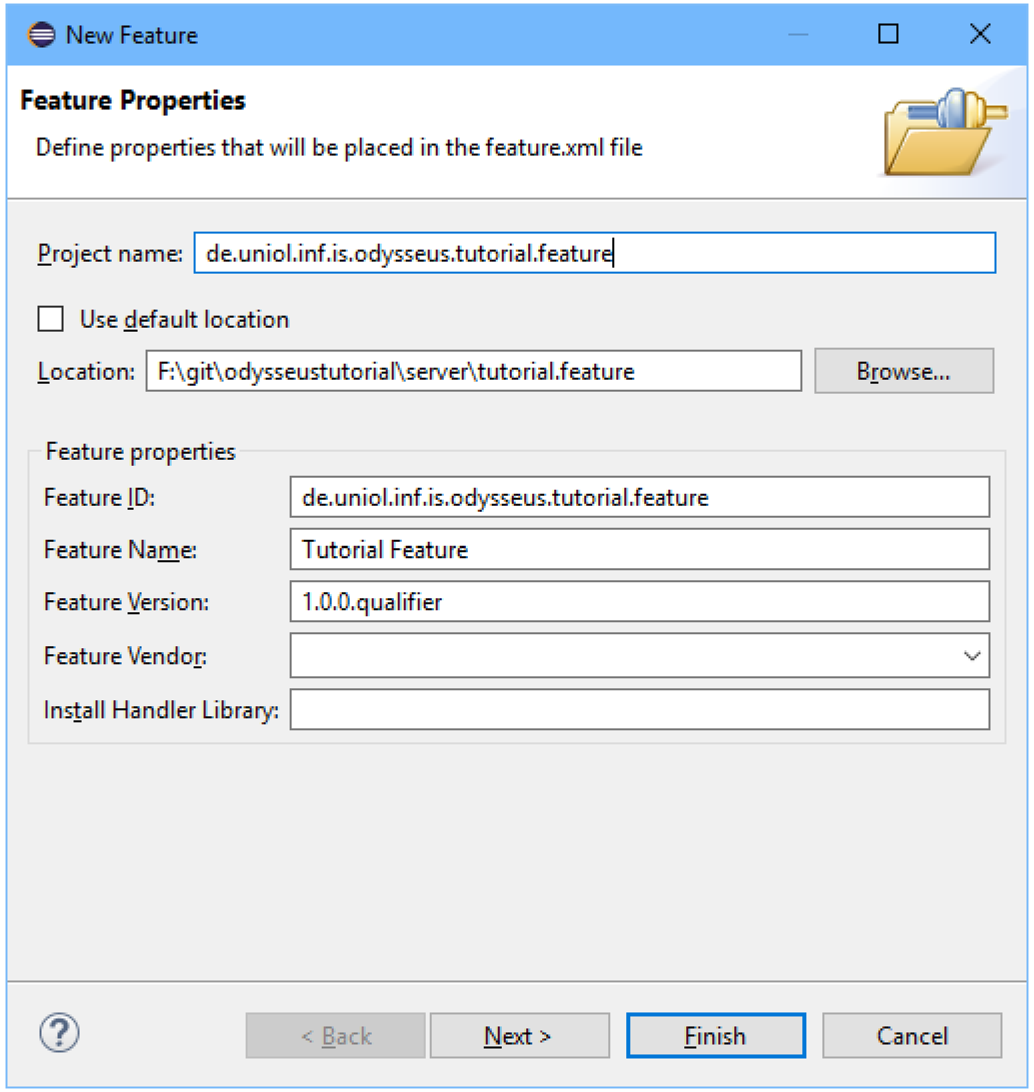
- Overview
- Dependencies
- Runtime
- Build

Problems | Target Platform State

0 items

Description	Resource	Path	Location	Type

Create Feature



New Feature

Feature Properties

Define properties that will be placed in the feature.xml file

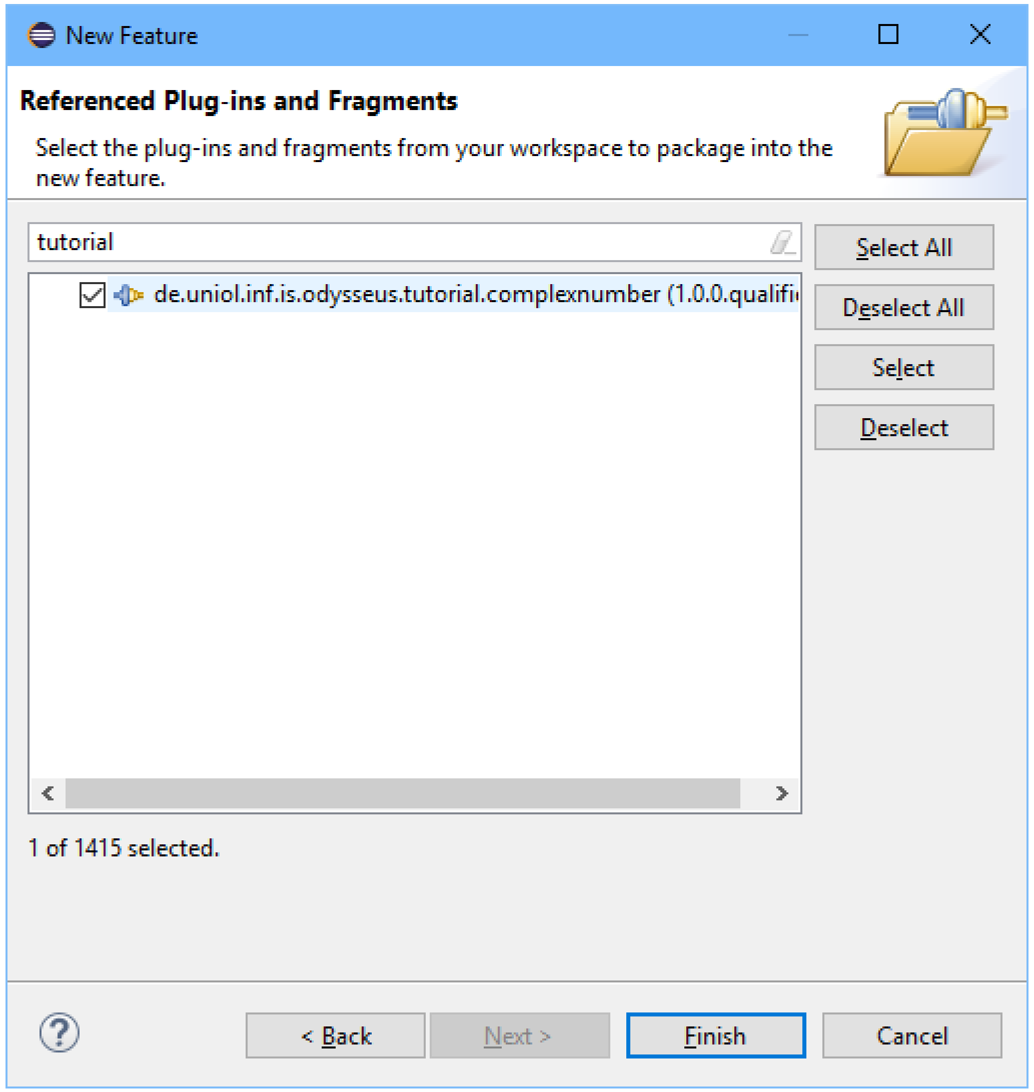
Project name: de.uniol.inf.is.odysseus.tutorial.feature

☐ Use default location

Location: F:\git\odysseustutorial\server\tutorial.feature

Feature properties

Feature ID:	de.uniol.inf.is.odysseus.tutorial.feature
Feature Name:	Tutorial Feature
Feature Version:	1.0.0.qualifier
Feature Vendor:	
Install Handler Library:	



New Feature

Referenced Plug-ins and Fragments

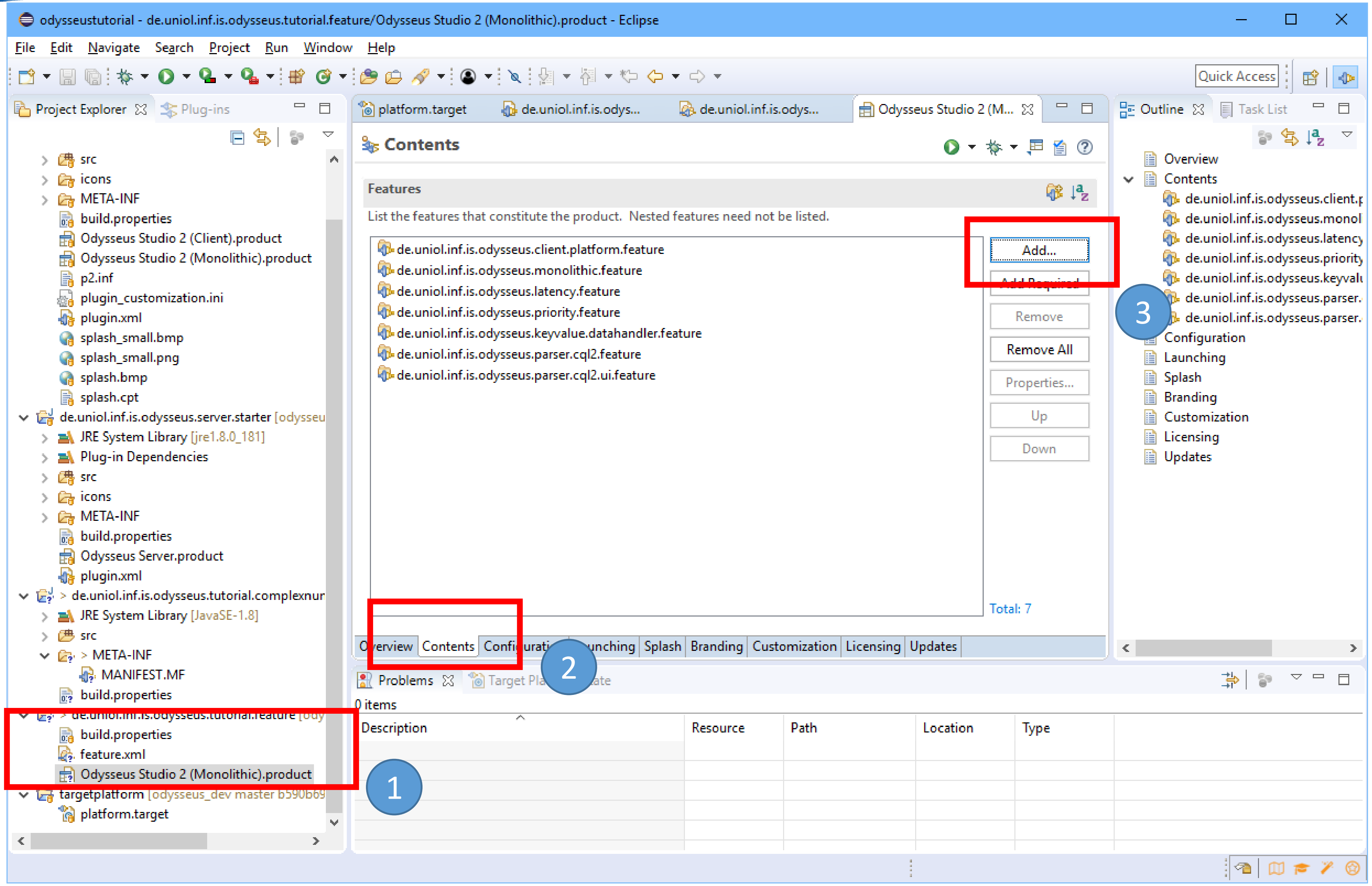
Select the plug-ins and fragments from your workspace to package into the new feature.

tutorial

☑ de.uniol.inf.is.odysseus.tutorial.complexnumber (1.0.0.qualifi

1 of 1415 selected.

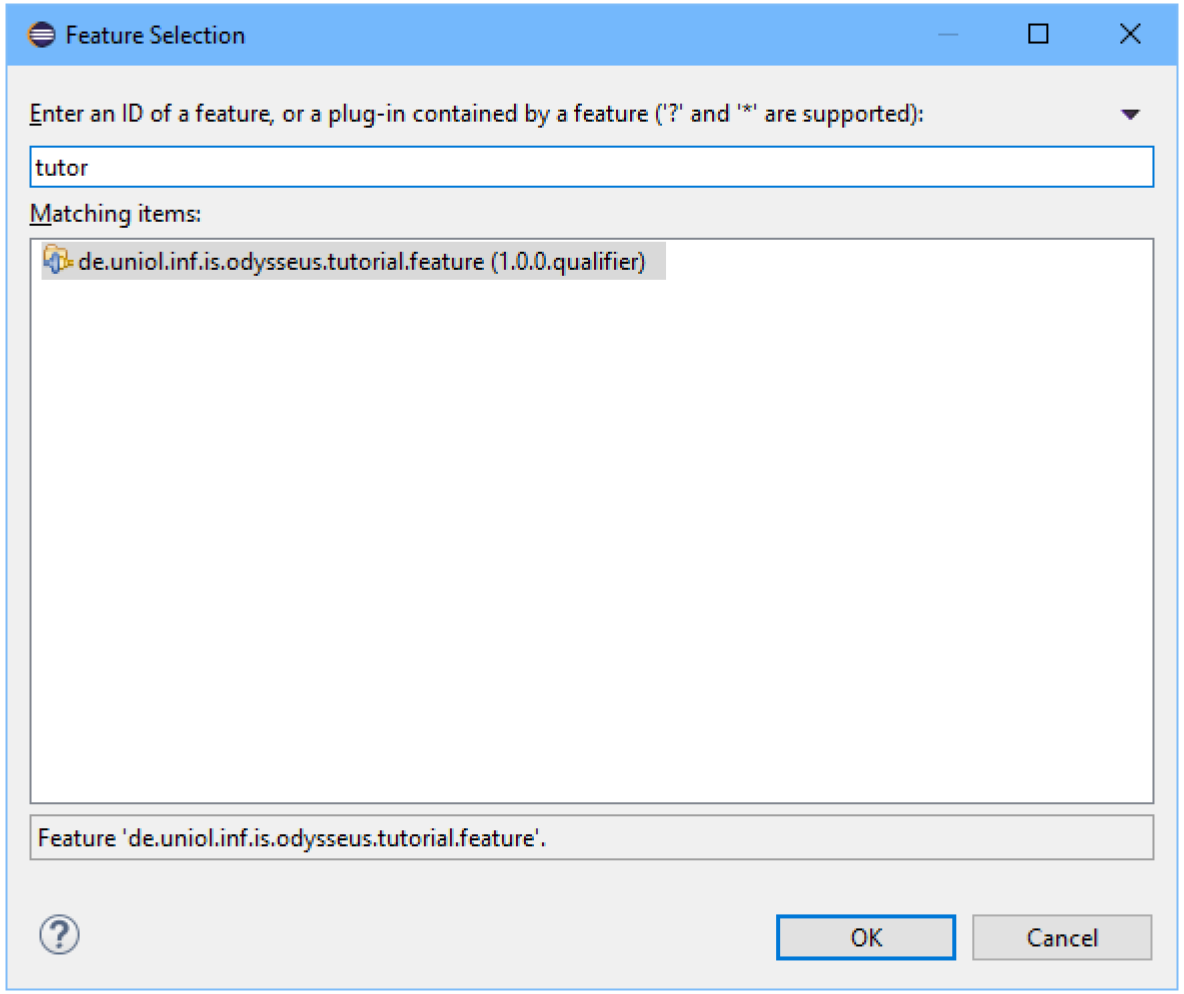
Copy product (e.g. to feature project), Add feature to product



odysseustutorial - de.uniol.inf.is.odysseus.tutorial.feature/Odysseus Studio 2 (Monolithic).product - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
Plug-ins
src
icons
META-INF
build.properties
Odysseus Studio 2 (Client).product
Odysseus Studio 2 (Monolithic).product
p2.inf
plugin_customization.ini
plugin.xml
splash_small.bmp
splash_small.png
splash.bmp
splash.cpt
de.uniol.inf.is.odysseus.server.starter [odysseu
JRE System Library [jre1.8.0_181]
Plug-in Dependencies
Odysseus Server.product
de.uniol.inf.is.odysseus.tutorial.complexnur
JRE System Library [JavaSE-1.8]
MANIFEST.MF
de.uniol.inf.is.odysseus.tutorial.feature
feature.xml
targetplatform [odysseus_dev master b590b69
platform.target
Contents
Features
List the features that constitute the product. Nested features need not be listed.
Add...
Add Required
Remove
Remove All
Properties...
Up
Down
Total: 7
Overview
Contents
Configuration
Launching
Splash
Branding
Customization
Licensing
Updates
Problems
0 items
Description
Resource
Path
Location
Type
Outline
Task List
Quick Access
1
2
3

de.uniol.inf.is.odysseus.client.platform.feature
de.uniol.inf.is.odysseus.monolithic.feature
de.uniol.inf.is.odysseus.latency.feature
de.uniol.inf.is.odysseus.priority.feature
de.uniol.inf.is.odysseus.keyvalue.datahandler.feature
de.uniol.inf.is.odysseus.parser.cql2.feature
de.uniol.inf.is.odysseus.parser.cql2.ui.feature

Add feature to product



Feature Selection

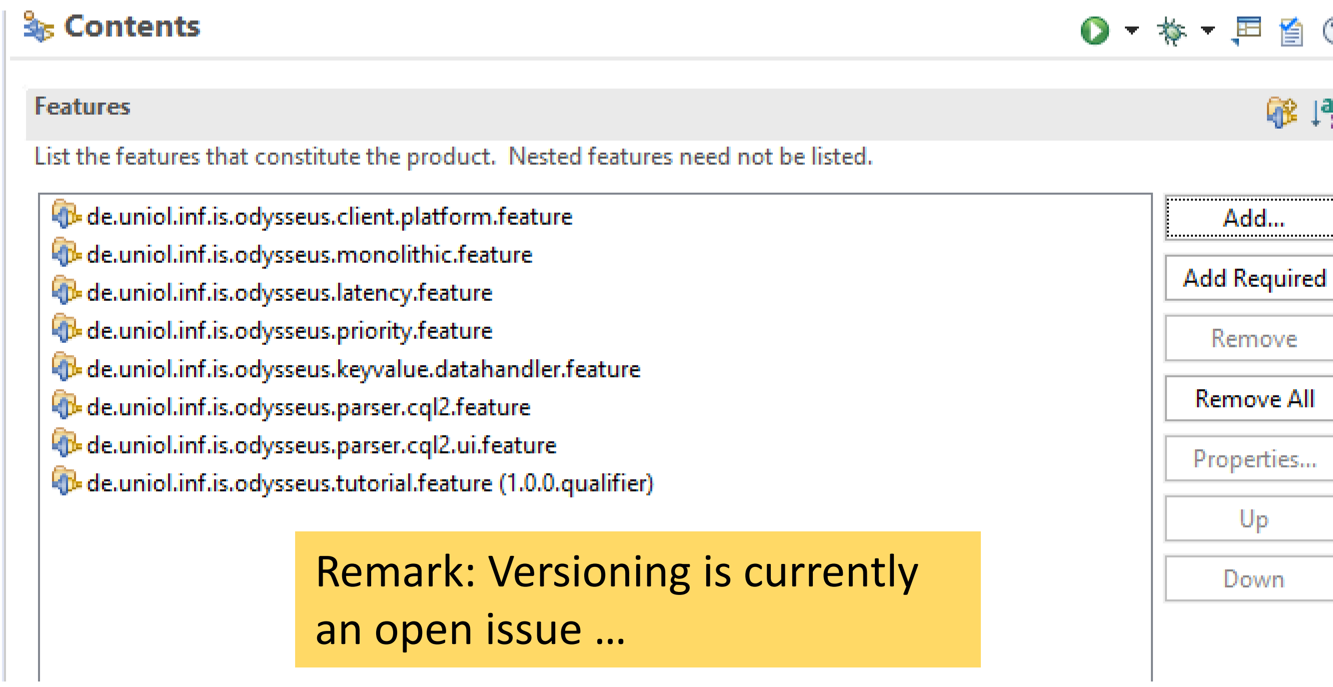
Enter an ID of a feature, or a plug-in contained by a feature ('?' and '*' are supported):

tutor

Matching items:

de.uniol.inf.is.odysseus.tutorial.feature (1.0.0.qualifier)

Feature 'de.uniol.inf.is.odysseus.tutorial.feature'.



Contents

Features

List the features that constitute the product. Nested features need not be listed.

de.uniol.inf.is.odysseus.client.platform.feature
de.uniol.inf.is.odysseus.monolithic.feature
de.uniol.inf.is.odysseus.latency.feature
de.uniol.inf.is.odysseus.priority.feature
de.uniol.inf.is.odysseus.keyvalue.datahandler.feature
de.uniol.inf.is.odysseus.parser.cql2.feature
de.uniol.inf.is.odysseus.parser.cql2.ui.feature
de.uniol.inf.is.odysseus.tutorial.feature (1.0.0.qualifier)

Remark: Versioning is currently an open issue …

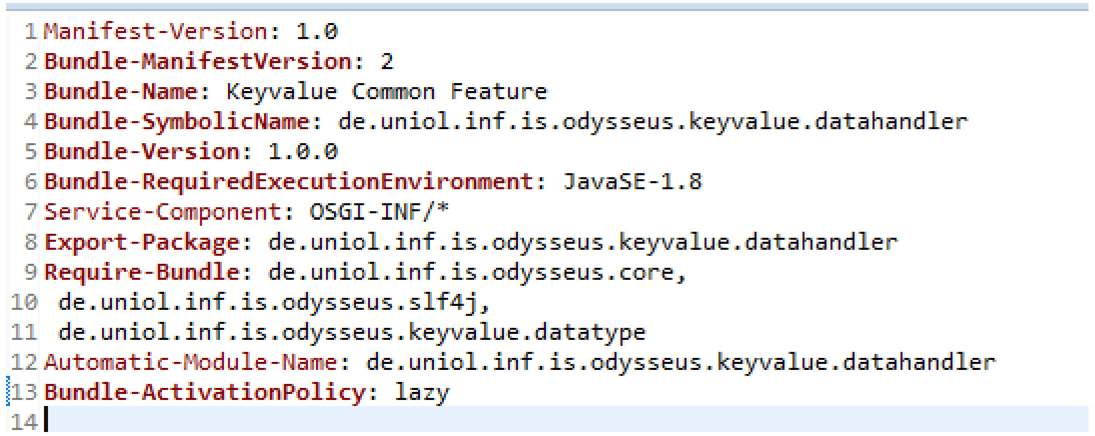
Tycho/Maven

- Maybe later …

Odysseus Development Best practices

- There should always be an **interface** (in Odysseus marked with an I, e.g. IExecutor)
- There should always be an **abstract class** (e.g. AbstractExecutor)
- For many cases we provide **default implementations**, typically named „Standard“ ... (e.g. StandardExecutor)

- In MANIFEST.MF use required plugins instead of Imported Packages
- Activation: lazy for bundles with services
- We often use: Service-Component: OSGI-INF/*
- → Attention when using eclipse wizzard to create new component: OSGI-INF/*,OSGI-INF/NewComponent.xml will not work



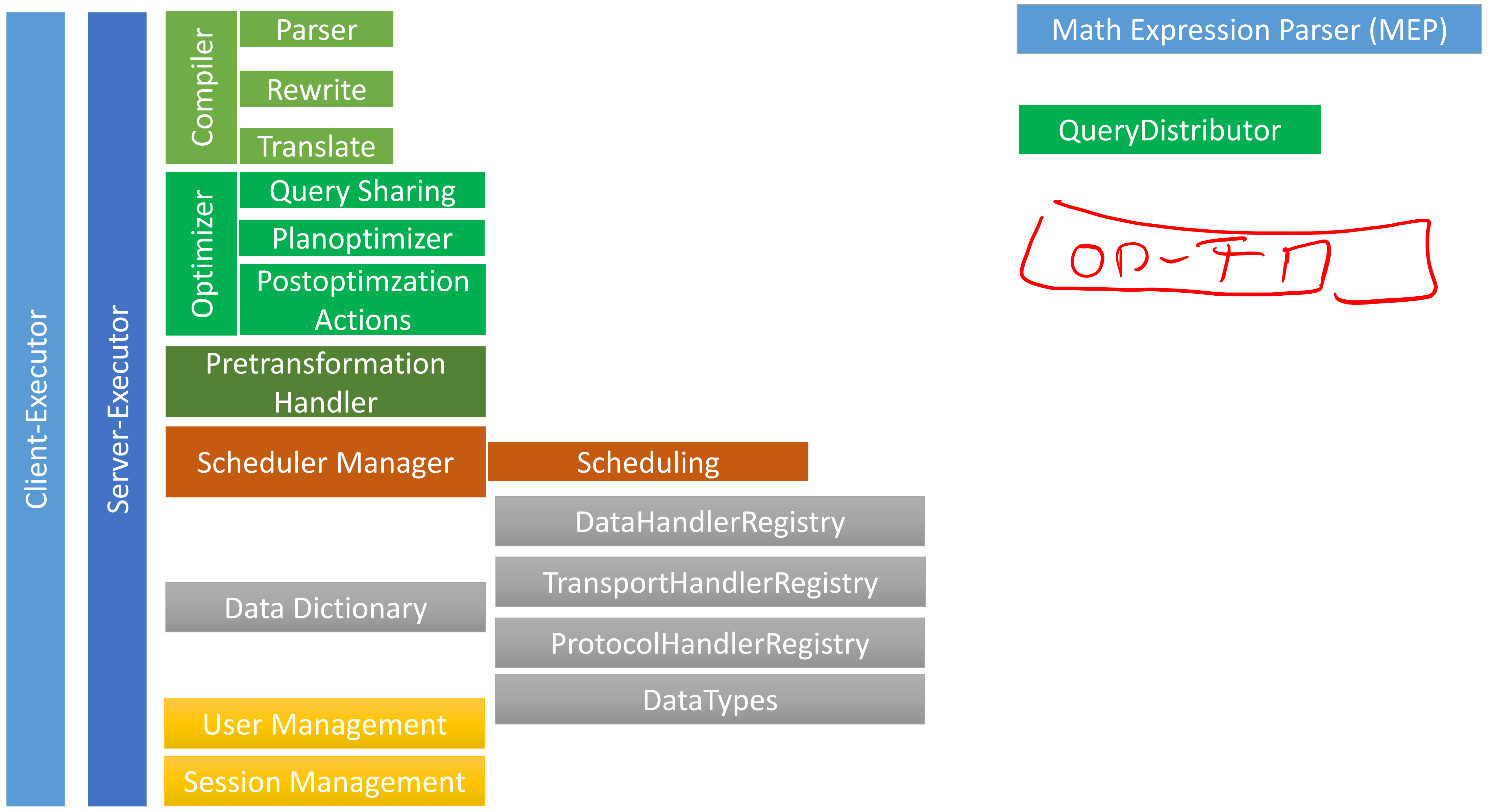
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Keyvalue Common Feature
Bundle-SymbolicName: de.uniol.inf.is.odysseus.keyvalue.datahandler
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: JavaSE-1.8
Service-Component: OSGI-INF/*
Export-Package: de.uniol.inf.is.odysseus.keyvalue.datahandler
Require-Bundle: de.uniol.inf.is.odysseus.core,
 de.uniol.inf.is.odysseus.slf4j,
 de.uniol.inf.is.odysseus.keyvalue.datatype
Automatic-Module-Name: de.uniol.inf.is.odysseus.keyvalue.datahandler
Bundle-ActivationPolicy: lazy

```

- Increase version number manually to allow updates via update site (we changed this recently … and due to some eclipse caching things will change this bac)

Extensions

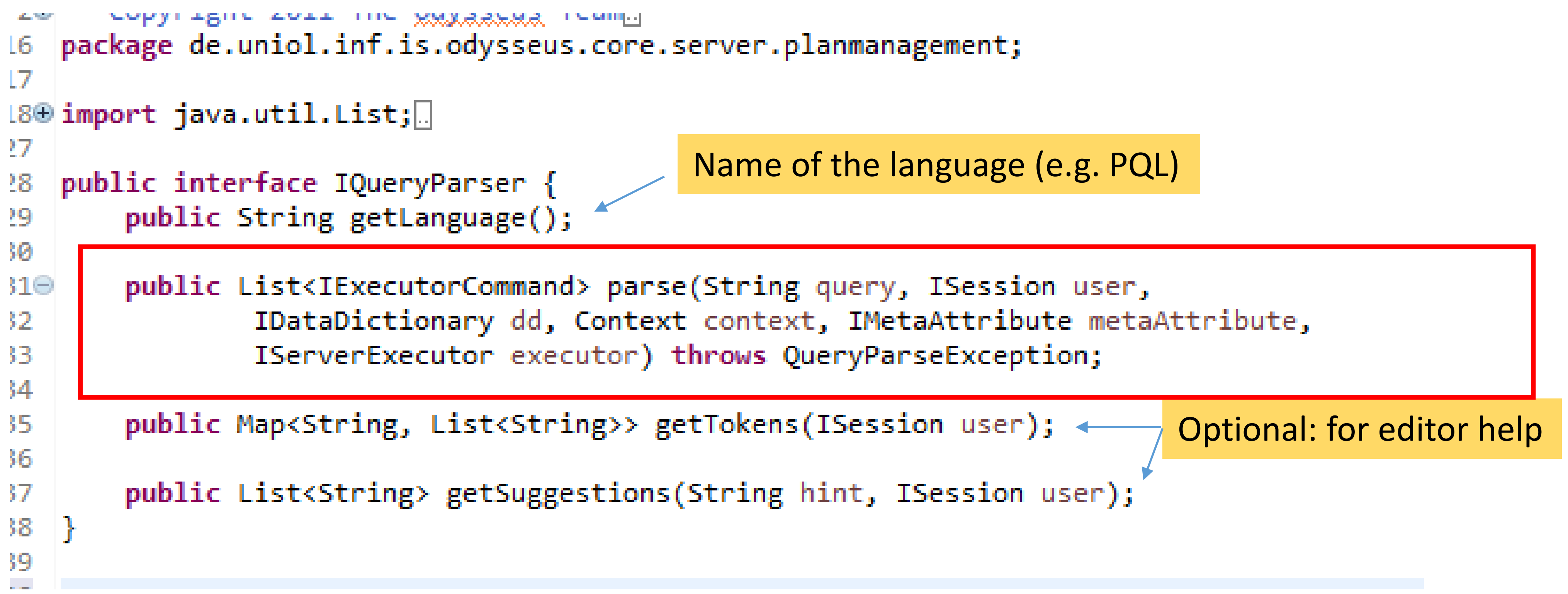
- Language extensions
  - Create a new language (not only for queries, could be a DSL for anything)
  - Create a new Odysseus Script Commands (#....)
  - Create a new Logical/PQL Operator
- Processing function extensions
  - Create new datatypes
  - Create new stream object types
    - Data Handler
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates
  - Create new aggregation functions
  - Create new operators

Architectural overview



Create a new language

- Need to implement IQueryParser Interface and register as OSGi-Service



```
package de.uniol.inf.is.odysseus.core.server.planmanagement;

import java.util.List;

public interface IQueryParser {
    public String getLanguage();

    public List<IExecutorCommand> parse(String query, ISession user,
            IDataDictionary dd, Context context, IMetaAttribute metaAttribute,
            IServerExecutor executor) throws QueryParseException;

    public Map<String, List<String>> getTokens(ISession user);

    public List<String> getSuggestions(String hint, ISession user);
}
```

- Methods are called from the framework
- Idea:
  - Translate Text (query) into sequence of **executor commands**
  - Execute each executor command on server side

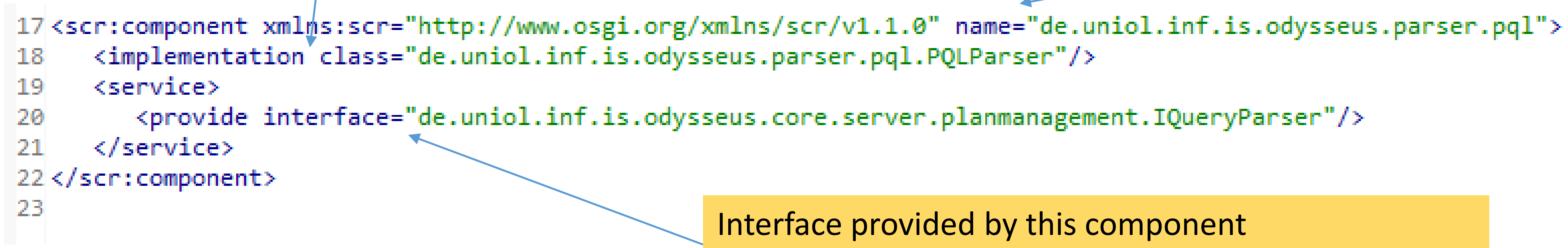
Create new Language: Executor Commands

- AddQueryCommand -ExecutorCommand:
  - Combine a parser, (query) text and some configurations
  - Is used by our „Mother Language“ Odysseus Script (e.g. #PARSER- and #ADDQUERY-Command)
  - In a query create a call to another compiler
- Our approach in (now) most cases: Translate to PQL Query
  - No need to handle with the underlying (logical) operator model
  - Easier to maintain
  - Can be easily moved around in the network
- Other ExecutorCommands for starting, stopping, pausing of queries or creating users, change user rights, …
- In our first CQL version: creating user, roles etc. was directly applied from the parser of the data dictionary → Now only the information is encapsulated by the Executor Commands

OSGi-Service (declarative service)

Class with implementation of this component

Unique id of the component



```xml
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="de.uniol.inf.is.odysseus.parser.pql">
   <implementation class="de.uniol.inf.is.odysseus.parser.pql.PQLParser"/>
   <service>
      <provide interface="de.uniol.inf.is.odysseus.core.server.planmanagement.IQueryParser"/>
   </service>
</scr:component>

```

Interface provided by this component
implementation class must implement this interface

As (OSGi-)default all service descriptions are placed in folder OSGI-INF

Manifest.MF must contain service reference (later more)

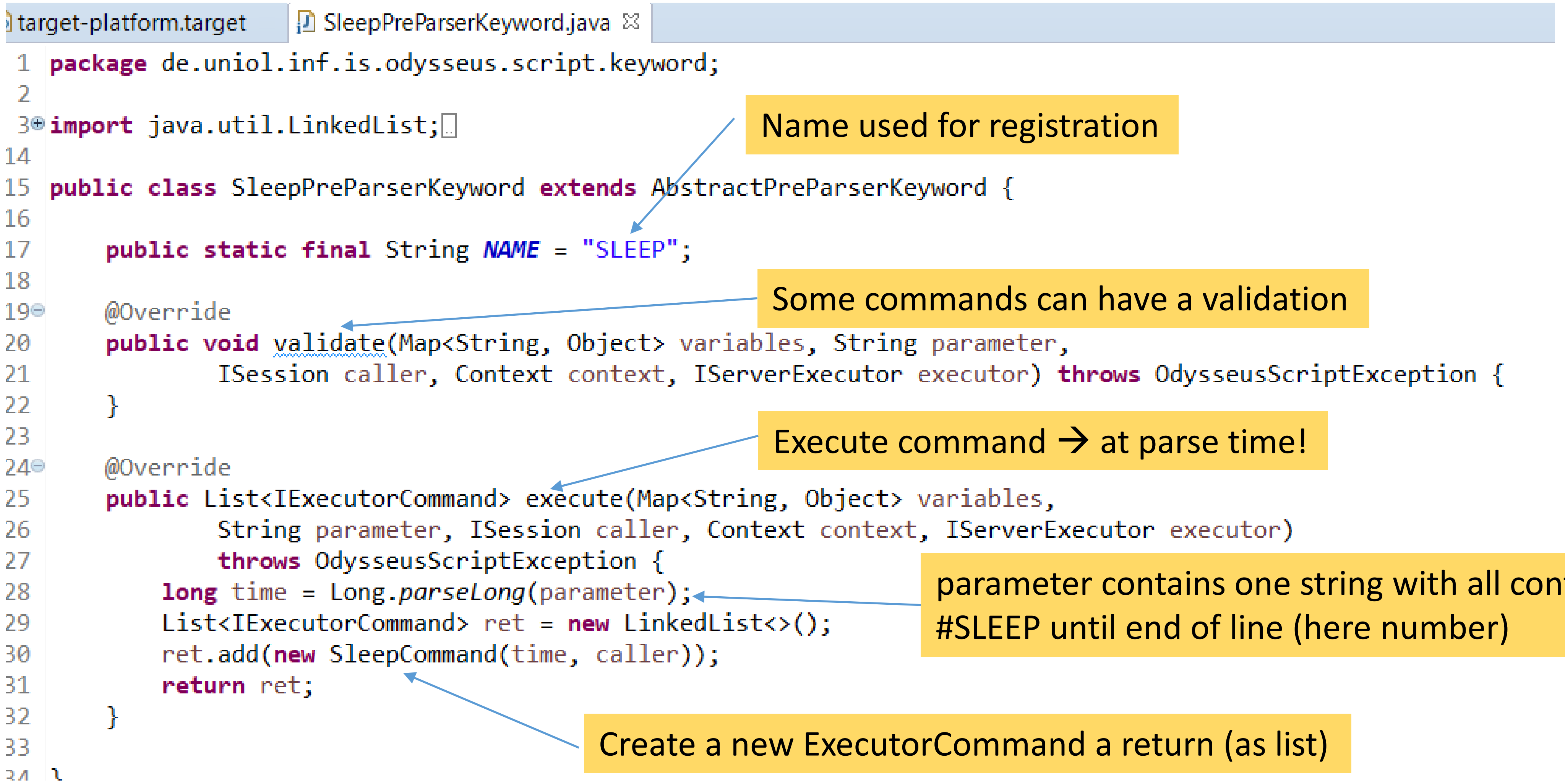
Extending Odysseus Script

- Odysseus script is our integration language
- Commands to define the next parser to be used (#PARSER)
- Commands to define a new query (#ADDQUERY)
- Commands to start and stop queries
- Some commands to control execution: #LOOP, #FOREACH, etc.
- Quite easy to integrate new commands

Example: Create a Sleep Command

- The current translation thread should do some work (e.g. start a query)
- After that the translation thread should wait some time (e.g. to allow query initialization)
- In Odysseus: `#SLEEP <timeInMs>`
- For this, we need to
  - Create an Odysseus Script Command (aka PreParserKeyword)
  - Create an ExecutorCommand

Sleep PreParserKeyword (Odysseus Script)



target-platform.target | SleepPreParserKeyword.java

```java
package de.uniol.inf.is.odysseus.script.keyword;

import java.util.LinkedList;

public class SleepPreParserKeyword extends AbstractPreParserKeyword {

    public static final String NAME = "SLEEP";

    @Override
    public void validate(Map<String, Object> variables, String parameter,
            ISession caller, Context context, IServerExecutor executor) throws OdysseusScriptException {
    }

    @Override
    public List<IExecutorCommand> execute(Map<String, Object> variables,
            String parameter, ISession caller, Context context, IServerExecutor executor)
            throws OdysseusScriptException {
        long time = Long.parseLong(parameter);
        List<IExecutorCommand> ret = new LinkedList<>();
        ret.add(new SleepCommand(time, caller));
        return ret;
    }

}
```

Name used for registration

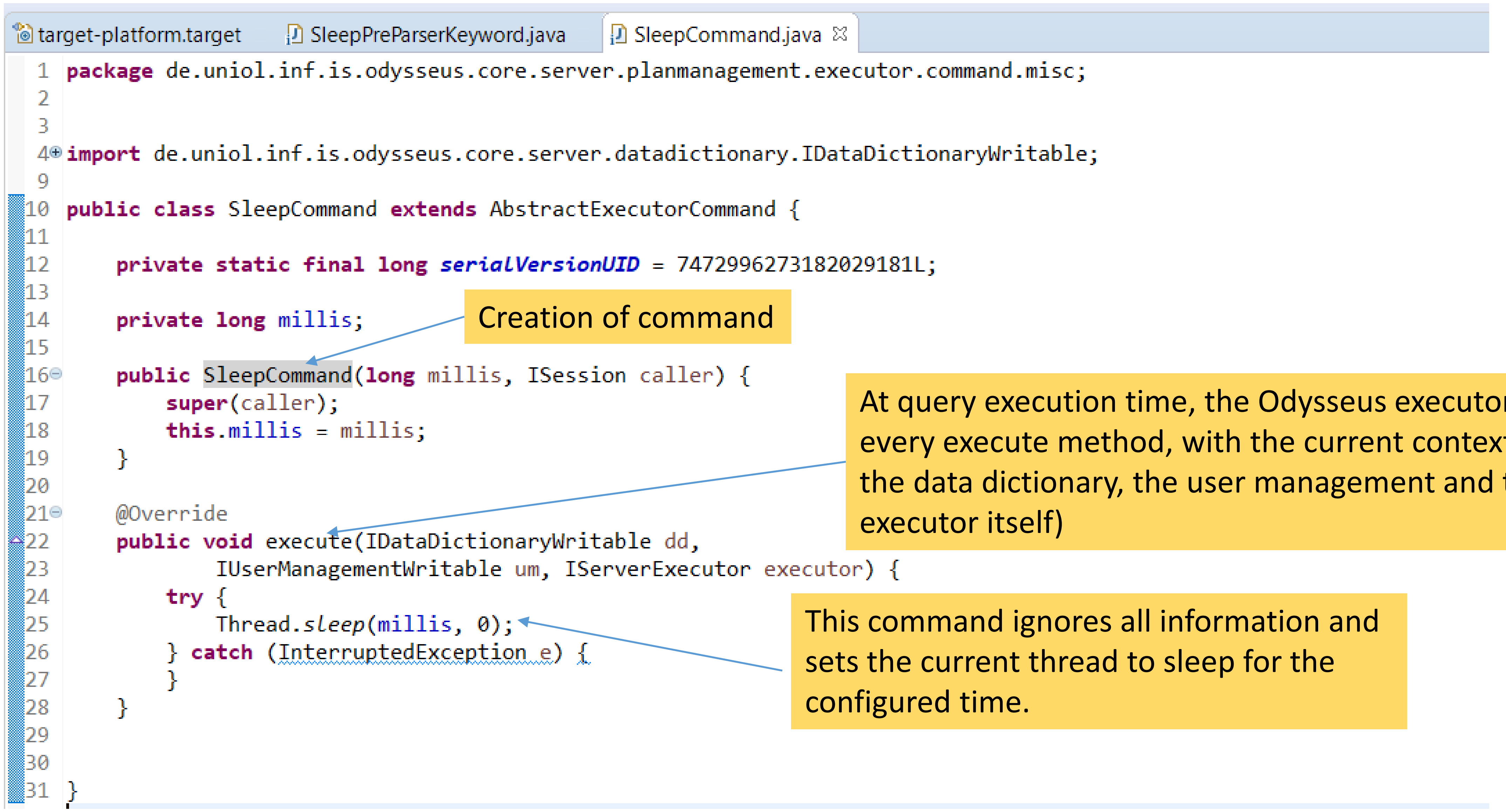
Some commands can have a validation

Execute command → at parse time!

parameter contains one string with all content after #SLEEP until end of line (here number)

Create a new ExecutorCommand a return (as list)

SleepCommand (Executor Command)



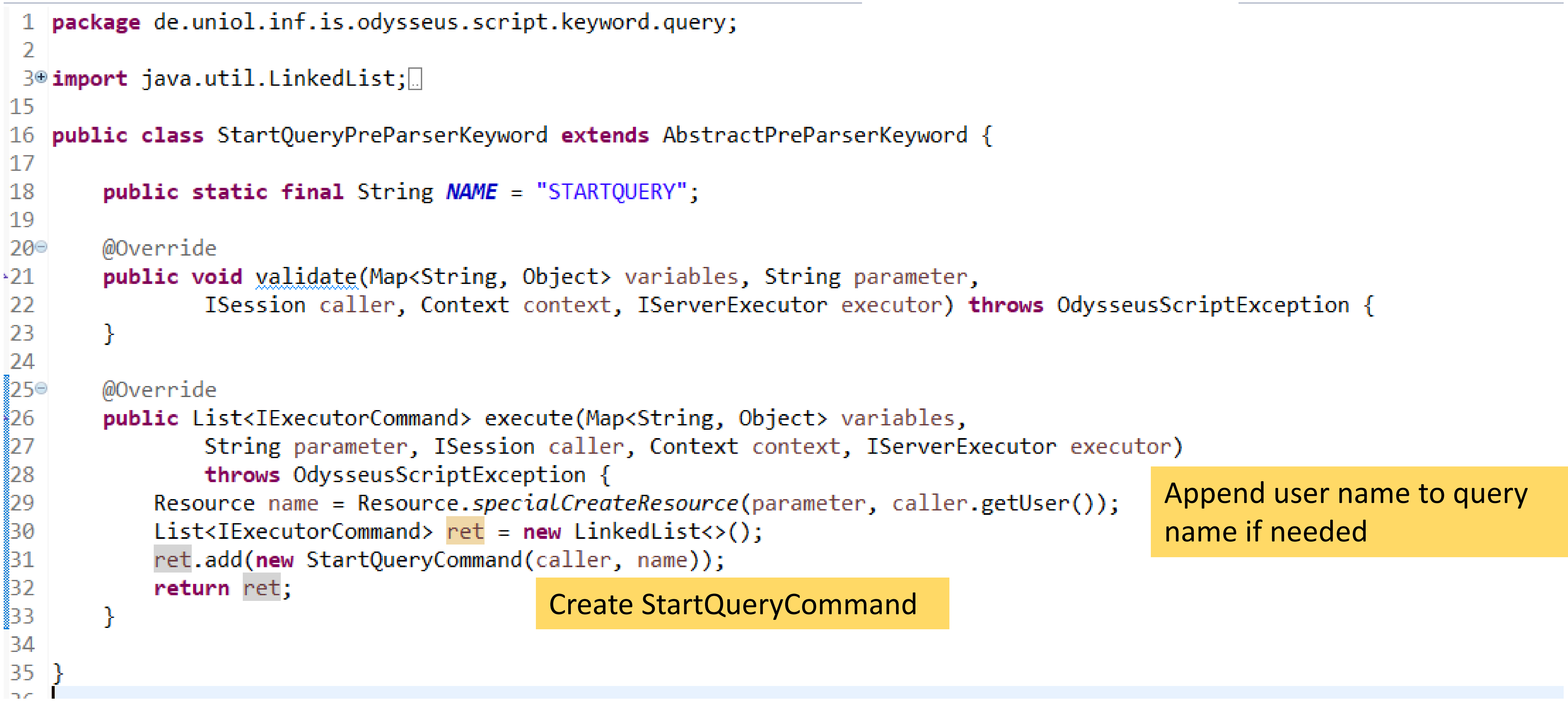
```java
package de.uniol.inf.is.odysseus.core.server.planmanagement.executor.command.misc;


import de.uniol.inf.is.odysseus.core.server.datadictionary.IDataDictionaryWritable;

public class SleepCommand extends AbstractExecutorCommand {

    private static final long serialVersionUID = 7472996273182029181L;

    private long millis;

    public SleepCommand(long millis, ISession caller) {
        super(caller);
        this.millis = millis;
    }

    @Override
    public void execute(IDataDictionaryWritable dd,
            IUserManagementWritable um, IServerExecutor executor) {
        try {
            Thread.sleep(millis, 0);
        } catch (InterruptedException e) {
        }
    }


}
```

Creation of command

At query execution time, the Odysseus executor calls every execute method, with the current context (i.e. the data dictionary, the user management and the executor itself)

This command ignores all information and sets the current thread to sleep for the configured time.

Start query command: #STARTQUERY <name>

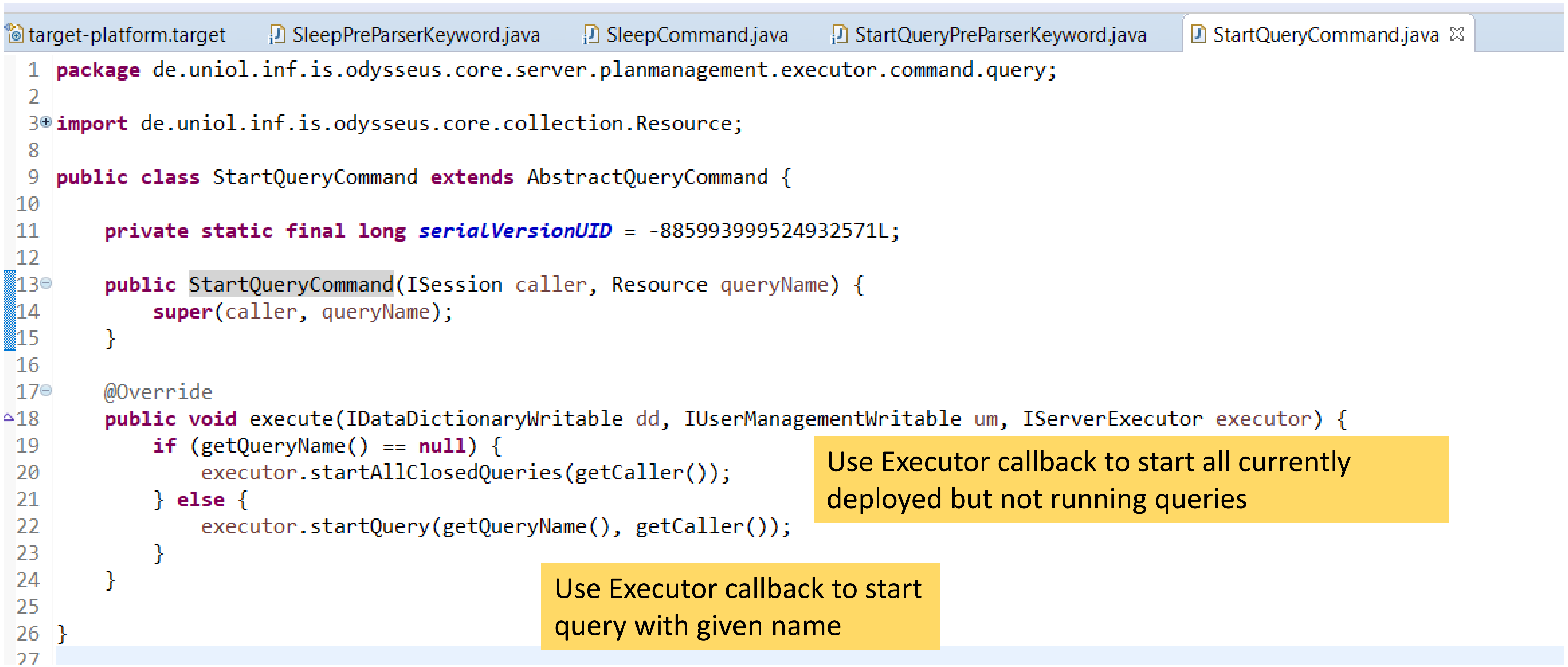


```
package de.uniol.inf.is.odysseus.script.keyword.query;

import java.util.LinkedList;

public class StartQueryPreParserKeyword extends AbstractPreParserKeyword {

    public static final String NAME = "STARTQUERY";

    @Override
    public void validate(Map<String, Object> variables, String parameter,
            ISession caller, Context context, IServerExecutor executor) throws OdysseusScriptException {
    }

    @Override
    public List<IExecutorCommand> execute(Map<String, Object> variables,
            String parameter, ISession caller, Context context, IServerExecutor executor)
            throws OdysseusScriptException {
        Resource name = Resource.specialCreateResource(parameter, caller.getUser());
        List<IExecutorCommand> ret = new LinkedList<>();
        ret.add(new StartQueryCommand(caller, name));
        return ret;
    }

}
```

Append user name to query name if needed

Create StartQueryCommand

Start query command (IExecutorCommand)



target-platform.target | SleepPreParserKeyword.java | SleepCommand.java | StartQueryPreParserKeyword.java | StartQueryCommand.java

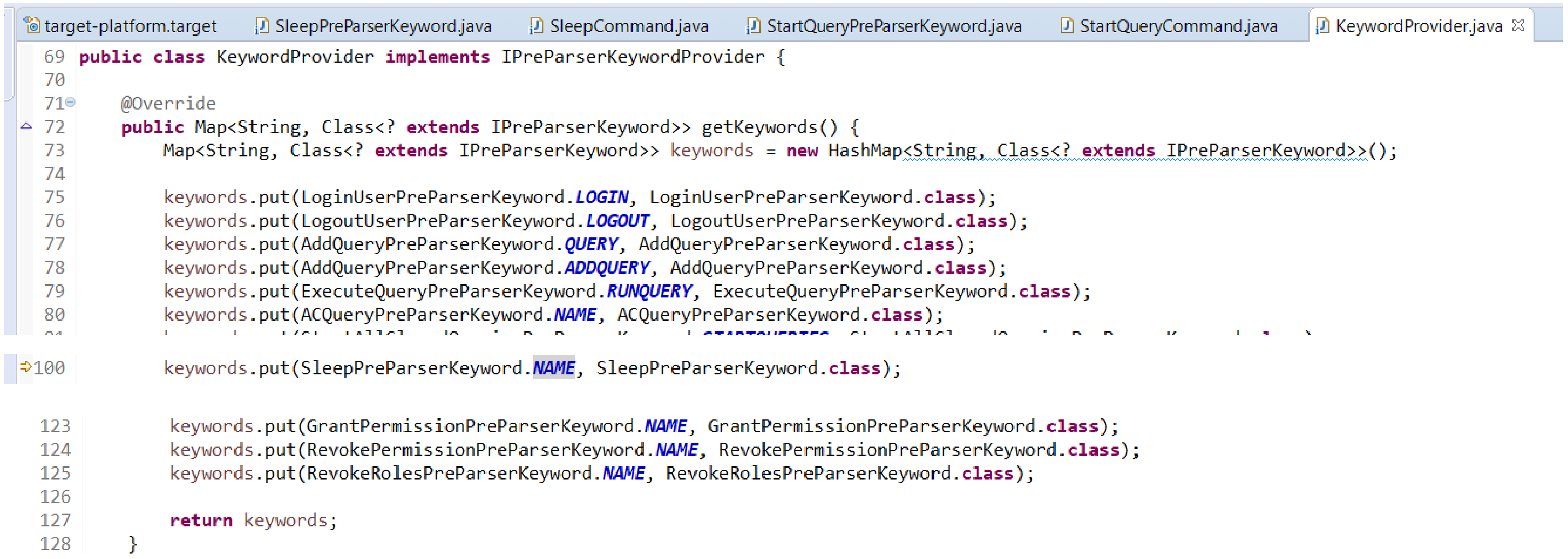
```java
package de.uniol.inf.is.odysseus.core.server.planmanagement.executor.command.query;

import de.uniol.inf.is.odysseus.core.collection.Resource;

public class StartQueryCommand extends AbstractQueryCommand {

    private static final long serialVersionUID = -8859939995249325711L;

    public StartQueryCommand(ISession caller, Resource queryName) {
        super(caller, queryName);
    }

    @Override
    public void execute(IDataDictionaryWritable dd, IUserManagementWritable um, IServerExecutor executor) {
        if (getQueryName() == null) {
            executor.startAllClosedQueries(getCaller());
        } else {
            executor.startQuery(getQueryName(), getCaller());
        }
    }

}
```

Use Executor callback to start all currently deployed but not running queries

Use Executor callback to start query with given name

Register commands inside Odysseus

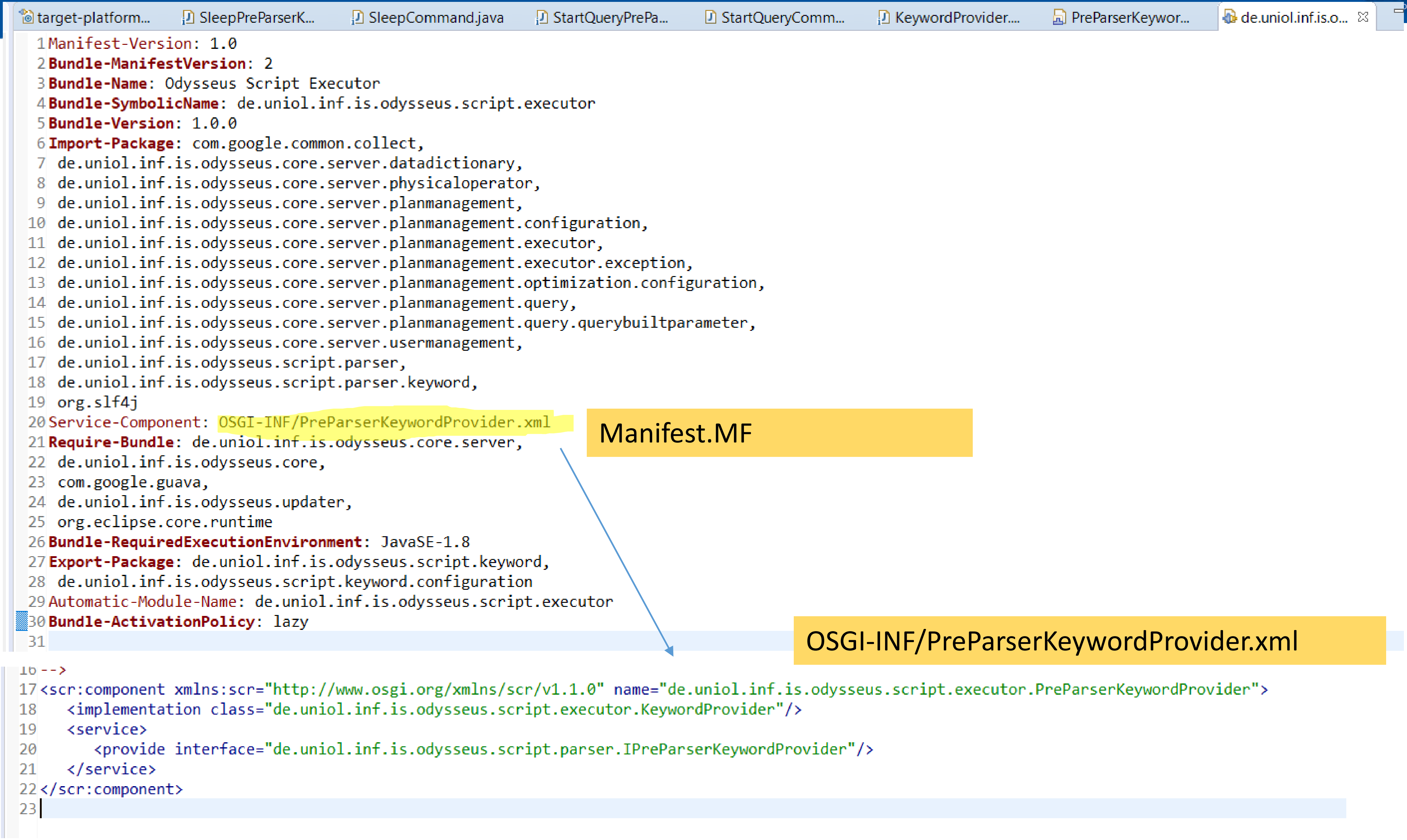
- Odysseus script commands: `implement IPreParserKeywordProvider` and returns `Map<String, Class (<? extends IPreParserKeyword>`



```java
public class KeywordProvider implements IPreParserKeywordProvider {

    @Override
    public Map<String, Class<? extends IPreParserKeyword>> getKeywords() {
        Map<String, Class<? extends IPreParserKeyword>> keywords = new HashMap<String, Class<? extends IPreParserKeyword>>();

        keywords.put(LoginUserPreParserKeyword.LOGIN, LoginUserPreParserKeyword.class);
        keywords.put(LogoutUserPreParserKeyword.LOGOUT, LogoutUserPreParserKeyword.class);
        keywords.put(AddQueryPreParserKeyword.QUERY, AddQueryPreParserKeyword.class);
        keywords.put(AddQueryPreParserKeyword.ADDQUERY, AddQueryPreParserKeyword.class);
        keywords.put(ExecuteQueryPreParserKeyword.RUNQUERY, ExecuteQueryPreParserKeyword.class);
        keywords.put(ACQueryPreParserKeyword.NAME, ACQueryPreParserKeyword.class);

        keywords.put(SleepPreParserKeyword.NAME, SleepPreParserKeyword.class);

        keywords.put(GrantPermissionPreParserKeyword.NAME, GrantPermissionPreParserKeyword.class);
        keywords.put(RevokePermissionPreParserKeyword.NAME, RevokePermissionPreParserKeyword.class);
        keywords.put(RevokeRolesPreParserKeyword.NAME, RevokeRolesPreParserKeyword.class);

        return keywords;
    }
```

This is quite typical in Odysseus to avoid the definition of a service for every single handler (here OdysseusScript keywords)

OSGi-Service



target-platform... | SleepPreParserK... | SleepCommand.java | StartQueryPrePa... | StartQueryComm... | KeywordProvider.... | PreParserKeywor... | de.uniol.inf.is.o...

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Odysseus Script Executor
Bundle-SymbolicName: de.uniol.inf.is.odysseus.script.executor
Bundle-Version: 1.0.0
Import-Package: com.google.common.collect,
 de.uniol.inf.is.odysseus.core.server.datadictionary,
 de.uniol.inf.is.odysseus.core.server.physicaloperator,
 de.uniol.inf.is.odysseus.core.server.planmanagement,
 de.uniol.inf.is.odysseus.core.server.planmanagement.configuration,
 de.uniol.inf.is.odysseus.core.server.planmanagement.executor,
 de.uniol.inf.is.odysseus.core.server.planmanagement.executor.exception,
 de.uniol.inf.is.odysseus.core.server.planmanagement.optimization.configuration,
 de.uniol.inf.is.odysseus.core.server.planmanagement.query,
 de.uniol.inf.is.odysseus.core.server.planmanagement.query.querybuiltparameter,
 de.uniol.inf.is.odysseus.core.server.usermanagement,
 de.uniol.inf.is.odysseus.script.parser,
 de.uniol.inf.is.odysseus.script.parser.keyword,
 org.slf4j
Service-Component: OSGI-INF/PreParserKeywordProvider.xml
Require-Bundle: de.uniol.inf.is.odysseus.core.server,
 de.uniol.inf.is.odysseus.core,
 com.google.guava,
 de.uniol.inf.is.odysseus.updater,
 org.eclipse.core.runtime
Bundle-RequiredExecutionEnvironment: JavaSE-1.8
Export-Package: de.uniol.inf.is.odysseus.script.keyword,
 de.uniol.inf.is.odysseus.script.keyword.configuration
Automatic-Module-Name: de.uniol.inf.is.odysseus.script.executor
Bundle-ActivationPolicy: lazy

```

Manifest.MF

OSGI-INF/PreParserKeywordProvider.xml

```
-->
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="de.uniol.inf.is.odysseus.script.executor.PreParserKeywordProvider">
   <implementation class="de.uniol.inf.is.odysseus.script.executor.KeywordProvider"/>
   <service>
      <provide interface="de.uniol.inf.is.odysseus.script.parser.IPreParserKeywordProvider"/>
   </service>
</scr:component>

```

New query processing functionality

- Until now: Extended (meta-)processing of queries
- Now: provide new query processing functionality
- If you want to
  - process a set of attribute value of a single input object: Write a new MEP-Function
  - process a set of attribute values of different input objects: Write a new aggregation function
  - do something else: Write a new operator
- Currently, we try to convert some operators to aggregation functions, too ...

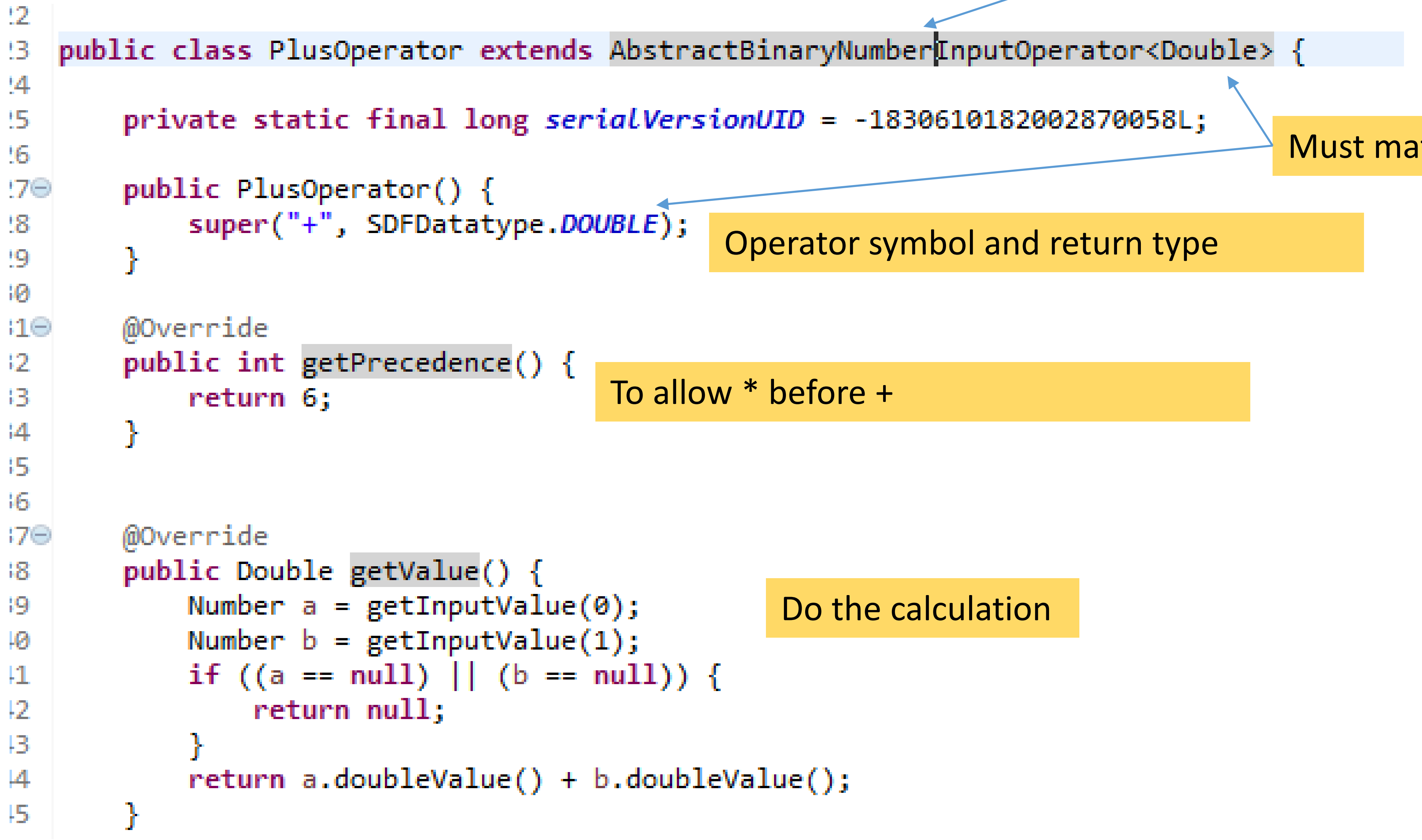
MEP

- MEP: Math Expression Parser
- A simple extendable framework to create and evaluate expressions (predicates) by MEP functions
- MEP is used as predicates (e.g. Select or Route) and expressions (e.g. MAP)
- MEP functions allow overloading, i.e. different implementations for different signature (e.g. equals on Strings is different to equals on Numbers)

- New operators
- New functions

Example: +-Operator

Base classes for many types available



```java
public class PlusOperator extends AbstractBinaryNumberInputOperator<Double> {

    private static final long serialVersionUID = -1830610182002870058L;

    public PlusOperator() {
        super("+", SDFDatatype.DOUBLE);
    }

    @Override
    public int getPrecedence() {
        return 6;
    }


    @Override
    public Double getValue() {
        Number a = getInputValue(0);
        Number b = getInputValue(1);
        if ((a == null) || (b == null)) {
            return null;
        }
        return a.doubleValue() + b.doubleValue();
    }
```

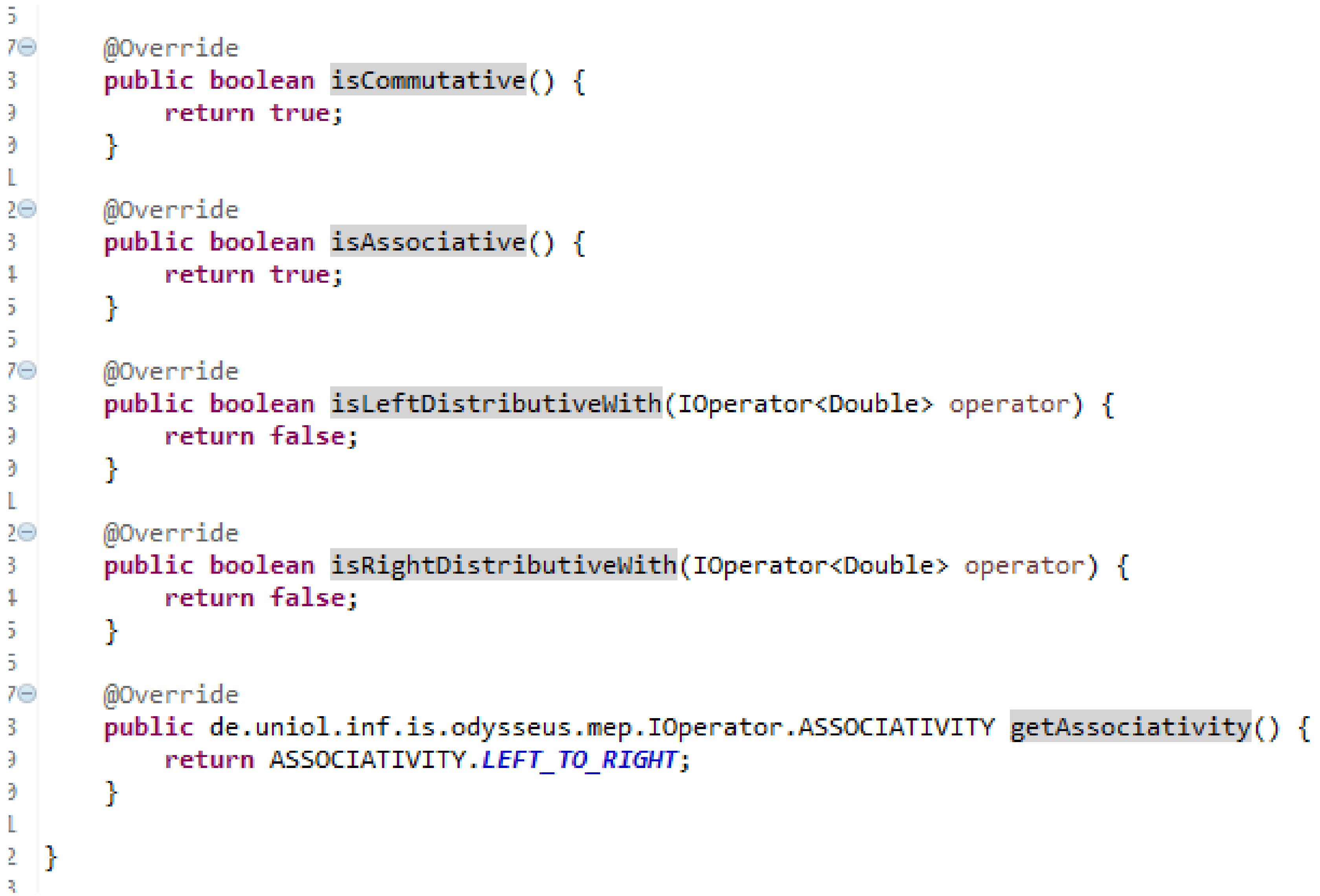
Must match!

Operator symbol and return type

To allow * before +

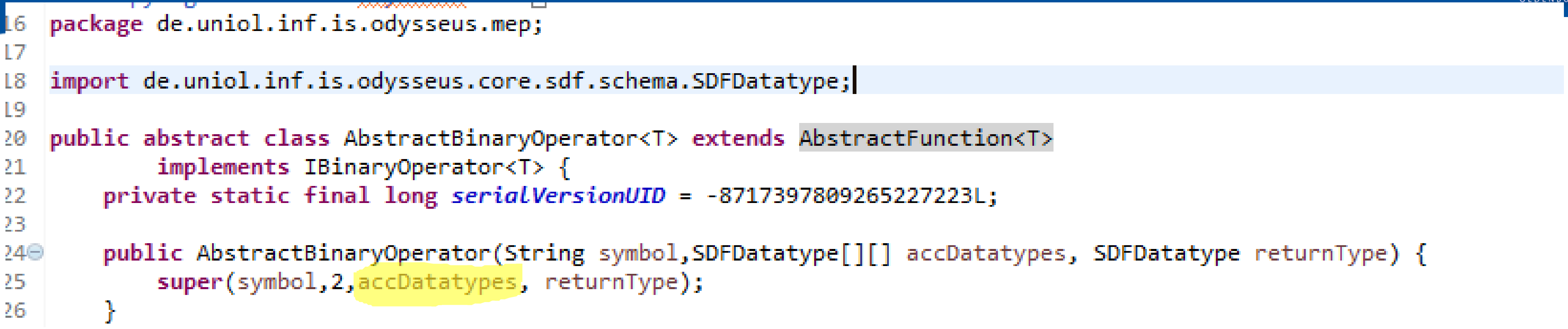
Do the calculation

Example: +-Operator: Additional information, used for optimization

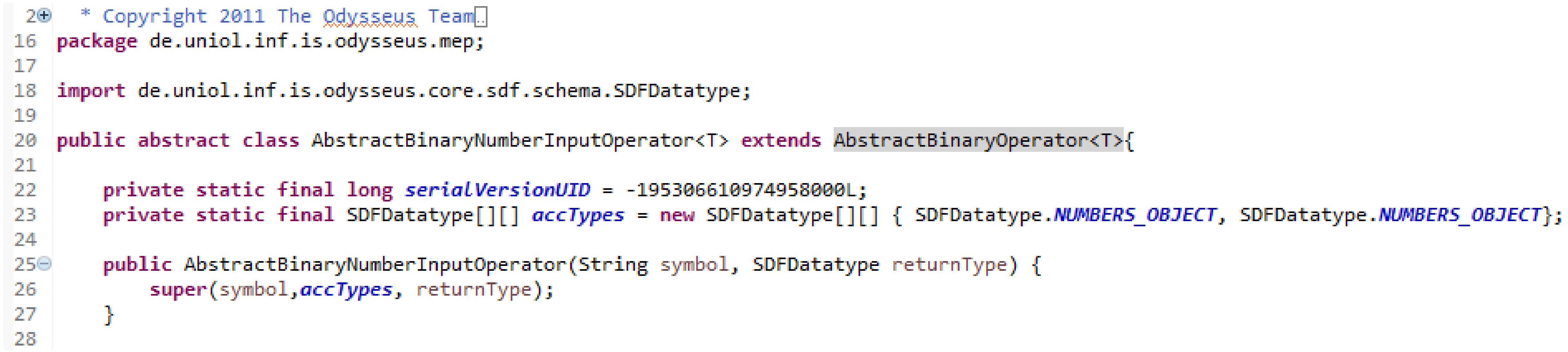


```
    @Override
    public boolean isCommutative() {
        return true;
    }

    @Override
    public boolean isAssociative() {
        return true;
    }

    @Override
    public boolean isLeftDistributiveWith(IOperator<Double> operator) {
        return false;
    }

    @Override
    public boolean isRightDistributiveWith(IOperator<Double> operator) {
        return false;
    }

    @Override
    public de.uniol.inf.is.odysseus.mep.IOperator.ASSOCIATIVITY getAssociativity() {
        return ASSOCIATIVITY.LEFT_TO_RIGHT;
    }

}
```

Bounding to types



```
package de.uniol.inf.is.odysseus.mep;

import de.uniol.inf.is.odysseus.core.sdf.schema.SDFDatatype;

public abstract class AbstractBinaryOperator<T> extends AbstractFunction<T>
        implements IBinaryOperator<T> {
    private static final long serialVersionUID = -8717397809265227223L;

    public AbstractBinaryOperator(String symbol,SDFDatatype[][] accDatatypes, SDFDatatype returnType) {
        super(symbol,2,accDatatypes, returnType);
    }
```



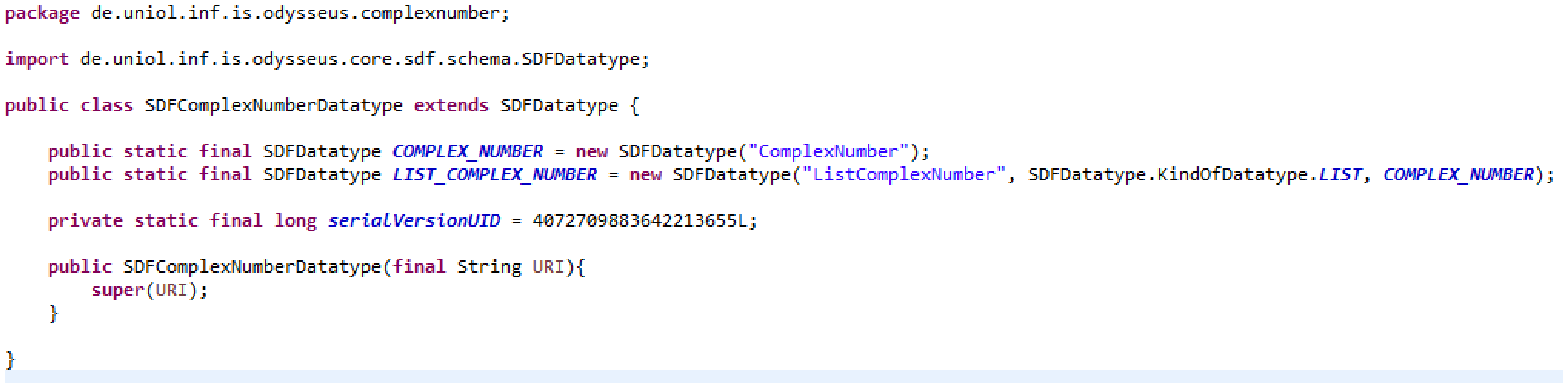
```
 * Copyright 2011 The Odysseus Team
package de.uniol.inf.is.odysseus.mep;

import de.uniol.inf.is.odysseus.core.sdf.schema.SDFDatatype;

public abstract class AbstractBinaryNumberInputOperator<T> extends AbstractBinaryOperator<T>{

    private static final long serialVersionUID = -1953066109749580001L;
    private static final SDFDatatype[][] accTypes = new SDFDatatype[][] { SDFDatatype.NUMBERS_OBJECT, SDFDatatype.NUMBERS_OBJECT};

    public AbstractBinaryNumberInputOperator(String symbol, SDFDatatype returnType) {
        super(symbol,accTypes, returnType);
    }

```

- Each function will be bound to the **first** occurence of a matching signature
- Here: Each <Number> + <Number> expression

Example

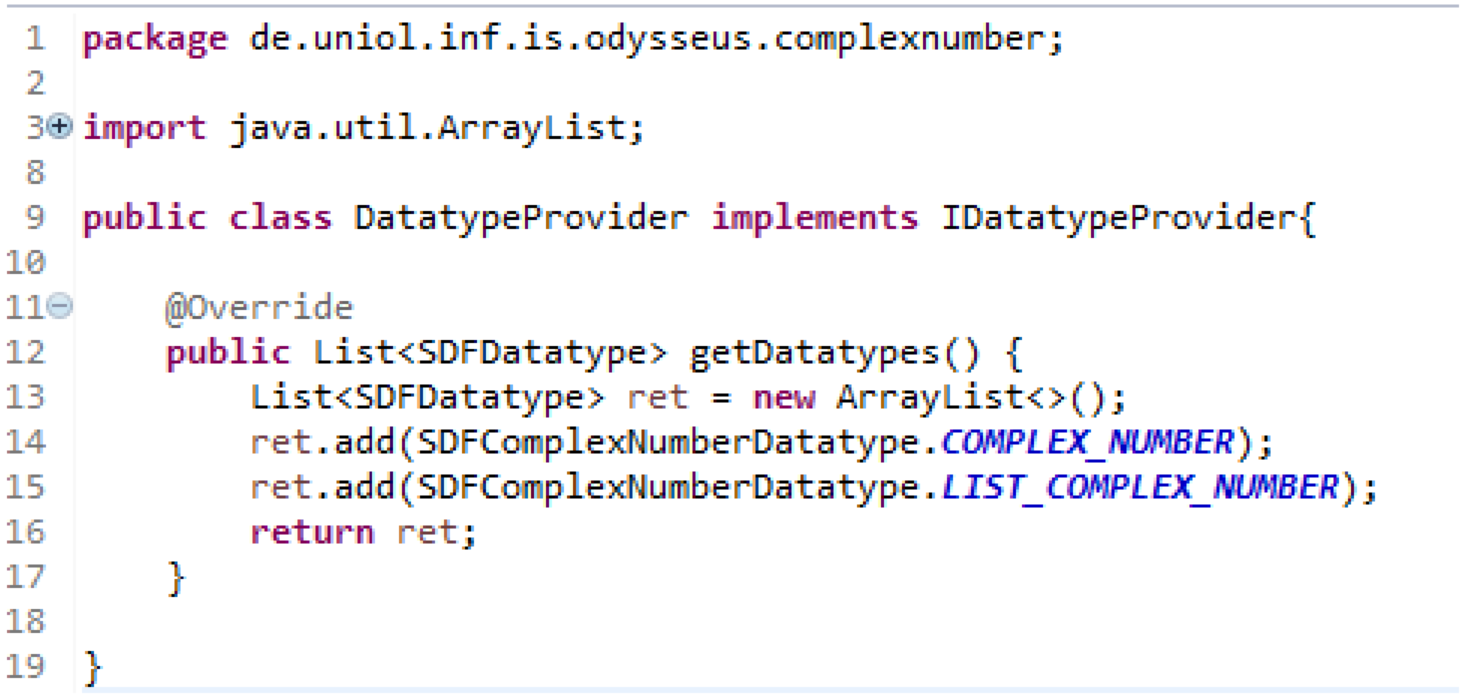
- Create a new data type
- Create functions on this data type
- Example: Complex number (de.uniol.inf.is.odysseus.complexnumber)

- Datatype register name

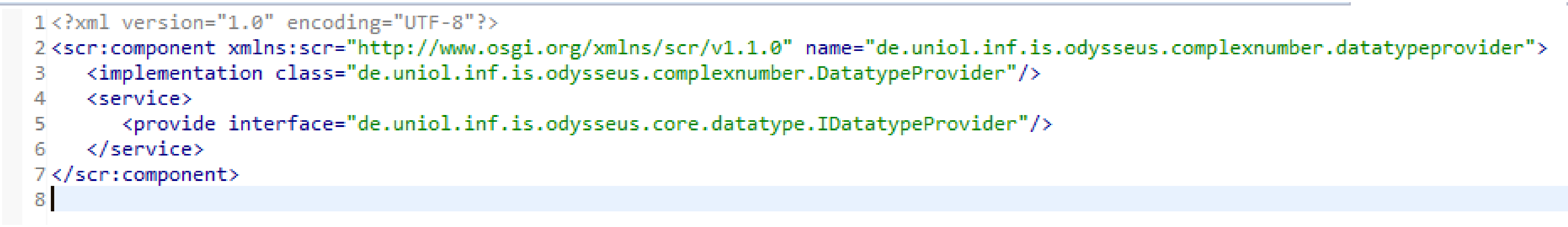


```java
package de.uniol.inf.is.odysseus.complexnumber;

import de.uniol.inf.is.odysseus.core.sdf.schema.SDFDatatype;

public class SDFComplexNumberDatatype extends SDFDatatype {

    public static final SDFDatatype COMPLEX_NUMBER = new SDFDatatype("ComplexNumber");
    public static final SDFDatatype LIST_COMPLEX_NUMBER = new SDFDatatype("ListComplexNumber", SDFDatatype.KindOfDatatype.LIST, COMPLEX_NUMBER);

    private static final long serialVersionUID = 4072709883642213655L;

    public SDFComplexNumberDatatype(final String URI){
        super(URI);
    }

}
```

Register new type

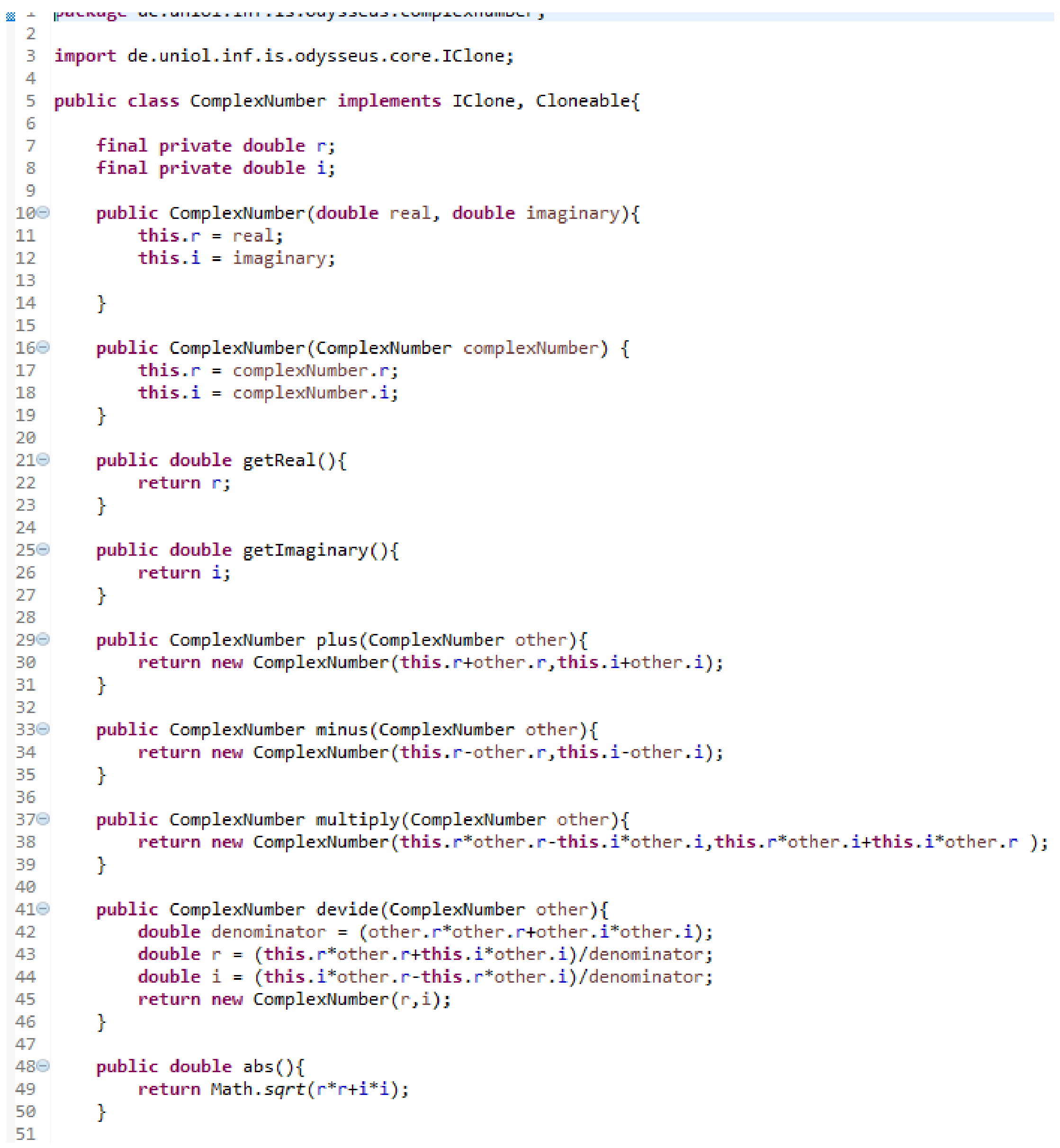


```java
package de.uniol.inf.is.odysseus.complexnumber;

import java.util.ArrayList;

public class DatatypeProvider implements IDatatypeProvider{

    @Override
    public List<SDFDatatype> getDatatypes() {
        List<SDFDatatype> ret = new ArrayList<>();
        ret.add(SDFComplexNumberDatatype.COMPLEX_NUMBER);
        ret.add(SDFComplexNumberDatatype.LIST_COMPLEX_NUMBER);
        return ret;
    }

}
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="de.uniol.inf.is.odysseus.complexnumber.datatypeprovider">
   <implementation class="de.uniol.inf.is.odysseus.complexnumber.DatatypeProvider"/>
   <service>
      <provide interface="de.uniol.inf.is.odysseus.core.datatype.IDatatypeProvider"/>
   </service>
</scr:component>

```

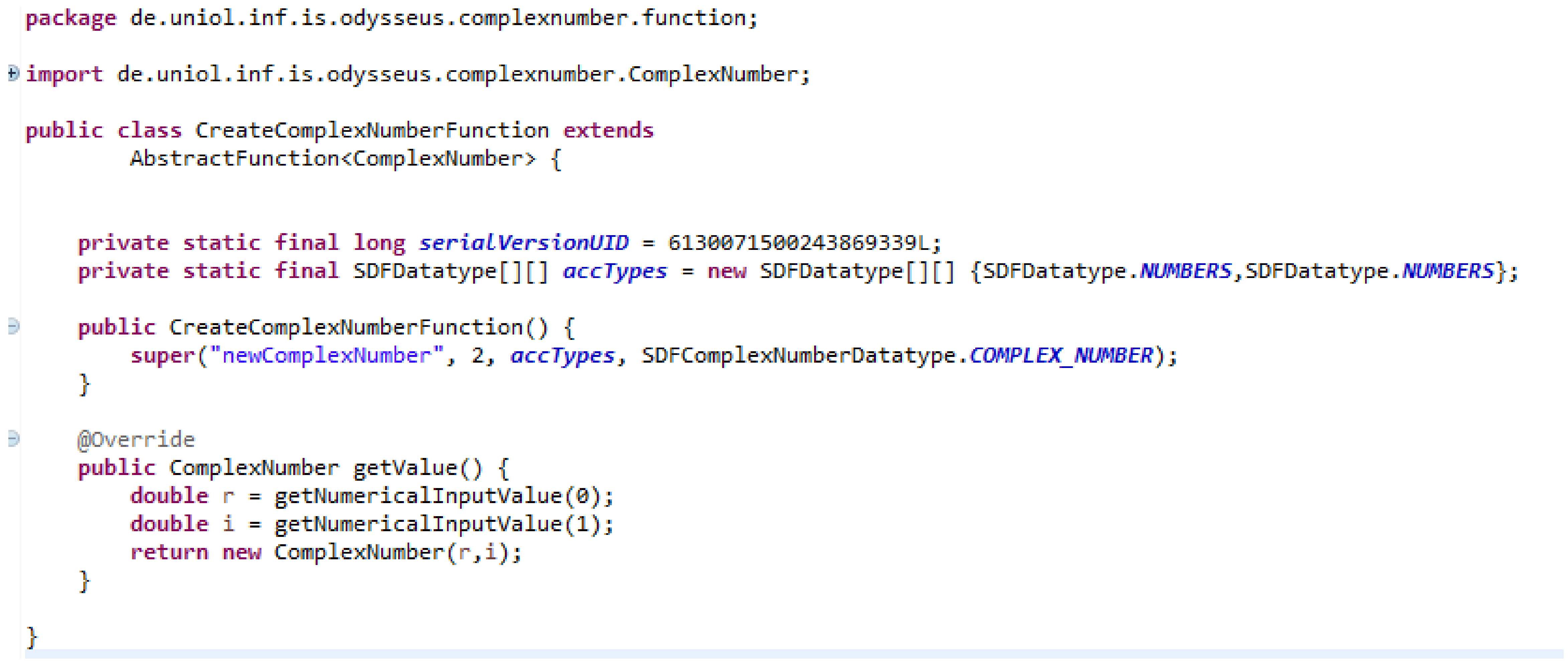
Create new Java class



```java
package de.uniol.inf.is.odysseus.complexnumber;

import de.uniol.inf.is.odysseus.core.IClone;

public class ComplexNumber implements IClone, Cloneable{

    final private double r;
    final private double i;

    public ComplexNumber(double real, double imaginary){
        this.r = real;
        this.i = imaginary;

    }

    public ComplexNumber(ComplexNumber complexNumber) {
        this.r = complexNumber.r;
        this.i = complexNumber.i;
    }

    public double getReal(){
        return r;
    }

    public double getImaginary(){
        return i;
    }

    public ComplexNumber plus(ComplexNumber other){
        return new ComplexNumber(this.r+other.r,this.i+other.i);
    }

    public ComplexNumber minus(ComplexNumber other){
        return new ComplexNumber(this.r-other.r,this.i-other.i);
    }

    public ComplexNumber multiply(ComplexNumber other){
        return new ComplexNumber(this.r*other.r-this.i*other.i,this.r*other.i+this.i*other.r );
    }

    public ComplexNumber devide(ComplexNumber other){
        double denominator = (other.r*other.r+other.i*other.i);
        double r = (this.r*other.r+this.i*other.i)/denominator;
        double i = (this.i*other.r-this.r*other.i)/denominator;
        return new ComplexNumber(r,i);
    }

    public double abs(){
        return Math.sqrt(r*r+i*i);
    }

```

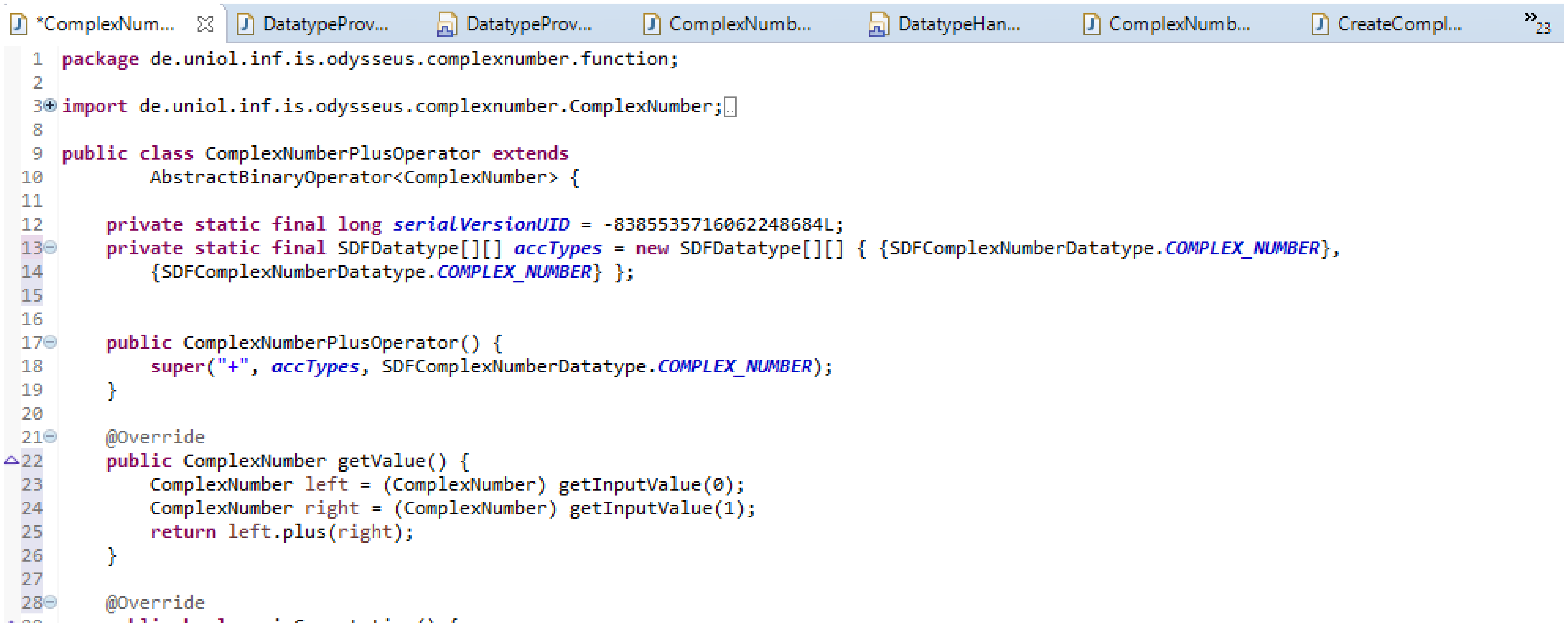
- To create a new object
- To clone from an object
- To handle operations
- …

Create Map functions and operators



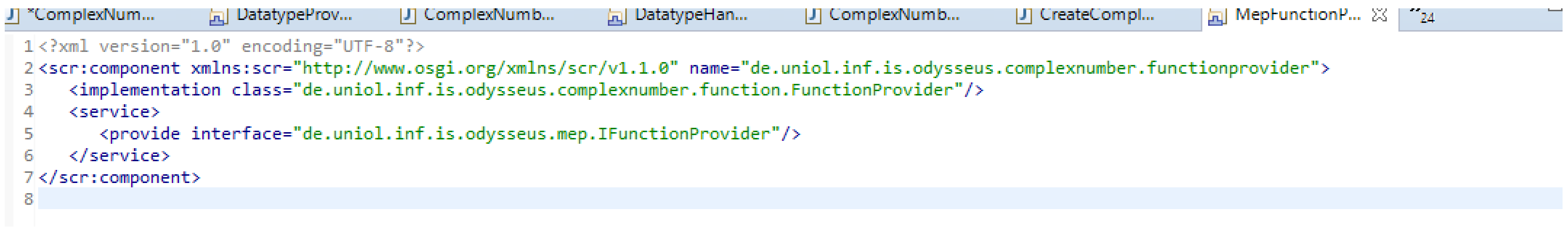
```java
package de.uniol.inf.is.odysseus.complexnumber.function;

import de.uniol.inf.is.odysseus.complexnumber.ComplexNumber;

public class CreateComplexNumberFunction extends
        AbstractFunction<ComplexNumber> {


    private static final long serialVersionUID = 6130071500243869339L;
    private static final SDFDatatype[][] accTypes = new SDFDatatype[][] {SDFDatatype.NUMBERS,SDFDatatype.NUMBERS};

    public CreateComplexNumberFunction() {
        super("newComplexNumber", 2, accTypes, SDFComplexNumberDatatype.COMPLEX_NUMBER);
    }

    @Override
    public ComplexNumber getValue() {
        double r = getNumericalInputValue(0);
        double i = getNumericalInputValue(1);
        return new ComplexNumber(r,i);
    }

}
```

Create Map functions and operators



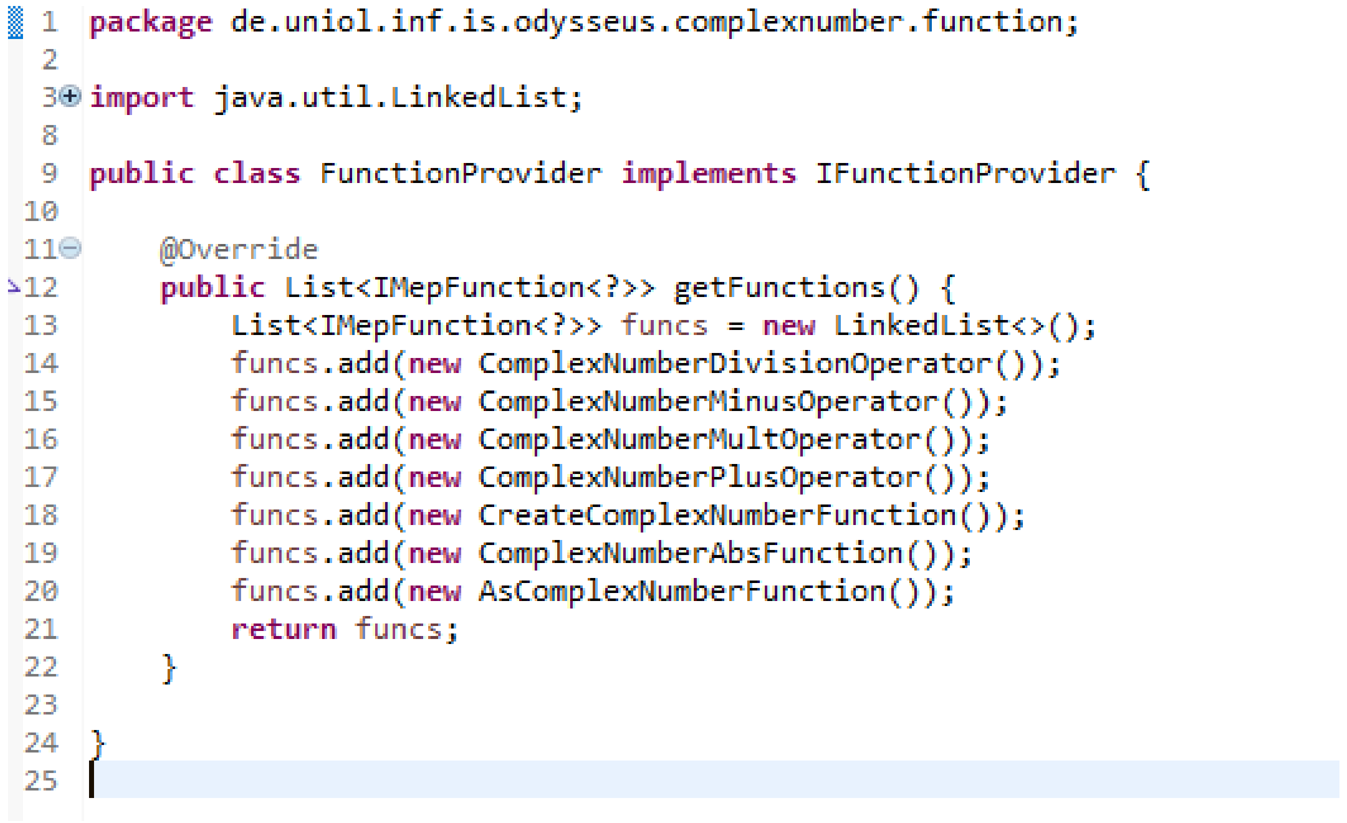
```java
package de.uniol.inf.is.odysseus.complexnumber.function;

import de.uniol.inf.is.odysseus.complexnumber.ComplexNumber;

public class ComplexNumberPlusOperator extends
        AbstractBinaryOperator<ComplexNumber> {

    private static final long serialVersionUID = -8385535716062248684L;
    private static final SDFDatatype[][] accTypes = new SDFDatatype[][] { {SDFComplexNumberDatatype.COMPLEX_NUMBER},
        {SDFComplexNumberDatatype.COMPLEX_NUMBER} };


    public ComplexNumberPlusOperator() {
        super("+", accTypes, SDFComplexNumberDatatype.COMPLEX_NUMBER);
    }

    @Override
    public ComplexNumber getValue() {
        ComplexNumber left = (ComplexNumber) getInputValue(0);
        ComplexNumber right = (ComplexNumber) getInputValue(1);
        return left.plus(right);
    }

    @Override
```

Registration of MEP-Functions: as usual



*ComplexNum... DatatypeProv... ComplexNumb... DatatypeHan... ComplexNumb... CreateCompl... MepFunctionP... »24

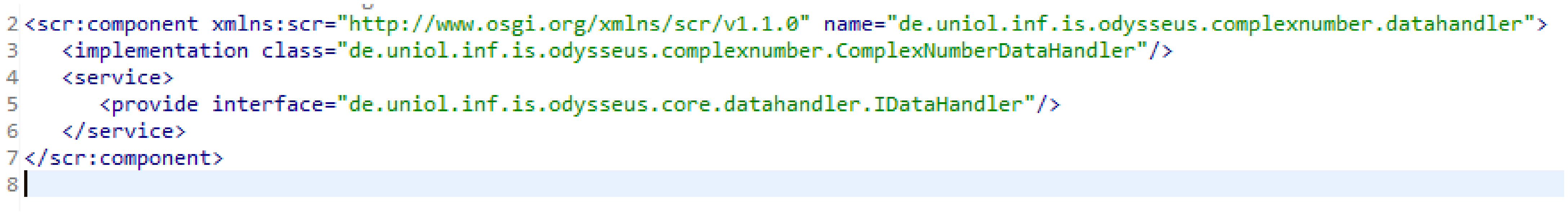
```xml
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="de.uniol.inf.is.odysseus.complexnumber.functionprovider">
   <implementation class="de.uniol.inf.is.odysseus.complexnumber.function.FunctionProvider"/>
   <service>
      <provide interface="de.uniol.inf.is.odysseus.mep.IFunctionProvider"/>
   </service>
</scr:component>

```



```java
package de.uniol.inf.is.odysseus.complexnumber.function;

import java.util.LinkedList;

public class FunctionProvider implements IFunctionProvider {

    @Override
    public List<IMepFunction<?>> getFunctions() {
        List<IMepFunction<?>> funcs = new LinkedList<>();
        funcs.add(new ComplexNumberDivisionOperator());
        funcs.add(new ComplexNumberMinusOperator());
        funcs.add(new ComplexNumberMultOperator());
        funcs.add(new ComplexNumberPlusOperator());
        funcs.add(new CreateComplexNumberFunction());
        funcs.add(new ComplexNumberAbsFunction());
        funcs.add(new AsComplexNumberFunction());
        return funcs;
    }

}

```

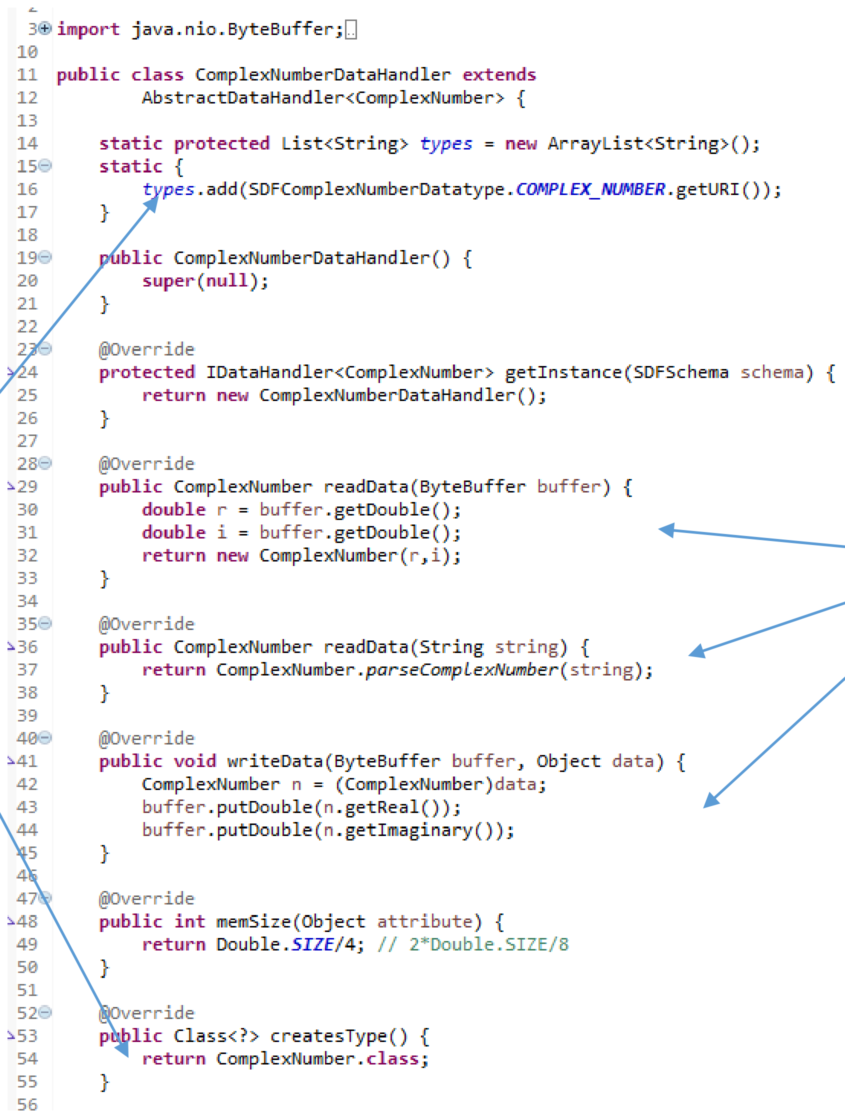
Read and write new Data type

- The following is only necessary, if you want to use the new data type with sources and sinks
- Create for the new data type a data type handler



```xml
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="de.uniol.inf.is.odysseus.complexnumber.datahandler">
   <implementation class="de.uniol.inf.is.odysseus.complexnumber.ComplexNumberDataHandler"/>
   <service>
      <provide interface="de.uniol.inf.is.odysseus.core.datahandler.IDataHandler"/>
   </service>
</scr:component>
```

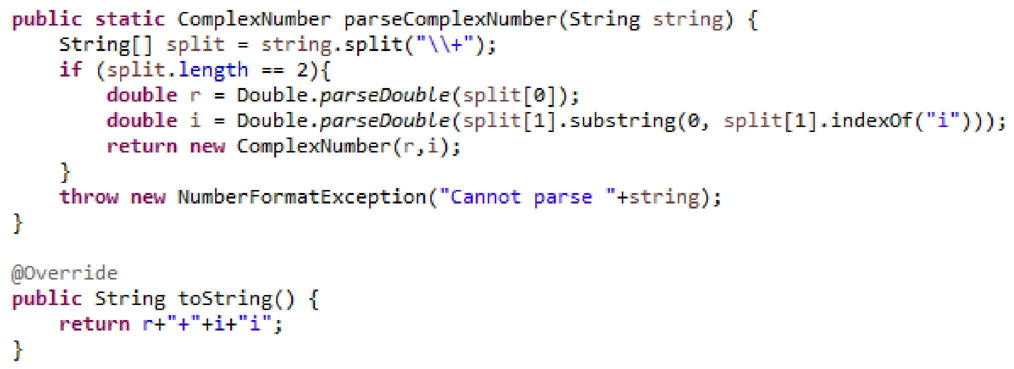
DataHandler



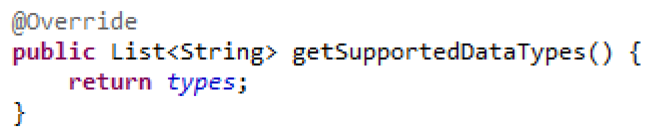
```java
import java.nio.ByteBuffer;

public class ComplexNumberDataHandler extends
        AbstractDataHandler<ComplexNumber> {

    static protected List<String> types = new ArrayList<String>();
    static {
        types.add(SDFComplexNumberDatatype.COMPLEX_NUMBER.getURI());
    }

    public ComplexNumberDataHandler() {
        super(null);
    }

    @Override
    protected IDataHandler<ComplexNumber> getInstance(SDFSchema schema) {
        return new ComplexNumberDataHandler();
    }

    @Override
    public ComplexNumber readData(ByteBuffer buffer) {
        double r = buffer.getDouble();
        double i = buffer.getDouble();
        return new ComplexNumber(r,i);
    }

    @Override
    public ComplexNumber readData(String string) {
        return ComplexNumber.parseComplexNumber(string);
    }

    @Override
    public void writeData(ByteBuffer buffer, Object data) {
        ComplexNumber n = (ComplexNumber)data;
        buffer.putDouble(n.getReal());
        buffer.putDouble(n.getImaginary());
    }

    @Override
    public int memSize(Object attribute) {
        return Double.SIZE/4; // 2*Double.SIZE/8
    }

    @Override
    public Class<?> createsType() {
        return ComplexNumber.class;
    }

```

Binding of SDFTypes and JavaTypes

Conversion from/to Buffer and String



```java
public static ComplexNumber parseComplexNumber(String string) {
    String[] split = string.split("\\+");
    if (split.length == 2){
        double r = Double.parseDouble(split[0]);
        double i = Double.parseDouble(split[1].substring(0, split[1].indexOf("i")));
        return new ComplexNumber(r,i);
    }
    throw new NumberFormatException("Cannot parse "+string);
}

@Override
public String toString() {
    return r+"+"+i+"i";
}
```



```java
@Override
public List<String> getSupportedDataTypes() {
    return types;
}
```

Extensions

- Language extensions
  - Create a new language ✓
  - Create a new Odysseus Script Command (#....) ✓
  - Create a new PQL Operator
- Processing function extensions
  - Create new datatypes ✓
  - Create new stream object types
    - Data Handler (✓)
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates ✓
  - Create new aggregationsfunctions
  - Create new operators
- Create new schedulers and scheduling strategies
- ...

Write a new aggregation function

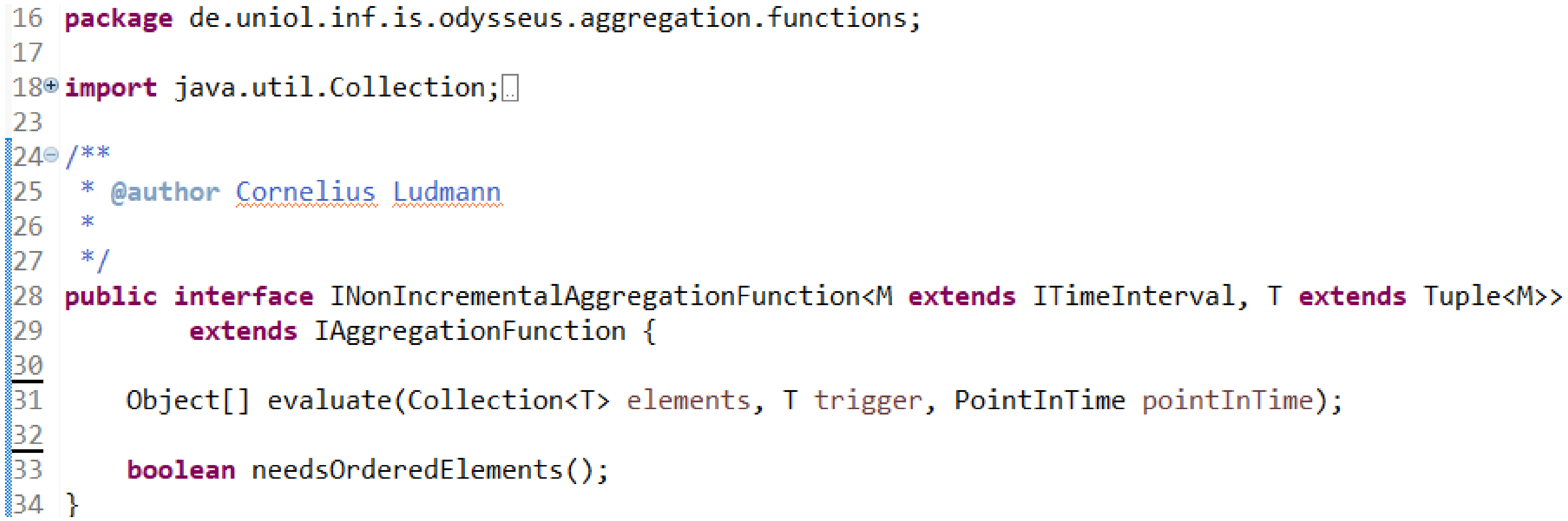
- Odyssseus has two operators for tuple (position) based aggregation
- Aggregate:
  - Works with partial aggregates
  - creates output, when a new aggregate is calculated
  - assures temporal window logic

- Aggregation:
  - Works with (incremental) aggregation function
  - is much faster
  - is much easier extendable
  - temporal correctness for every case must be prooven
  - creates output without endtime stamps → no need to wait for the next element
  - Together with 1-element-window same timestamps as Aggregate

Create a new aggregation function

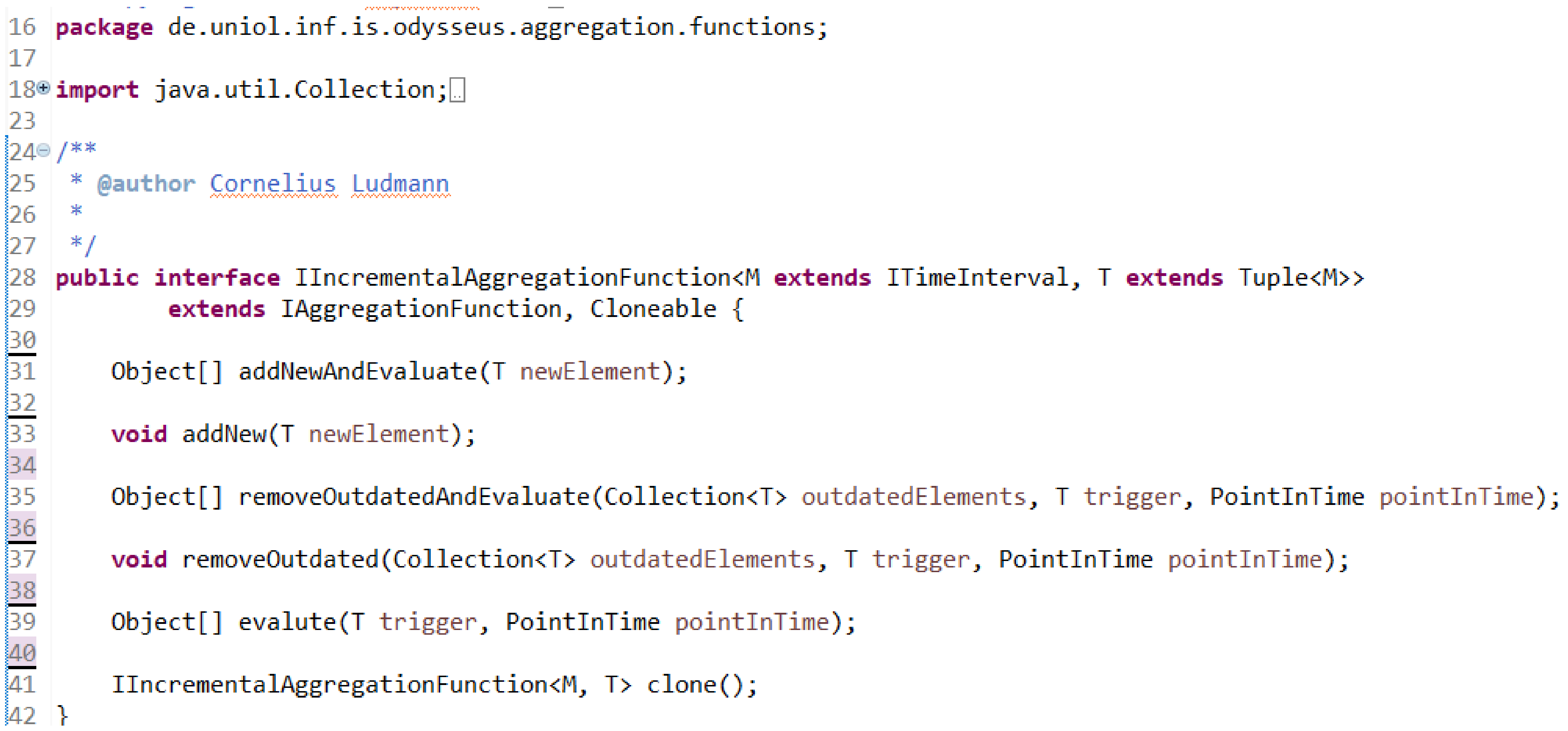
- The are two kinds of aggregation functions
  - Incremental function (IIncrementalAggregationFunction):
    - This function gets informed, when an element gets valid (enters the window) or gets invalid (leaves) the current window
    - The function has a state and returns the current aggregation state on demand
  - Nonincremental function (INonIncrementalAggregationFunction):
    - This function gets the set of all current valid elements and calculates for this the aggregation
    - The function does not have a state
- Additionally, there needs to be a factory, that creates an IAggregationFunction from a set of options
- Typically, combine both interfaces in one implementation
- Abstract base classes for Incremental and NonIncremental

Non incremental



```
package de.uniol.inf.is.odysseus.aggregation.functions;

import java.util.Collection;

/**
 * @author Cornelius Ludmann
 *
 */
public interface INonIncrementalAggregationFunction<M extends ITimeInterval, T extends Tuple<M>>
        extends IAggregationFunction {

    Object[] evaluate(Collection<T> elements, T trigger, PointInTime pointInTime);

    boolean needsOrderedElements();
}
```

Incremental



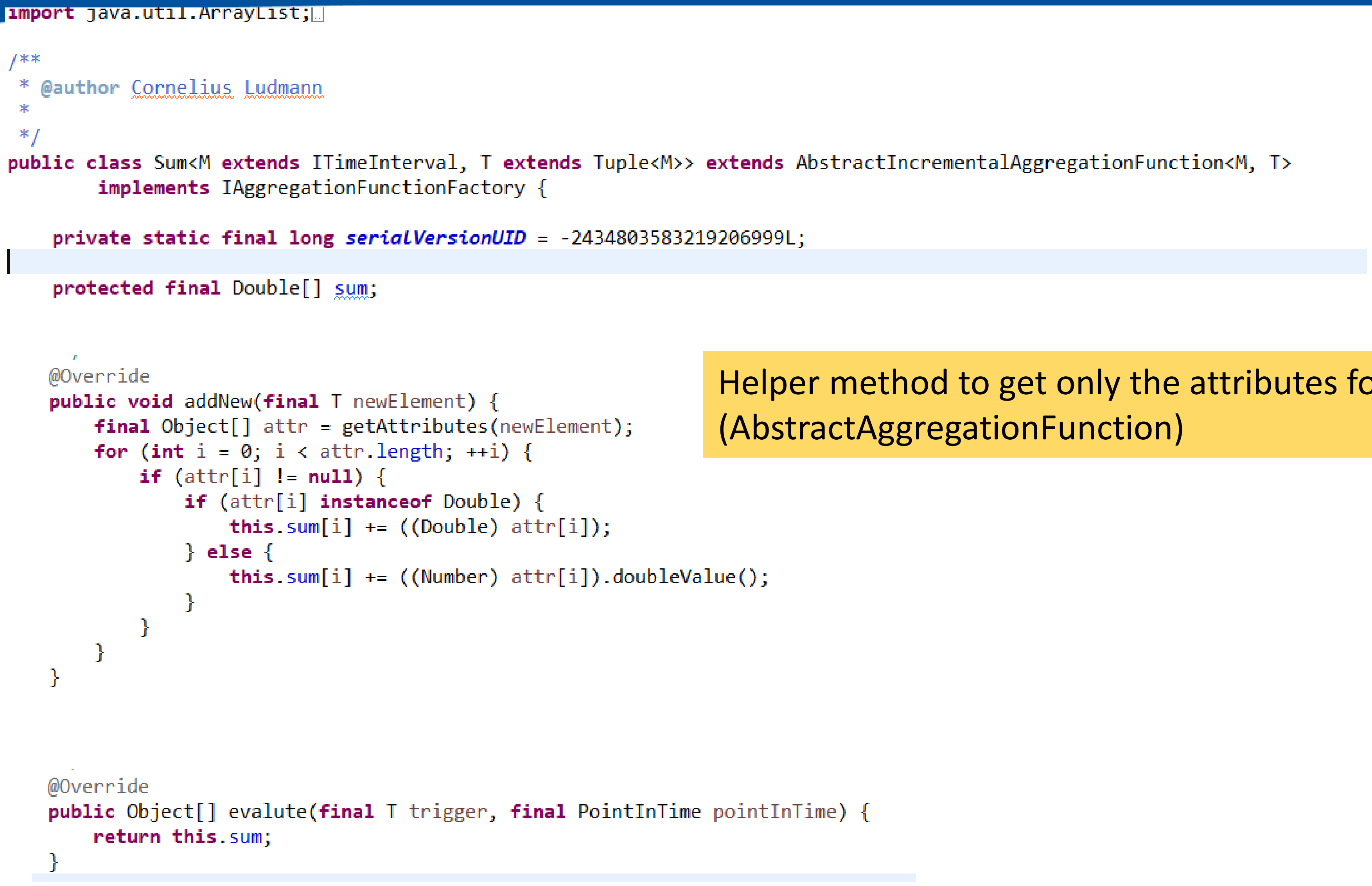
```java
package de.uniol.inf.is.odysseus.aggregation.functions;

import java.util.Collection;

/**
 * @author Cornelius Ludmann
 *
 */
public interface IIncrementalAggregationFunction<M extends ITimeInterval, T extends Tuple<M>>
        extends IAggregationFunction, Cloneable {

    Object[] addNewAndEvaluate(T newElement);

    void addNew(T newElement);

    Object[] removeOutdatedAndEvaluate(Collection<T> outdatedElements, T trigger, PointInTime pointInTime);

    void removeOutdated(Collection<T> outdatedElements, T trigger, PointInTime pointInTime);

    Object[] evalute(T trigger, PointInTime pointInTime);

    IIncrementalAggregationFunction<M, T> clone();
}
```

Example

```
counted = AGGREGATION({AGGREGATIONS = [
        ['FUNCTION' = 'Count‘],
        ['FUNCTION' = 'Sum', 'INPUT_ATTRIBUTES' = 'value1, value2']],
        GROUP_BY = ['publisher', 'item']}, windowed)
```

- The input of each aggregation function is a tuple
- Some functions can calculate more than on input (see SUM above), same can be done with multiple SUM aggregations
- Group by is handled outside → aggregation functions need to know nothing about groups
- Aggregation functions can return an Object-array of values, each value is copied to the position in the output tuple
  - above would be something like: count, sum_value1, sum_value2, publisher, item

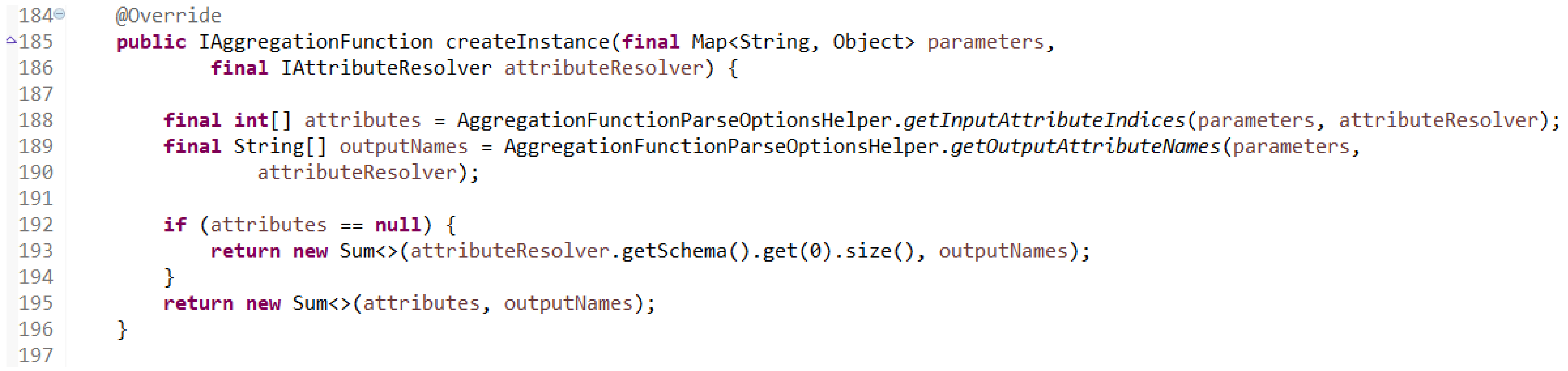
Example Sum Aggregation function



```
import java.util.ArrayList;

/**
 * @author Cornelius Ludmann
 *
 */
public class Sum<M extends ITimeInterval, T extends Tuple<M>> extends AbstractIncrementalAggregationFunction<M, T>
        implements IAggregationFunctionFactory {

    private static final long serialVersionUID = -2434803583219206999L;

    protected final Double[] sum;


    @Override
    public void addNew(final T newElement) {
        final Object[] attr = getAttributes(newElement);
        for (int i = 0; i < attr.length; ++i) {
            if (attr[i] != null) {
                if (attr[i] instanceof Double) {
                    this.sum[i] += ((Double) attr[i]);
                } else {
                    this.sum[i] += ((Number) attr[i]).doubleValue();
                }
            }
        }
    }


    @Override
    public Object[] evalute(final T trigger, final PointInTime pointInTime) {
        return this.sum;
    }
```

Helper method to get only the attributes for SUM (AbstractAggregationFunction)

Factory to create new Sum aggregation function

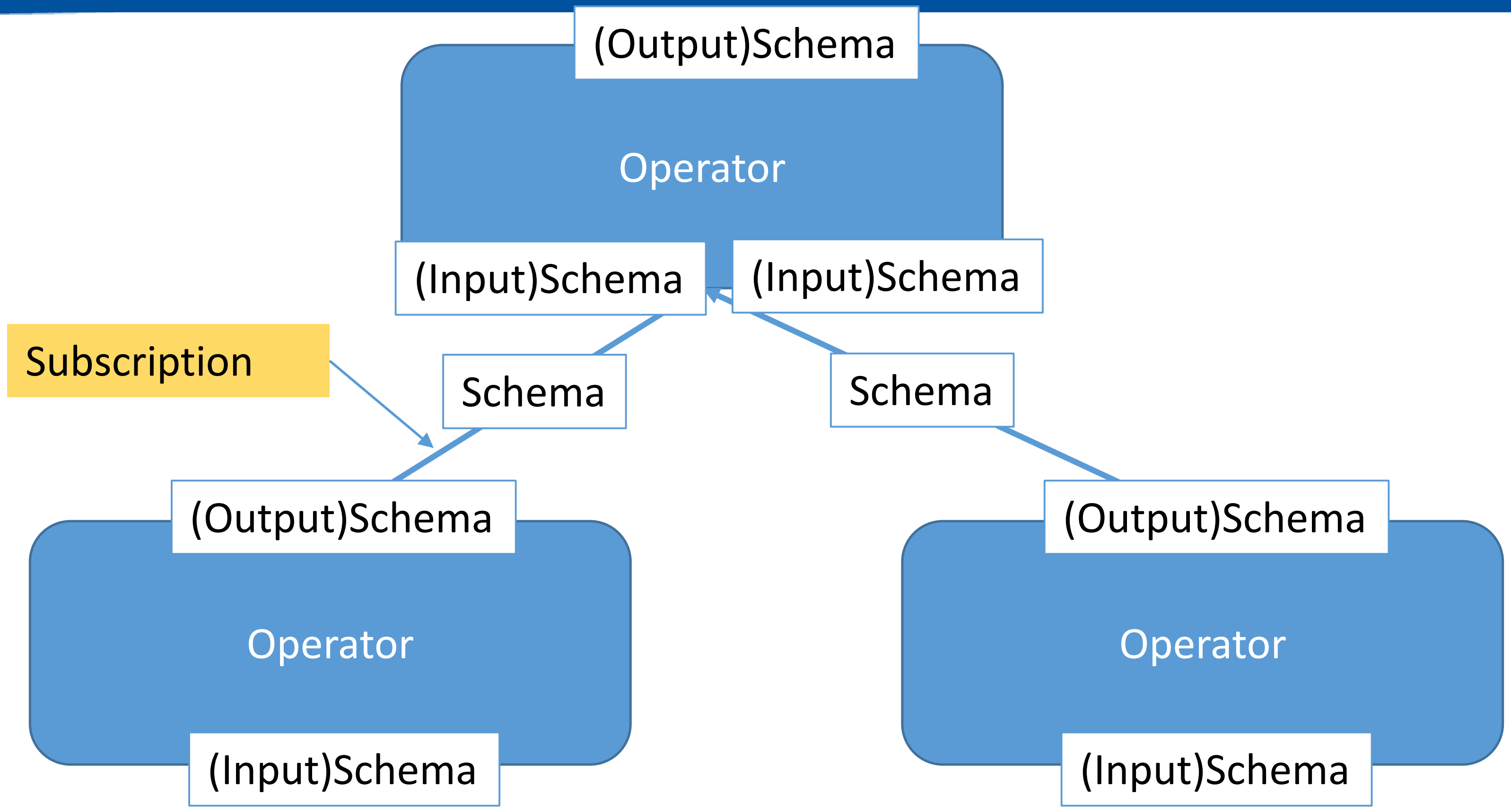


```java
@Override
public IAggregationFunction createInstance(final Map<String, Object> parameters,
        final IAttributeResolver attributeResolver) {

    final int[] attributes = AggregationFunctionParseOptionsHelper.getInputAttributeIndices(parameters, attributeResolver);
    final String[] outputNames = AggregationFunctionParseOptionsHelper.getOutputAttributeNames(parameters,
            attributeResolver);

    if (attributes == null) {
        return new Sum<>(attributeResolver.getSchema().get(0).size(), outputNames);
    }
    return new Sum<>(attributes, outputNames);
}
```

Extending PQL

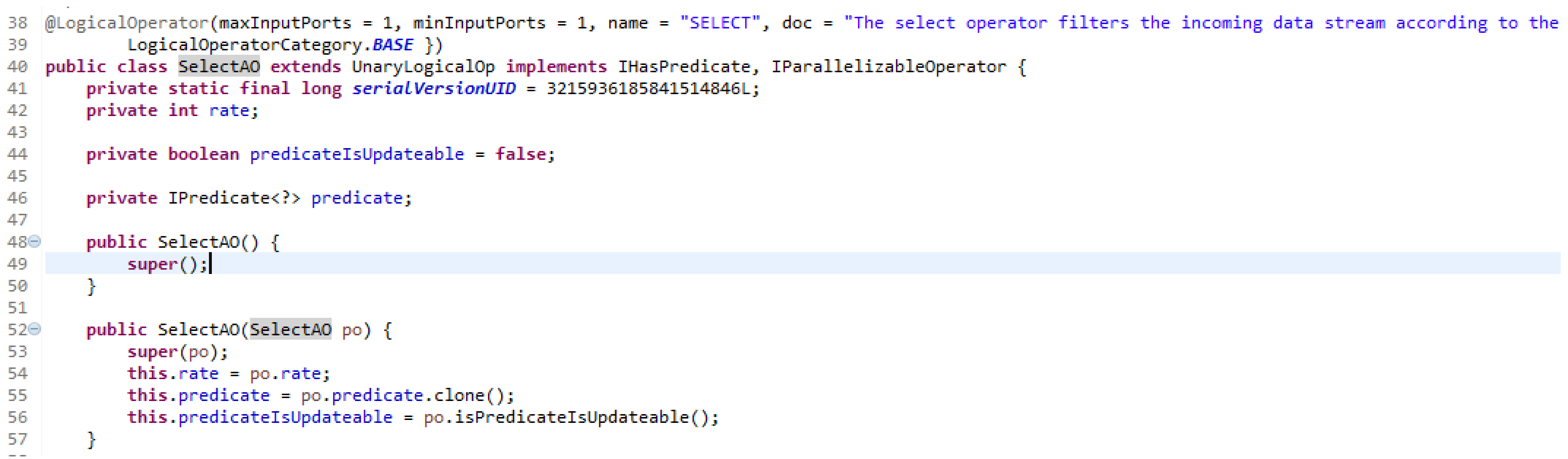
- PQL a our language to create algebra expressions → Create a logical query model (quite similar to the relational algebra)
- To allow an easy extension, the PQL parser knows nothing about concrete operators
- The parser only knows about (abstract) logical operators and configurations
- Concrete operators are pluged into the parser
- For this:
  - An annotation model
  - An automatic loading mechanismen (no need to write services for operators)

Logical Operator Model



- Schema provides (meta-)information about the stream:
  - What stream type (tuple, key value, xml …)
  - Out of order
  - For tuple: set of attributes
  - Constraints
  - …

Annotation model

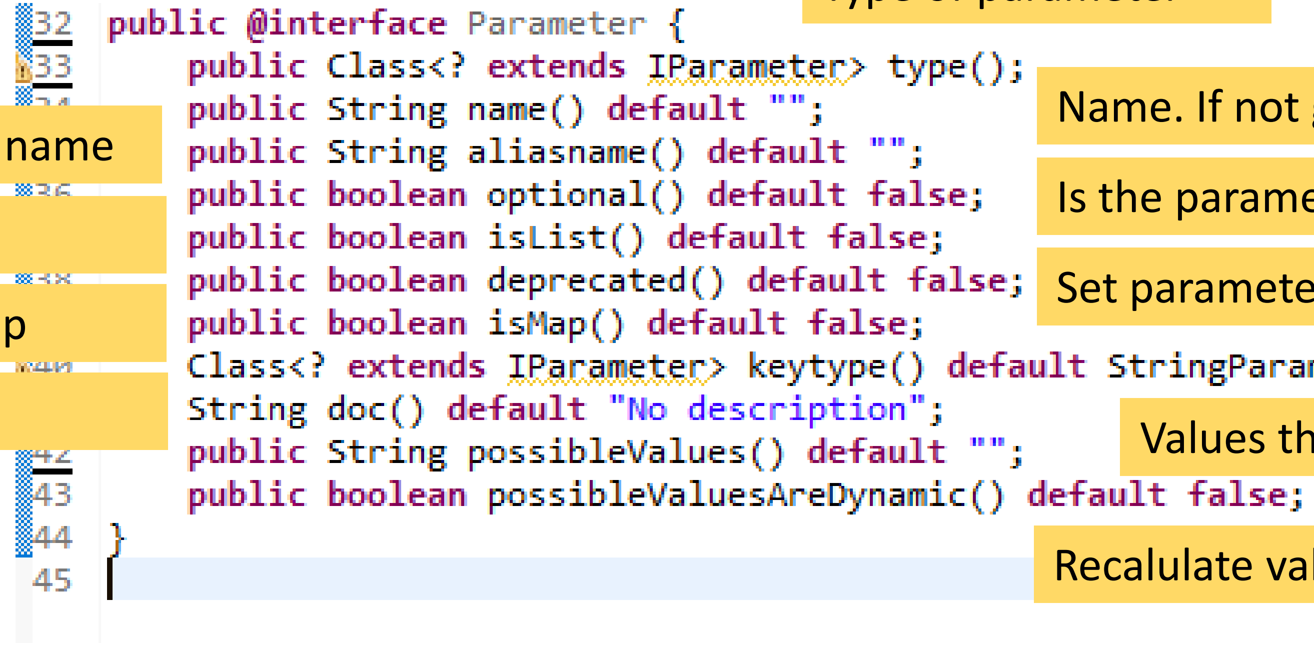


```
@LogicalOperator(maxInputPorts = 1, minInputPorts = 1, name = "SELECT", doc = "The select operator filters the incoming data stream according to the
        LogicalOperatorCategory.BASE })
public class SelectAO extends UnaryLogicalOp implements IHasPredicate, IParallelizableOperator {
    private static final long serialVersionUID = 3215936185841514846L;
    private int rate;

    private boolean predicateIsUpdateable = false;

    private IPredicate<?> predicate;

    public SelectAO() {
        super();
    }

    public SelectAO(SelectAO po) {
        super(po);
        this.rate = po.rate;
        this.predicate = po.predicate.clone();
        this.predicateIsUpdateable = po.isPredicateIsUpdateable();
    }
```

- @LogicalOperator: State ILogicalOperator implementation as PQL operator
- maxInputPorts, minInputPorts: How many input must be bound, can be bound
- Name: Name of the operator in PQL
- Helper for user interface
  - Doc: A text describing the operator function
  - Url: Link to documentation
  - Category: Kind of operator

Annotation model

- Typically, each logical operator has a set of properties (e.g. in SELECT the predicate)
- A large set of specialized parameter handler (e.g.PredicateParameter)
- Can (of course) be extended ;-)
- The PQL parser does some preprocessing and delegates parameter handling to the parameter classes



```
public @interface Parameter {
    public Class<? extends IParameter> type();
    public String name() default "";
    public String aliasname() default "";
    public boolean optional() default false;
    public boolean isList() default false;
    public boolean deprecated() default false;
    public boolean isMap() default false;
    Class<? extends IParameter> keytype() default StringParameter.class;
    String doc() default "No description";
    public String possibleValues() default "";
    public boolean possibleValuesAreDynamic() default false;
}
```

Type of parameter

Name. If not given, read from method name

Alternative name

Is the parameter required for the operator

Is this a list

Set parameter as deprecated

Is this a map

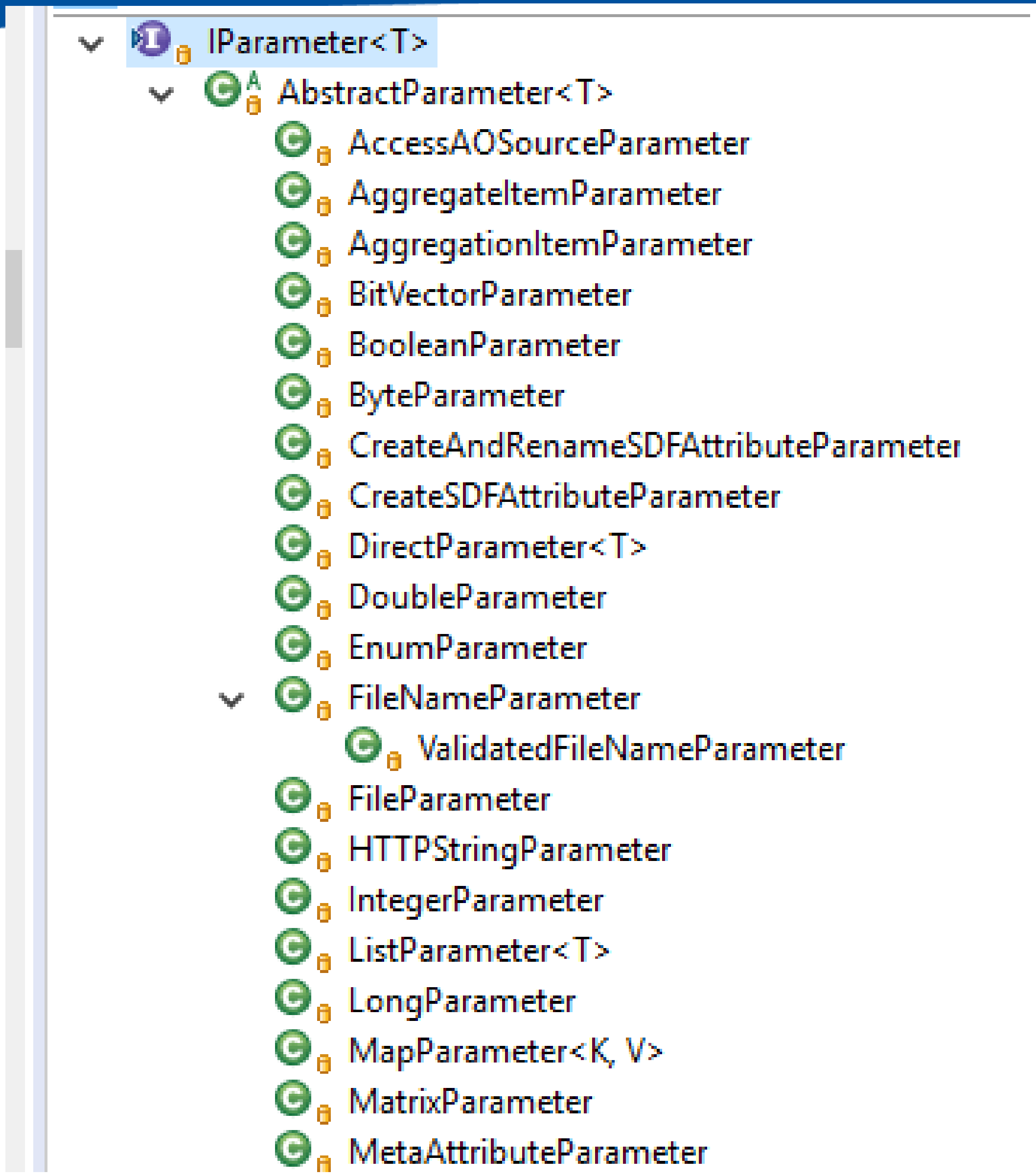
If map, which keytype

Doc for gui

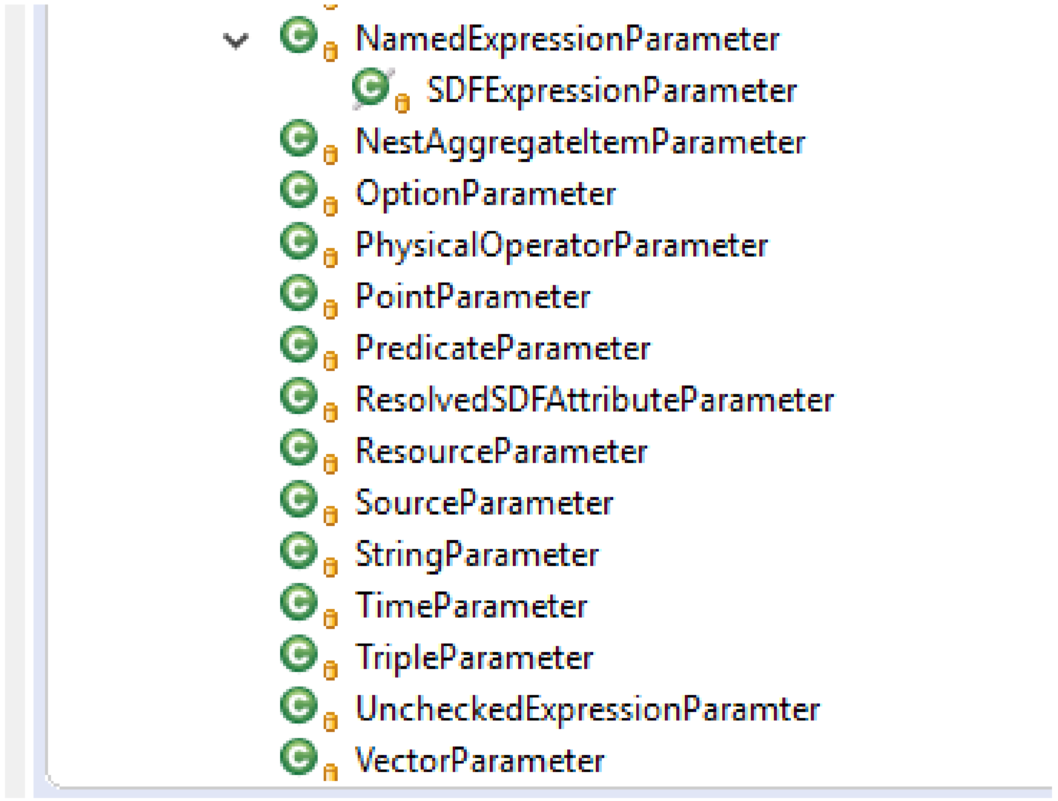
Values that can be used

Recalulate values at each access?

Different predefined parameter classes



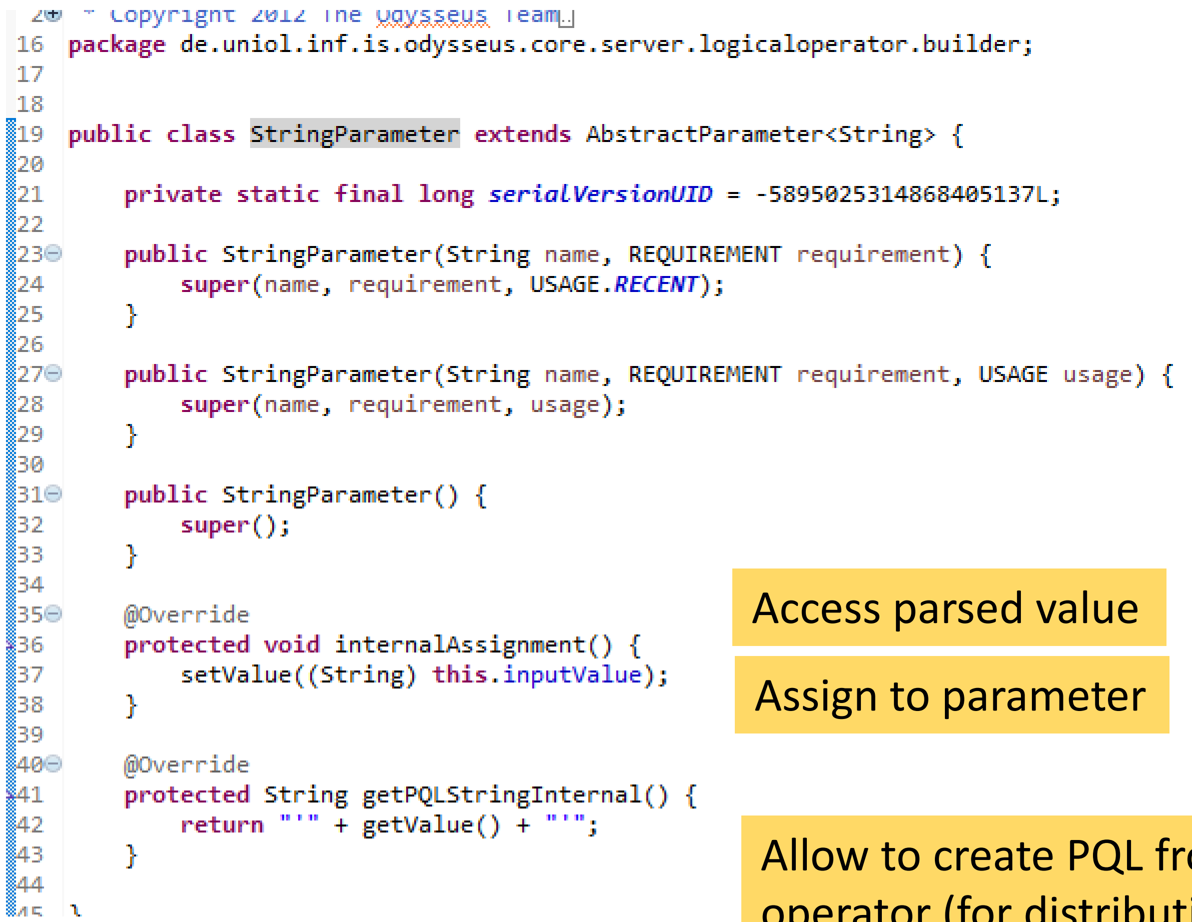
```
IParameter<T>
  AbstractParameter<T>
    AccessAOSourceParameter
    AggregateItemParameter
    AggregationItemParameter
    BitVectorParameter
    BooleanParameter
    ByteParameter
    CreateAndRenameSDFAttributeParameter
    CreateSDFAttributeParameter
    DirectParameter<T>
    DoubleParameter
    EnumParameter
    FileNameParameter
      ValidatedFileNameParameter
    FileParameter
    HTTPStringParameter
    IntegerParameter
    ListParameter<T>
    LongParameter
    MapParameter<K, V>
    MatrixParameter
    MetaAttributeParameter
```



```
    NamedExpressionParameter
      SDFExpressionParameter
    NestAggregateItemParameter
    OptionParameter
    PhysicalOperatorParameter
    PointParameter
    PredicateParameter
    ResolvedSDFAttributeParameter
    ResourceParameter
    SourceParameter
    StringParameter
    TimeParameter
    TripleParameter
    UncheckedExpressionParamter
    VectorParameter
```

PQL and Parameter

- Text inside of „“ or ‚‘ are directly parsed, e.g. StringParameter



```
 * Copyright 2012 The Odysseus Team
package de.uniol.inf.is.odysseus.core.server.logicaloperator.builder;


public class StringParameter extends AbstractParameter<String> {

    private static final long serialVersionUID = -5895025314868405137L;

    public StringParameter(String name, REQUIREMENT requirement) {
        super(name, requirement, USAGE.RECENT);
    }

    public StringParameter(String name, REQUIREMENT requirement, USAGE usage) {
        super(name, requirement, usage);
    }

    public StringParameter() {
        super();
    }

    @Override
    protected void internalAssignment() {
        setValue((String) this.inputValue);
    }

    @Override
    protected String getPQLStringInternal() {
        return "'" + getValue() + "'";
    }

}
```

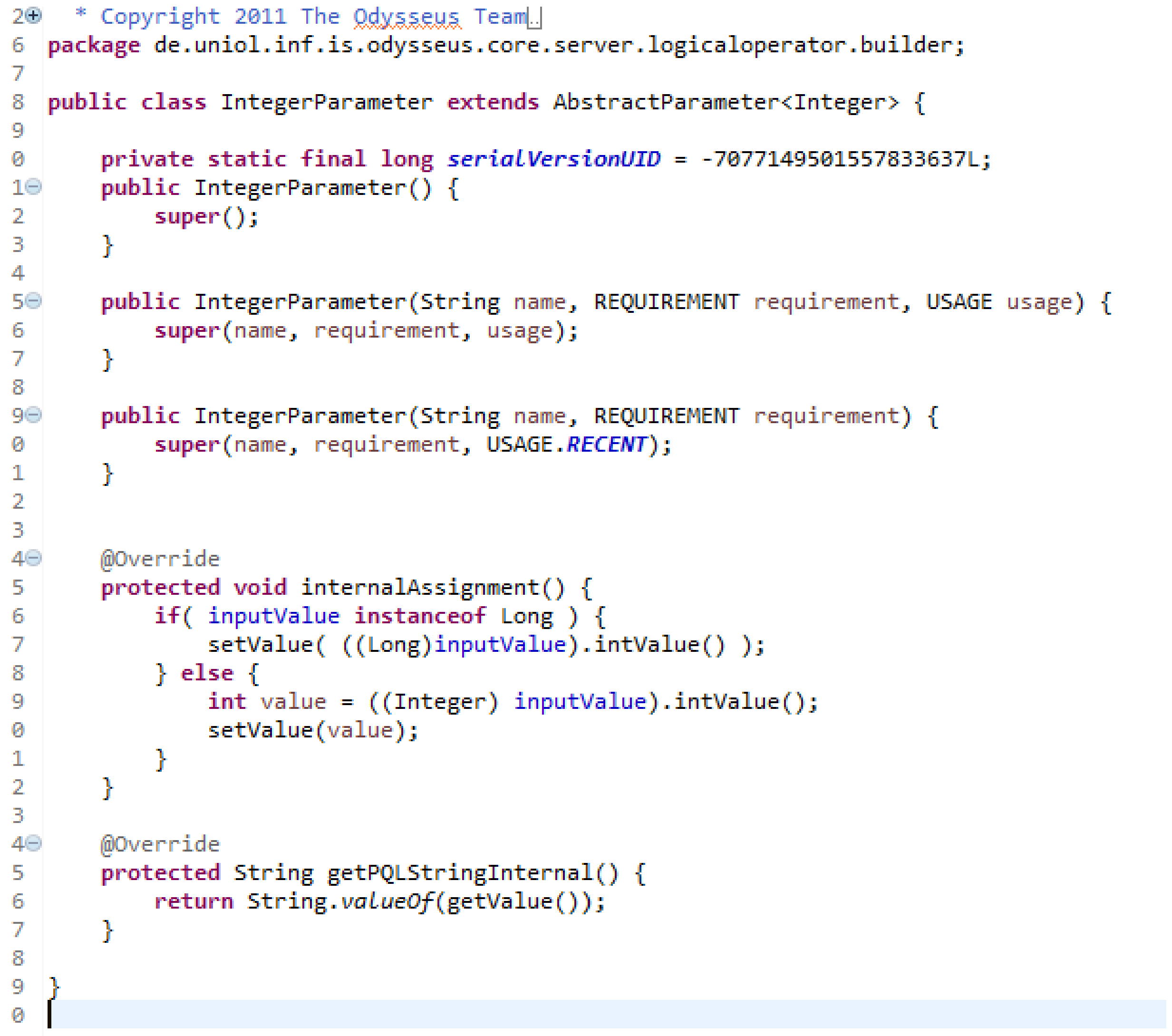
Access parsed value

Assign to parameter

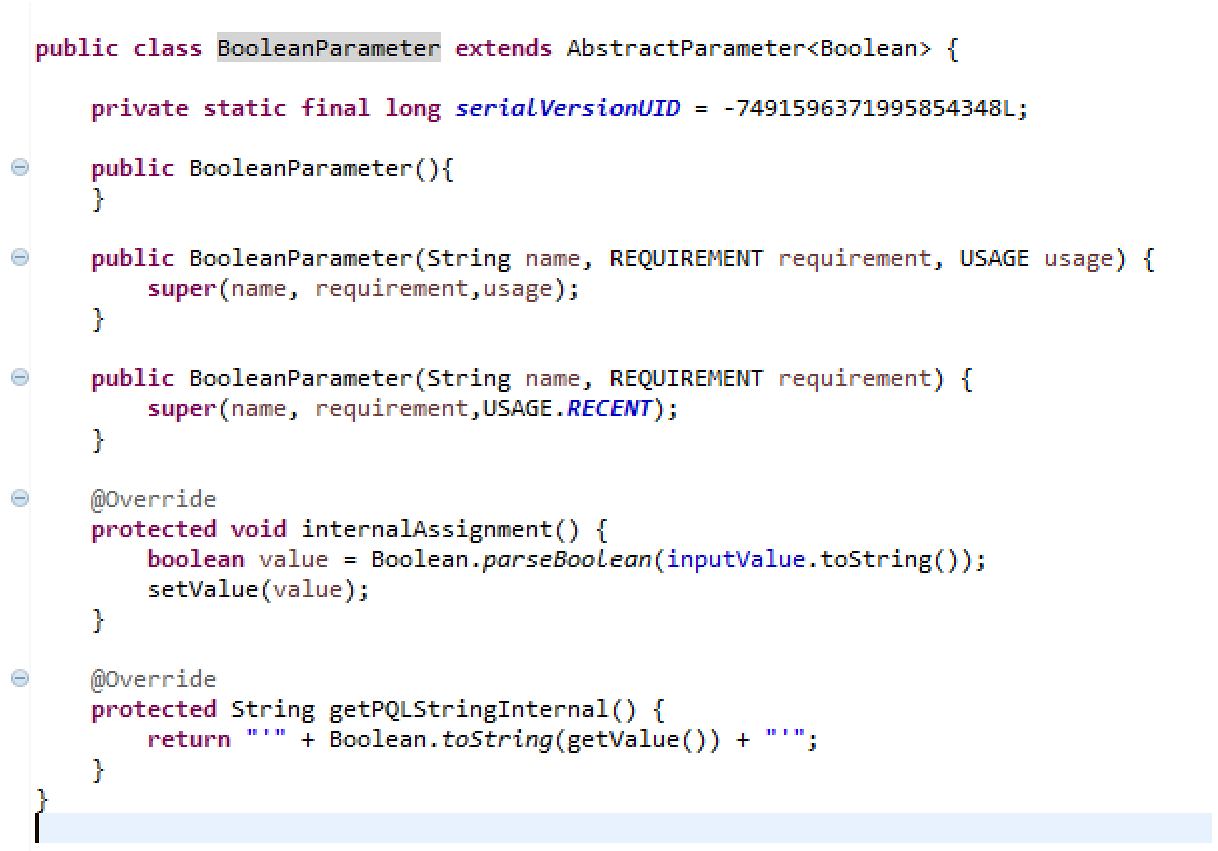
Allow to create PQL from an logical operator (for distribution)

PQL and Parameter

- Numbers/Boolean are parsed as types, e.g. IntegerParameter



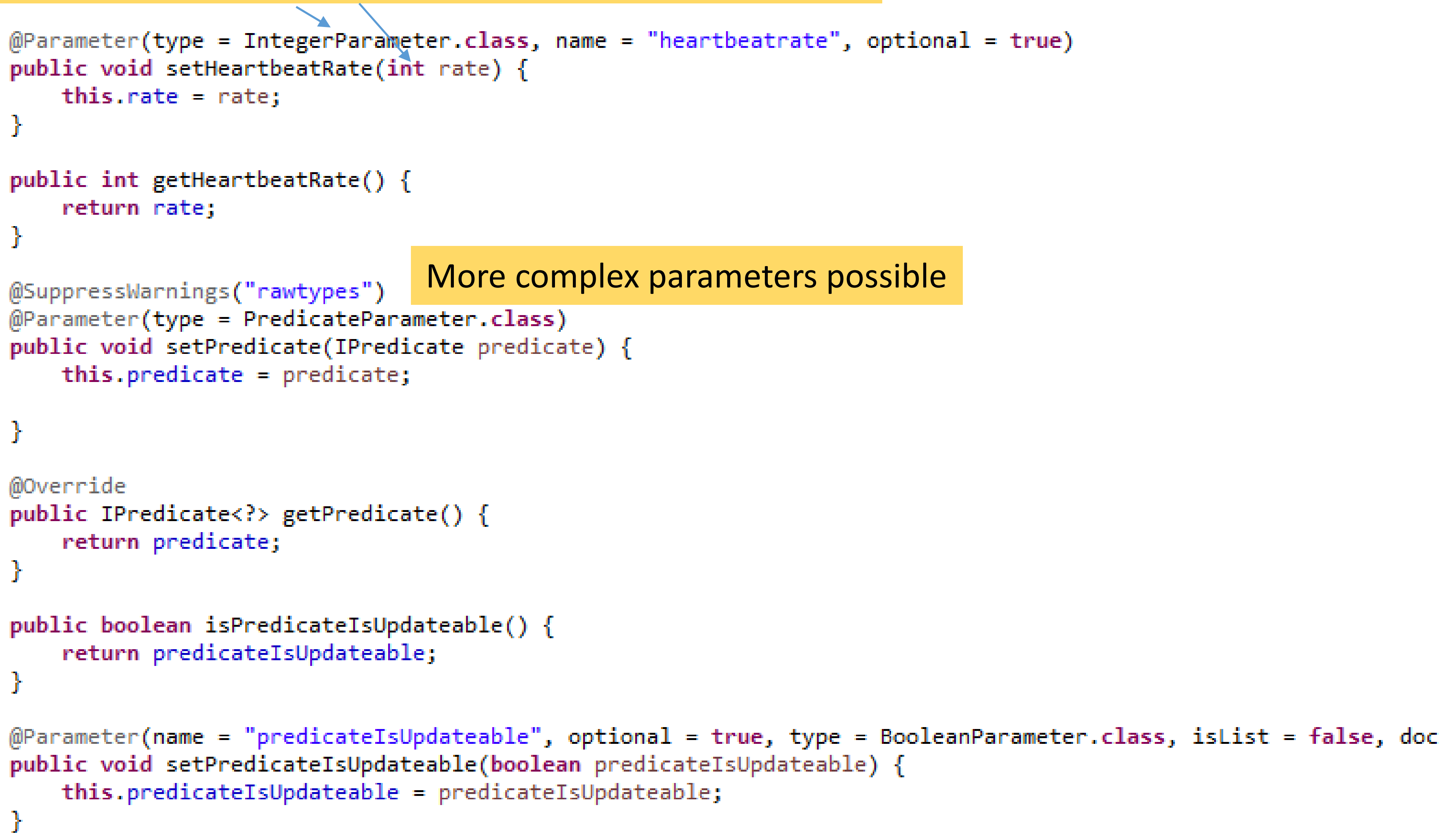
```java
 * Copyright 2011 The Odysseus Team
package de.uniol.inf.is.odysseus.core.server.logicaloperator.builder;

public class IntegerParameter extends AbstractParameter<Integer> {

    private static final long serialVersionUID = -7077149501557833637L;
    public IntegerParameter() {
        super();
    }

    public IntegerParameter(String name, REQUIREMENT requirement, USAGE usage) {
        super(name, requirement, usage);
    }

    public IntegerParameter(String name, REQUIREMENT requirement) {
        super(name, requirement, USAGE.RECENT);
    }


    @Override
    protected void internalAssignment() {
        if( inputValue instanceof Long ) {
            setValue( ((Long)inputValue).intValue() );
        } else {
            int value = ((Integer) inputValue).intValue();
            setValue(value);
        }
    }

    @Override
    protected String getPQLStringInternal() {
        return String.valueOf(getValue());
    }

}
```



```java
public class BooleanParameter extends AbstractParameter<Boolean> {

    private static final long serialVersionUID = -7491596371995854348L;

    public BooleanParameter(){
    }

    public BooleanParameter(String name, REQUIREMENT requirement, USAGE usage) {
        super(name, requirement,usage);
    }

    public BooleanParameter(String name, REQUIREMENT requirement) {
        super(name, requirement,USAGE.RECENT);
    }

    @Override
    protected void internalAssignment() {
        boolean value = Boolean.parseBoolean(inputValue.toString());
        setValue(value);
    }

    @Override
    protected String getPQLStringInternal() {
        return "'" + Boolean.toString(getValue()) + "'";
    }
}
```

Usage in logical operator

Types must match, is not checked at compile time → runtime error



```
@Parameter(type = IntegerParameter.class, name = "heartbeatrate", optional = true)
public void setHeartbeatRate(int rate) {
    this.rate = rate;
}

public int getHeartbeatRate() {
    return rate;
}

@SuppressWarnings("rawtypes")
@Parameter(type = PredicateParameter.class)
public void setPredicate(IPredicate predicate) {
    this.predicate = predicate;

}

@Override
public IPredicate<?> getPredicate() {
    return predicate;
}

public boolean isPredicateIsUpdateable() {
    return predicateIsUpdateable;
}

@Parameter(name = "predicateIsUpdateable", optional = true, type = BooleanParameter.class, isList = false, doc
public void setPredicateIsUpdateable(boolean predicateIsUpdateable) {
    this.predicateIsUpdateable = predicateIsUpdateable;
}
```

More complex parameters possible

Predicate parameter

Only one handler for all predicate types!

```
@Override
protected void internalAssignment() {
    String predicateType = "";
    String predicate = "";
    if (inputValue instanceof PredicateItem) {
        PredicateItem pItem = (PredicateItem) inputValue;
        predicateType = pItem.getPredicateType();
        predicate = pItem.getPredicate();
    } else if (inputValue instanceof String) {
        if (getAttributeResolver().getSchema().size() > 0) {
            predicateType = getAttributeResolver().getSchema().get(0)
                    .getType().getName();
        }
        predicate = (String) inputValue;
    } else if( inputValue instanceof IPredicate ) {
        setValue((IPredicate<?>)inputValue);
        return;
    }
    @SuppressWarnings("rawtypes")
    IExpressionBuilder pBuilder = OperatorBuilderFactory
            .getExpressionBuilder(predicateType);

    if (pBuilder == null) {
        throw new IllegalArgumentException("unkown type of predicate: "
                + predicateType);
    }
    setValue(pBuilder.createPredicate(getAttributeResolver(), predicate));
}
```

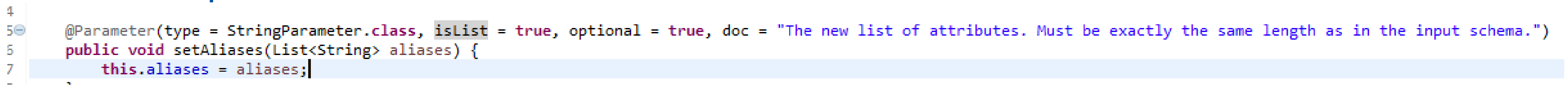
The current type (e.g. tuple, key value) can be read from the input schema of the operator

Factory for different predicate types

Create and assign predicate

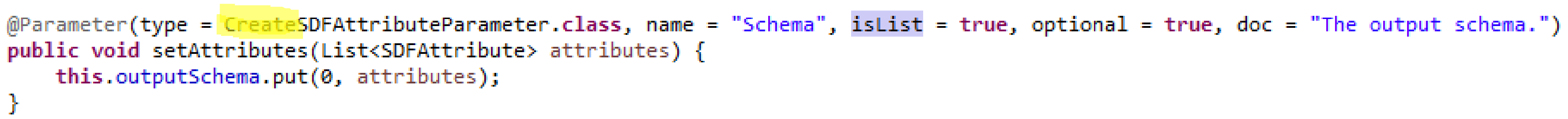
Parameter: isList = true

- If isList = true, then input will be interpreted as List, must be enclosed by „[" and „]“
- Create List of elements
- No new parameter handler is needed!
- Example Rename:



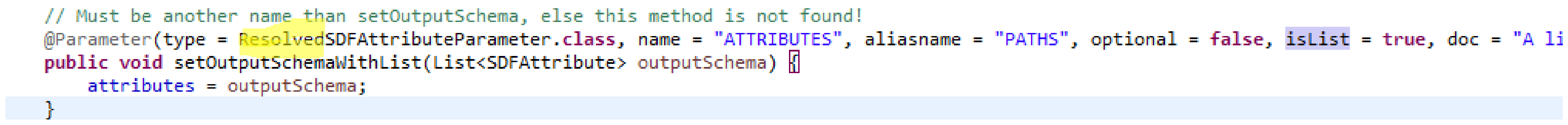
```java
    @Parameter(type = StringParameter.class, isList = true, optional = true, doc = "The new list of attributes. Must be exactly the same length as in the input schema.")
    public void setAliases(List<String> aliases) {
        this.aliases = aliases;
```

- Example Schemacreation (here from AbstractAccessAO)



```java
@Parameter(type = CreateSDFAttributeParameter.class, name = "Schema", isList = true, optional = true, doc = "The output schema.")
public void setAttributes(List<SDFAttribute> attributes) {
    this.outputSchema.put(0, attributes);
}
```

- Example Attributereference (here from ProjectAO)



```java
// Must be another name than setOutputSchema, else this method is not found!
@Parameter(type = ResolvedSDFAttributeParameter.class, name = "ATTRIBUTES", aliasname = "PATHS", optional = false, isList = true, doc = "A li
public void setOutputSchemaWithList(List<SDFAttribute> outputSchema) {
    attributes = outputSchema;
}
```

Extensions

- Language extensions
  - Create a new language ✓
  - Create a new Odysseus Script Command (#....) ✓
  - Create a new PQL Operator ✓
- Processing function extensions
  - Create new datatypes ✓
  - Create new stream object types
    - Data Handler (✓)
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates ✓
  - Create new aggregationsfunctions
  - Create new operators
- Create new schedulers and scheduling strategies
- ...

- The logical operator is only one part
- There must be a corresponding physical operator, the operator that does the „work“
- In Odysseus typical: One logical and one physical operator
- Transformation from logical to physical with a transformation rule

- Rewrite rules to optimize plan and to handle speficic situations, e.g. out of order processing

Physical operators

- We‘ve created too many operators ;-)
  - So, before creating a new operator, think about:
    - Could this be a MAP operation → write MEP function
    - Could this be an aggregation → write Aggregation function
- If you finally decide to write an operator: It is quite easy to do:
  1. Create a logical operator
  2. Create a physical operator
  3. Create a transformation rule

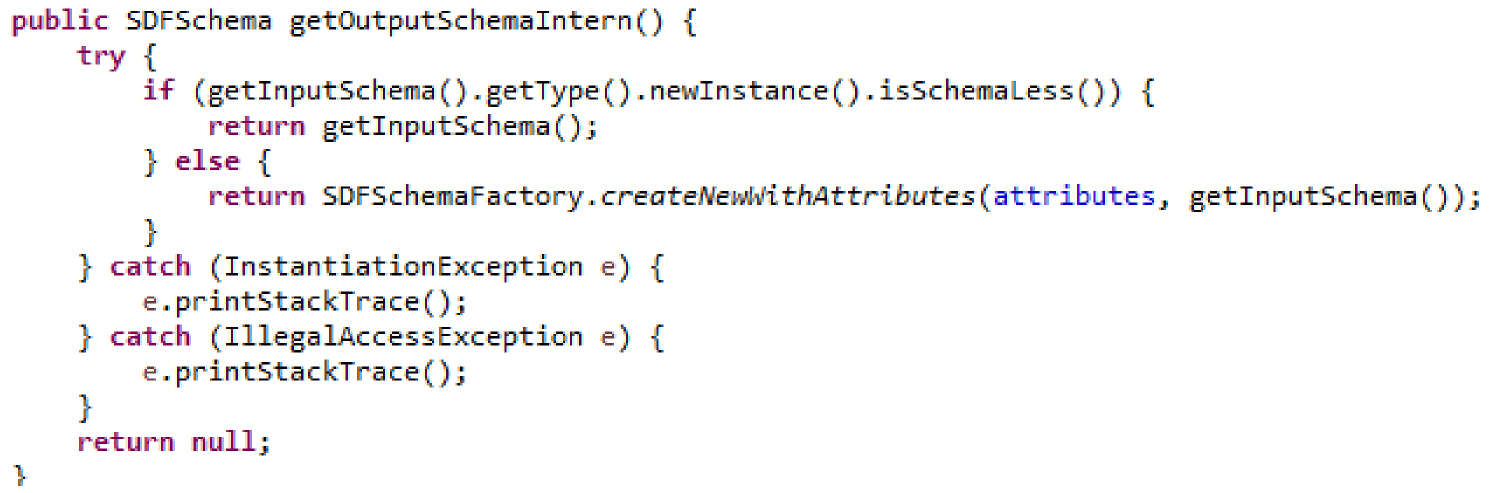
Create a logical operator

- Odysseus has an automated operator location function
- For this: The operator needs to be placed in a package ending with logicaloperator
- Create your new operator by extending AbstractLogicalOperator (or Binary/UnaryLogicalOperator)
- As naming convention: The operator should end with AO (for algebra operator)
- Very important:
  - The new class must provide a least two constructor
    - The default constructor
    - A copy constructor (i.e. inialization from itself). The copy constructor must call its super copy constructor!
  - There must be a clone method that calls the copy constructor
- Use annotations to define name and setters

```java
private List<SDFAttribute> attributes = new ArrayList<>();

public ProjectAO() {
    super();
}

public ProjectAO(ProjectAO ao) {
    super(ao);
    this.attributes = new ArrayList<>(ao.attributes);
}

public @Override ProjectAO clone() {
    return new ProjectAO(this);
}
```

Output schema

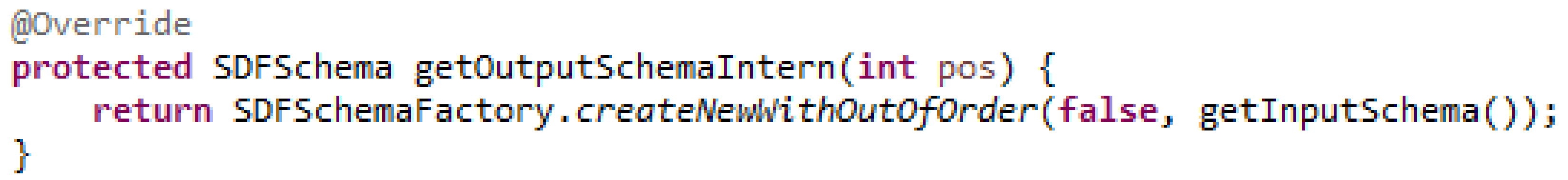
- Each operator must provide schema information, i.e. what kind of data is created
- Schema has information about:
  - The processing type (IStreamObject): e.g. tuple, keyvalueobject, xml
  - The schema of the type (typically only for tuples)
  - The schema of the meta types (SDFMetaSchema): e.g. TimeInterval, Latency, …
  - Some contraints on the schema
  - A flag indicating, if the stream is potentially out of order
- Use SDFSchemaFactory to create output schema from input schema (and preserve information that is not changed by the operator)



```java
public SDFSchema getOutputSchemaIntern() {
    try {
        if (getInputSchema().getType().newInstance().isSchemaLess()) {
            return getInputSchema();
        } else {
            return SDFSchemaFactory.createNewWithAttributes(attributes, getInputSchema());
        }
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
    return null;
}
```

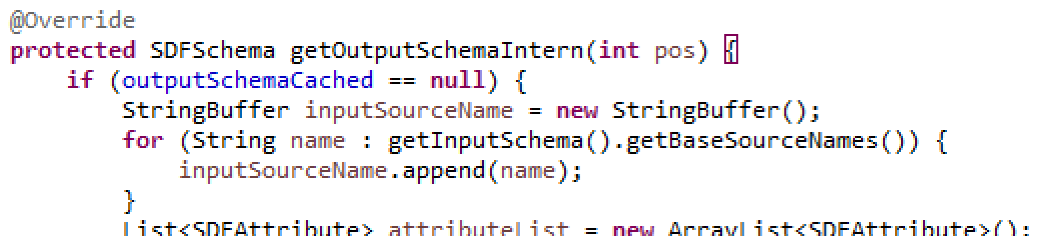
Output schema examples

- ReOrderAO (Schema same as input, but ordered)



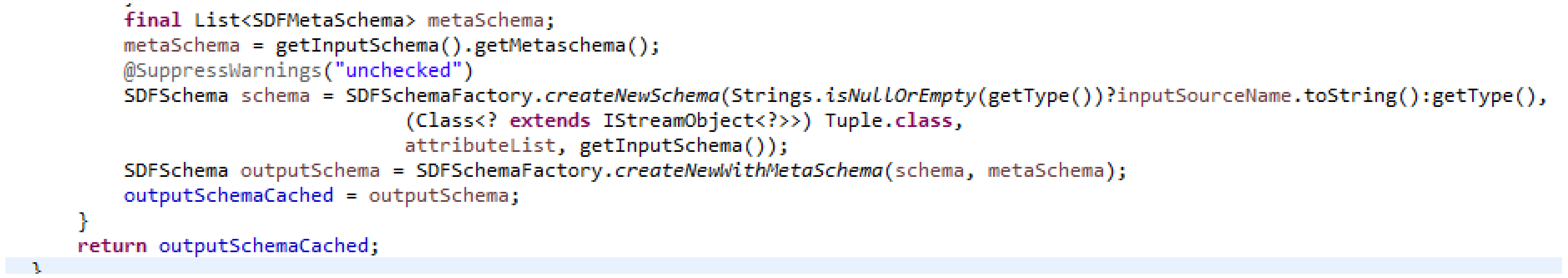
```java
@Override
protected SDFSchema getOutputSchemaIntern(int pos) {
    return SDFSchemaFactory.createNewWithOutOfOrder(false, getInputSchema());
}
```

- ToTupleAO (Input schema is another type, output is tuple, keep meta schema)



```java
@Override
protected SDFSchema getOutputSchemaIntern(int pos) {
    if (outputSchemaCached == null) {
        StringBuffer inputSourceName = new StringBuffer();
        for (String name : getInputSchema().getBaseSourceNames()) {
            inputSourceName.append(name);
        }
        List<SDFAttribute> attributeList = new ArrayList<SDFAttribute>();
```

…



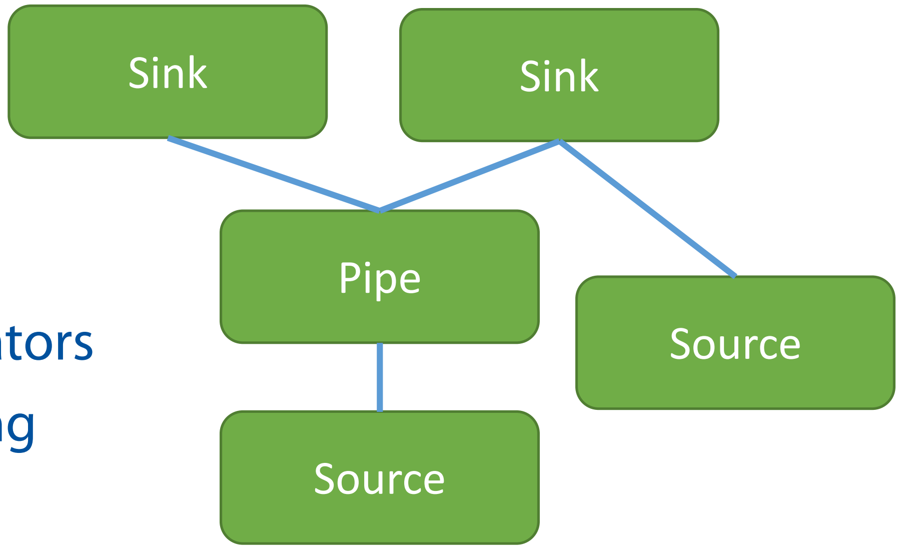
```java
        final List<SDFMetaSchema> metaSchema;
        metaSchema = getInputSchema().getMetaschema();
        @SuppressWarnings("unchecked")
        SDFSchema schema = SDFSchemaFactory.createNewSchema(Strings.isNullOrEmpty(getType())?inputSourceName.toString():getType(),
                                (Class<? extends IStreamObject<?>>) Tuple.class,
                                attributeList, getInputSchema());
        SDFSchema outputSchema = SDFSchemaFactory.createNewWithMetaSchema(schema, metaSchema);
        outputSchemaCached = outputSchema;
    }
    return outputSchemaCached;
}
```

Physical subscriptions

- Similar to the logical subscription model, there is a physical pendant
- Operators are connected to each other by physical subscriptions (in both directions)
- Subscriptions can be opened or closed
- Only on open subscriptions data is processed
- A subscription has a source (data sender) and a sink (data receiver), a source port number and a sink port number
- A subscription has a schema
- Subscriptions handle object cloning
- There are two types of physical subscriptions:
  - UncontrollablePhysicalSubscription
  - ControllablePhysicalSubscription
- Controllable subscriptions provide a buffer and can e.g. be paused and resumed, or can do load shedding

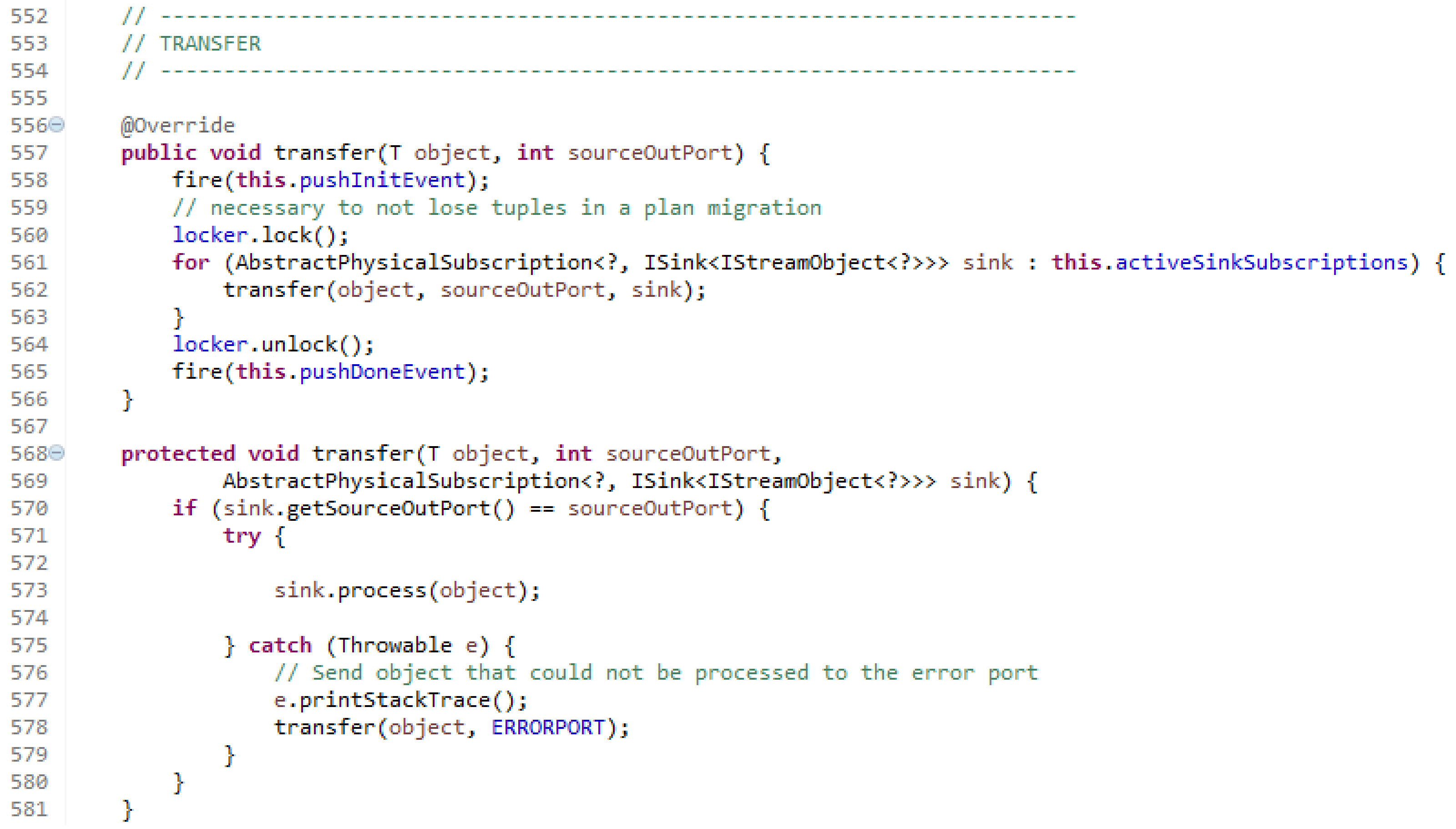
Physical operators

- Each physical operator is
  - A sink: can receive data (AbstractSink)
  - A source: can send data (AbstractSource)
  - A pipe: can send and receive data (AbstractPipe)
- Odysseus has a special processing model (protocol) for operators
- Open and close: Initialize processing and terminate processing
  - are called from top (phyiscal query)
  - Will be recursevely called on children of each operator
  - If operator needs inialization override: process_open()
  - If operator needs termination override: process_close()
  - Multiple sinks can be connected to same source, source can deliver data to multipe sinks
  - Only the first call from any sink, leads to process_*, in other cases only the subscription is activated
- Operator are shared between queries, i.e. an operator can be part of many queries



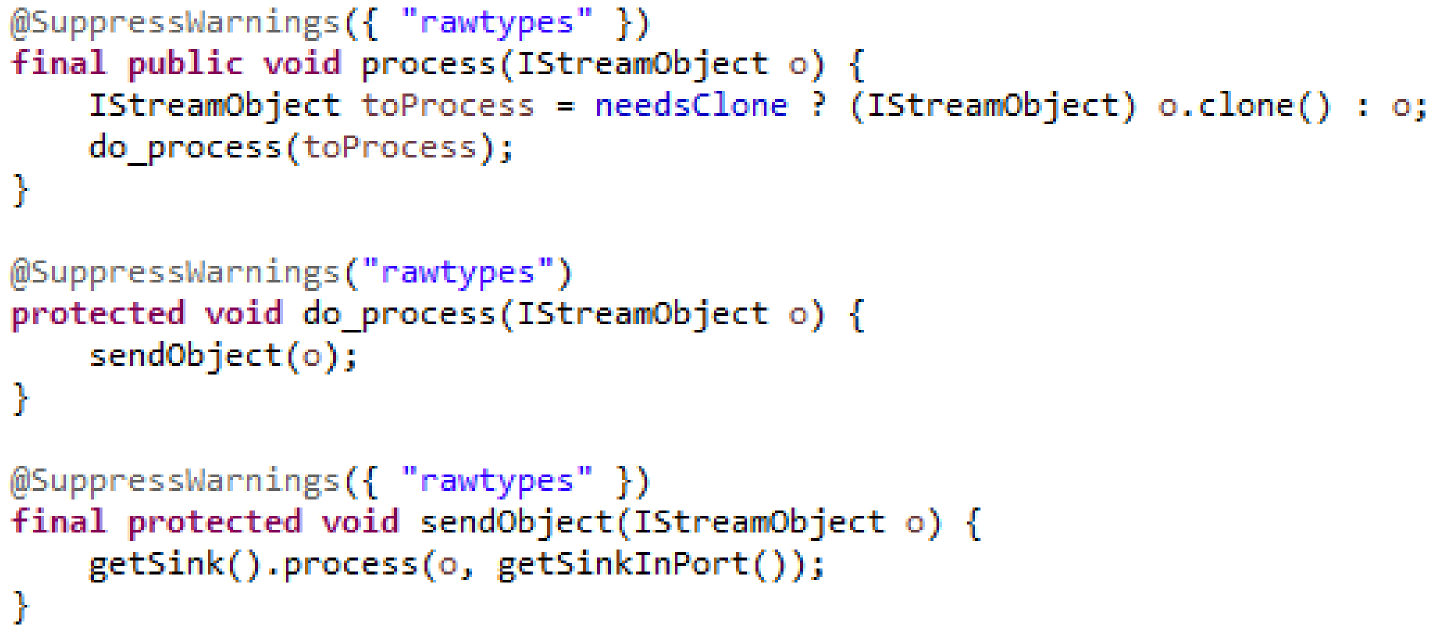
Physical operators

- Processing is done from sources to sinks
- For each new element: transfer() is called from sources
- Transfer leads to call of process () in any active sink subscription



```java
// ------------------------------------------------------------------------
// TRANSFER
// ------------------------------------------------------------------------

@Override
public void transfer(T object, int sourceOutPort) {
    fire(this.pushInitEvent);
    // necessary to not lose tuples in a plan migration
    locker.lock();
    for (AbstractPhysicalSubscription<?, ISink<IStreamObject<?>>> sink : this.activeSinkSubscriptions) {
        transfer(object, sourceOutPort, sink);
    }
    locker.unlock();
    fire(this.pushDoneEvent);
}

protected void transfer(T object, int sourceOutPort,
        AbstractPhysicalSubscription<?, ISink<IStreamObject<?>>> sink) {
    if (sink.getSourceOutPort() == sourceOutPort) {
        try {

            sink.process(object);

        } catch (Throwable e) {
            // Send object that could not be processed to the error port
            e.printStackTrace();
            transfer(object, ERRORPORT);
        }
    }
}
```

Physical operators: PhysicalSubscription

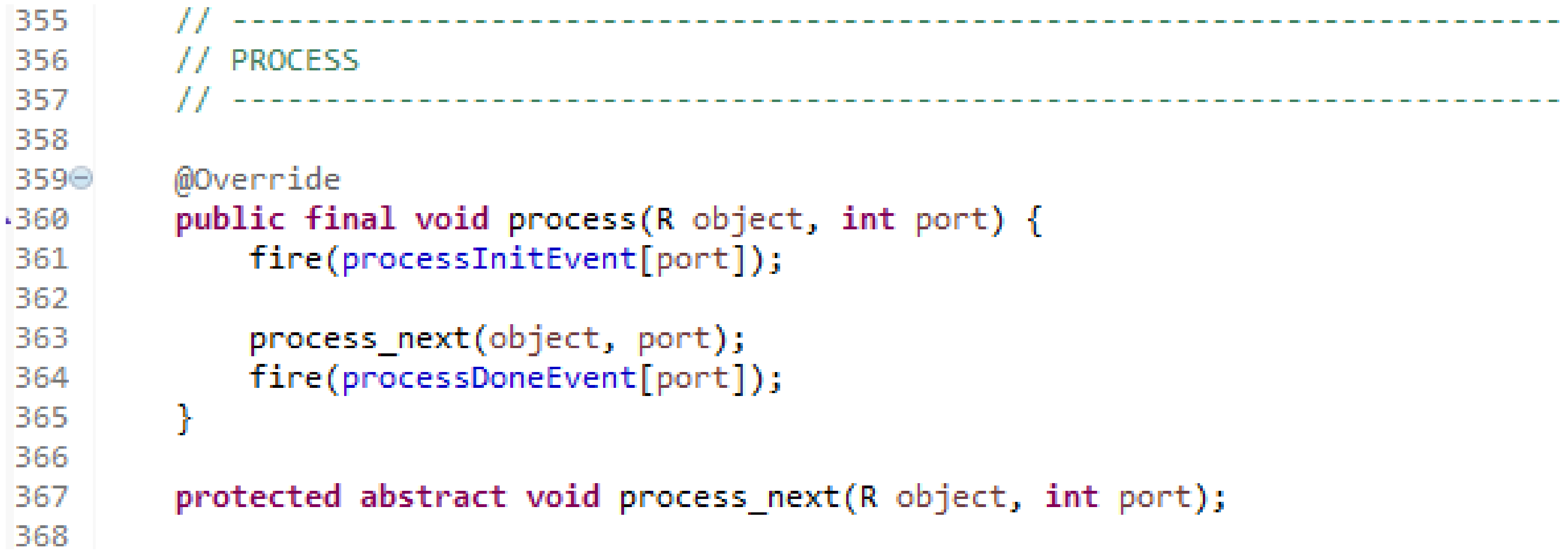


```
@SuppressWarnings({ "rawtypes" })
final public void process(IStreamObject o) {
    IStreamObject toProcess = needsClone ? (IStreamObject) o.clone() : o;
    do_process(toProcess);
}

@SuppressWarnings("rawtypes")
protected void do_process(IStreamObject o) {
    sendObject(o);
}

@SuppressWarnings({ "rawtypes" })
final protected void sendObject(IStreamObject o) {
    getSink().process(o, getSinkInPort());
}
```

- Subscription does cloning of input and sends object to output and calles process on connected input port
- ControllablePhysicalSubscription overwrites do_process to allow interruption (e.g. suspending)

Physical operators

- AbstractSink finally delegates processing of object to process_next() method

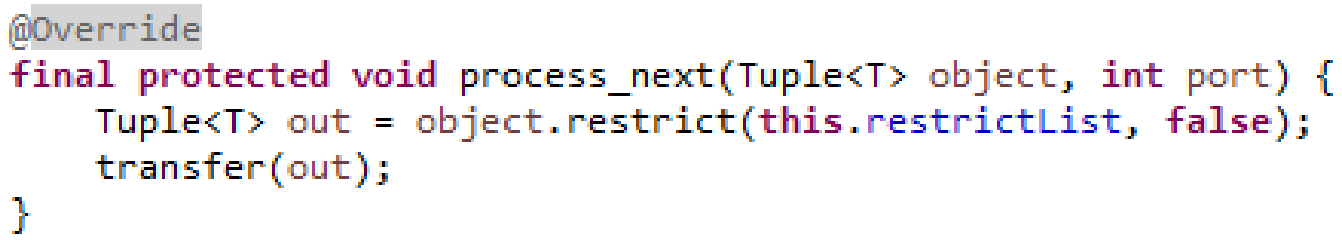


```
// ---------------------------------------------------------------------
// PROCESS
// ---------------------------------------------------------------------

@Override
public final void process(R object, int port) {
    fire(processInitEvent[port]);

    process_next(object, port);
    fire(processDoneEvent[port]);
}

protected abstract void process_next(R object, int port);

```

- … and operators can handle each incomming new object here

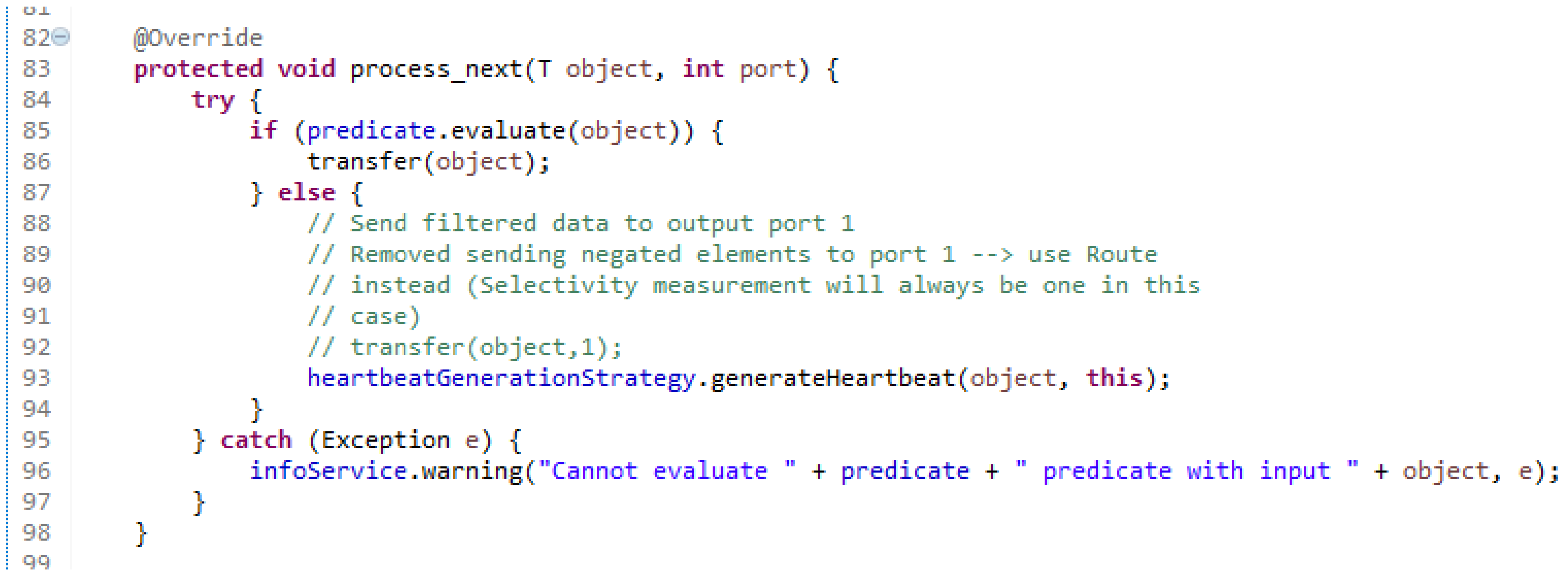
Physical operators: process_next

- RelationalProjectPO



```java
@Override
final protected void process_next(Tuple<T> object, int port) {
    Tuple<T> out = object.restrict(this.restrictList, false);
    transfer(out);
}
```

- SelectPO



```java
    @Override
    protected void process_next(T object, int port) {
        try {
            if (predicate.evaluate(object)) {
                transfer(object);
            } else {
                // Send filtered data to output port 1
                // Removed sending negated elements to port 1 --> use Route
                // instead (Selectivity measurement will always be one in this
                // case)
                // transfer(object,1);
                heartbeatGenerationStrategy.generateHeartbeat(object, this);
            }
        } catch (Exception e) {
            infoService.warning("Cannot evaluate " + predicate + " predicate with input " + object, e);
        }
    }
```

Physical operators: Punctuations and Heartbeats

- Punctuations:
  - Additional elements that can be added to the stream
  - Special handling of punctuations in own methods
  - Operators without any state can resend punctuations
  - Operators with state must assure order (→ TransferAreas)
- Heartbeats:
  - Are special punctuations
  - They state the current time progress at must be in-order
  - In out-of-order scenarios they are the only marker for time progress

Physical operators: Cloning of stream objects

- We need a way to handle object copies
- We want as less object copies as possible,
  - e.g., a following operator takes the input of an operator and creates new output in a new object → no need to clone
  - E.g., a following operator takes the input, and modifies the input and the current operator does not need the input anymore → no need to clone
  - But, if there are two following operators → one operator needs a clone of the input
- Odysseus handles cloning itself, but it needs some information from the operator
- For this, the physical operator must implement `getOutputMode`:
  - `INPUT`: elements will transfer unmodified input element (e.g. selection)
  - `MODIFIED_INPUT`: input element will be changed and transfered (e.g. projection)
  - `NEW_ELEMENT`: the operator creates a new element (e.g. join)
- Remark: In case of `NEW_ELEMENT`, meta data must be handled by the operator

TransferAreas

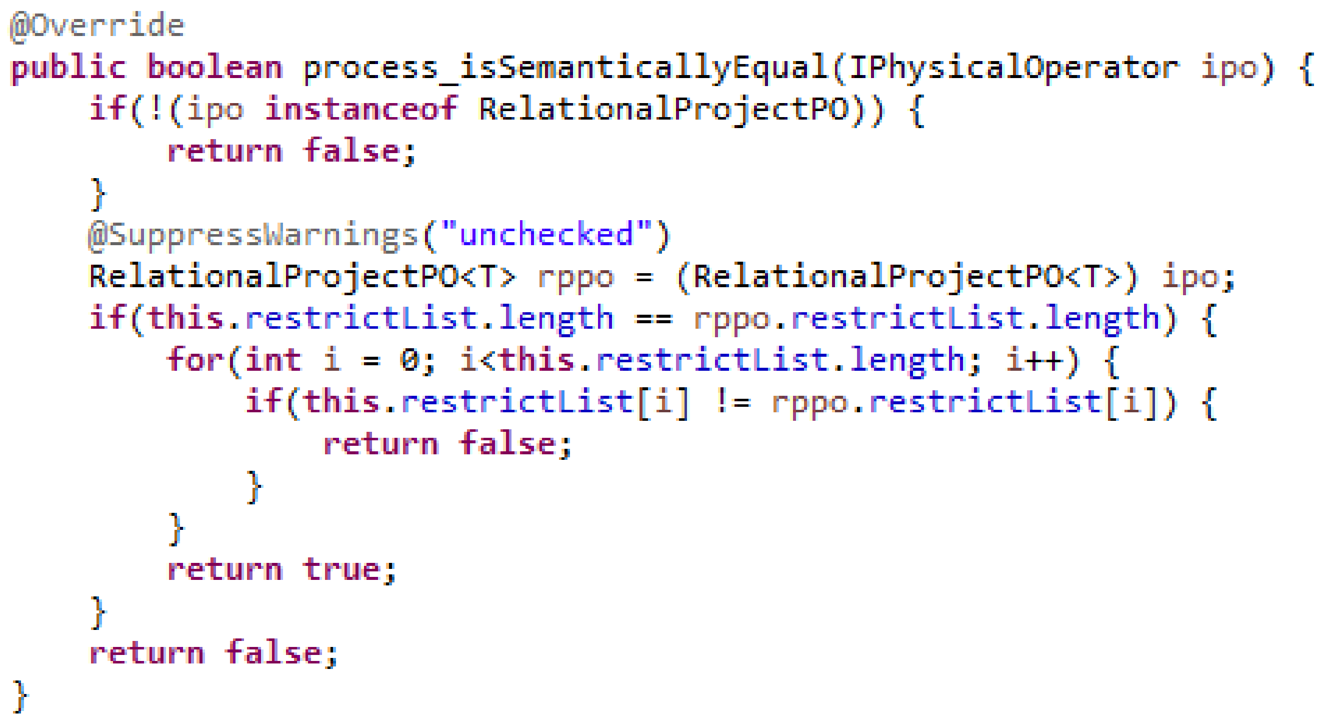
- Odysseus allows out of order processing, but this typically leads to latency overhead
- So, if the input stream is ordered, Odysseus should use the in-order-approach
- In in-order, each operator must assure the output is in-order
- To allow easier handling TransferAreas/ISyncArea can be used
- Two kinds of methods:
  - Inform area of current time state (and allow transfer to next operator)
    - `void newElement(IStreamable object, int inPort);`
    - `void newHeartbeat(PointInTime heartbeat, int inPort);`
  - Store element
    - `void transfer(W object, int toPort);`
    - `void sendPunctuation(IPunctuation punctuation, int toPort);`
- Some more, e.g. done (see later)

SweepAreas

- Sometimes, operators need a way to handle window information
- Sweep areas can be used for this
- Remove outdated elements or find matching elements

Query sharing

- A physical operator is shared, if
  1. all inputs are the same
  2. the operator is semantically equal to the other operator
- Overwrite `process_isSemanticallyEqual` to allow query sharing, default is false



```
@Override
public boolean process_isSemanticallyEqual(IPhysicalOperator ipo) {
    if(!(ipo instanceof RelationalProjectPO)) {
        return false;
    }
    @SuppressWarnings("unchecked")
    RelationalProjectPO<T> rppo = (RelationalProjectPO<T>) ipo;
    if(this.restrictList.length == rppo.restrictList.length) {
        for(int i = 0; i<this.restrictList.length; i++) {
            if(this.restrictList[i] != rppo.restrictList[i]) {
                return false;
            }
        }
        return true;
    }
    return false;
}
```

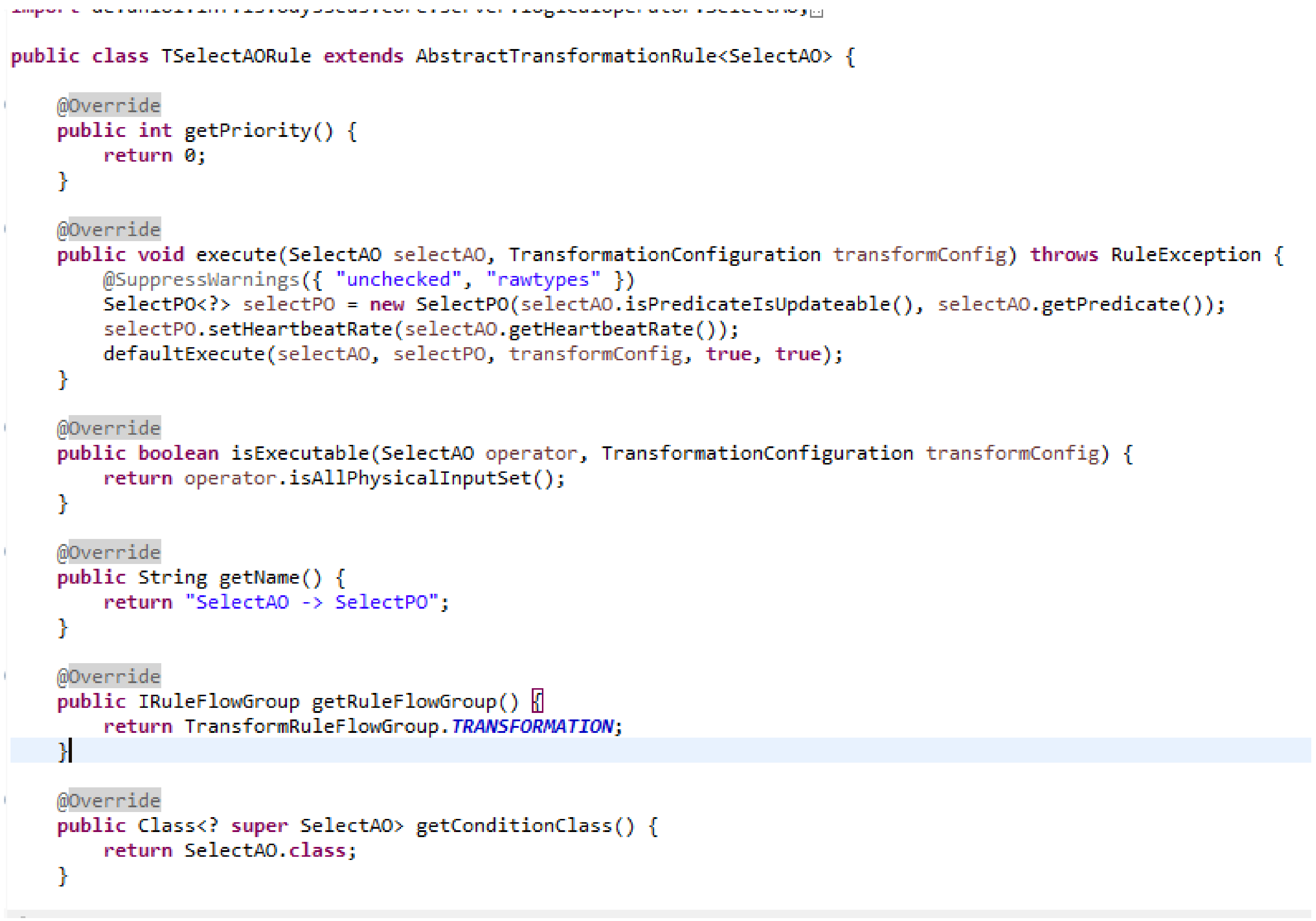
Optional: Add Rewrite Rules

- Rewrite Rules can be added to allow rewriting on logical level
- Selection push down etc.

Add Transformation Rule

- A transformation rule is used to translate a logical operator into a physical operator
- Similar to logical operators, rules are found automatically if placed in package ending with „`.rules`“
- By default, TransformationRules start with T

Example rule for selection



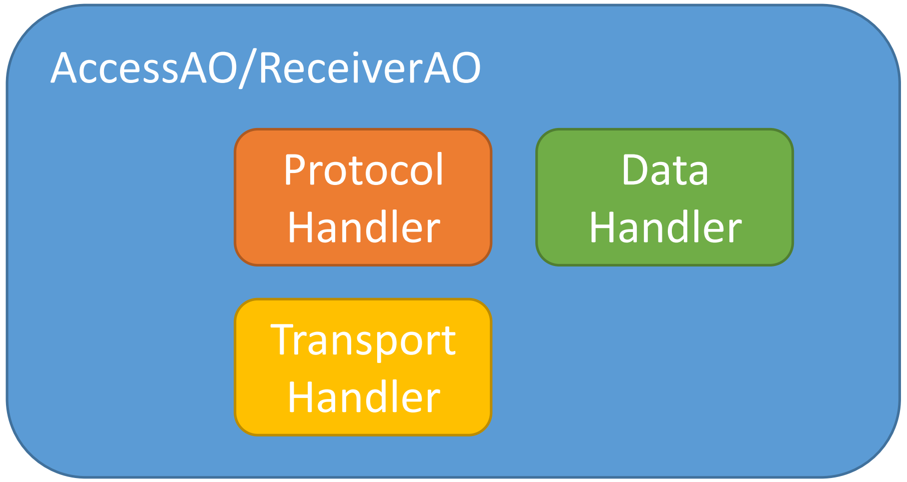
```
public class TSelectAORule extends AbstractTransformationRule<SelectAO> {

    @Override
    public int getPriority() {
        return 0;
    }

    @Override
    public void execute(SelectAO selectAO, TransformationConfiguration transformConfig) throws RuleException {
        @SuppressWarnings({ "unchecked", "rawtypes" })
        SelectPO<?> selectPO = new SelectPO(selectAO.isPredicateIsUpdateable(), selectAO.getPredicate());
        selectPO.setHeartbeatRate(selectAO.getHeartbeatRate());
        defaultExecute(selectAO, selectPO, transformConfig, true, true);
    }

    @Override
    public boolean isExecutable(SelectAO operator, TransformationConfiguration transformConfig) {
        return operator.isAllPhysicalInputSet();
    }

    @Override
    public String getName() {
        return "SelectAO -> SelectPO";
    }

    @Override
    public IRuleFlowGroup getRuleFlowGroup() {
        return TransformRuleFlowGroup.TRANSFORMATION;
    }

    @Override
    public Class<? super SelectAO> getConditionClass() {
        return SelectAO.class;
    }
```

Extensions

- Language extensions
  - Create a new language ✓
  - Create a new Odysseus Script Command (#....) ✓
  - Create a new PQL Operator ✓
- Processing function extensions
  - Create new datatypes ✓
  - Create new stream object types ✓
    - Data Handler (✓)
  - Create a new wrapper
    - Transport handler
    - Protocol handler
  - Create new functions for expressions and predicates ✓
  - Create new aggregationsfunctions ✓
  - Create new operators ✓
- Create new schedulers and scheduling strategies
- ...

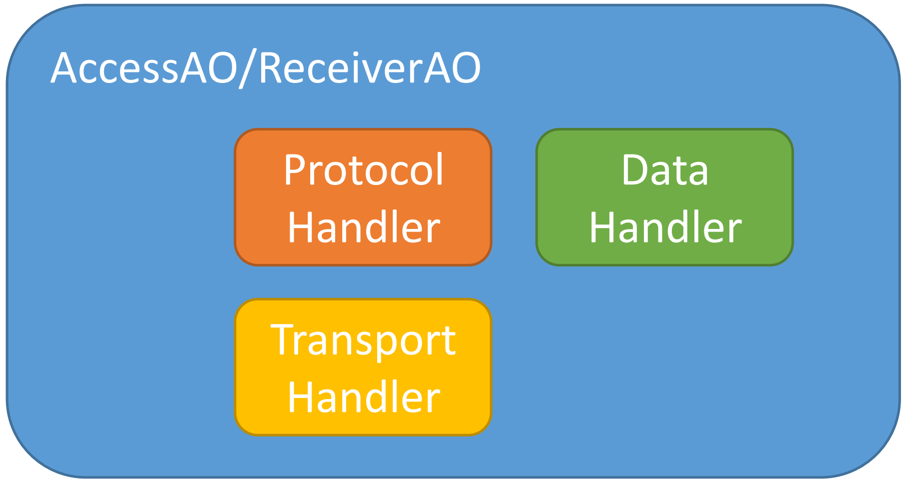
Creating Wrapper

- A wrapper is simply an operator, that accesses external data and pushes it to Odysseus
- To allow an easy creation of new wrappers Odysseus provides a wrapper framework
- A wrapper can be
  - push based, i.e. the source sends data
  - pull based, i.e. the data must be retrieved from the source (and the operator must be scheduled)
- Transport Handler:
  - Communicate with external sources (e.g. Sockets)
  - Has no knowledge about object/events
- Protocol Handler: Interprete data from Transport Handler (e.g. csv)
- Data Handler: Conversion of data from Protocol Handler to StreamObjects
- Currently, push and pulled based versions of handlers are implemented in the same class
  - Maybe we will change this in the future (but will stay compatible to the current approach)
- There are cases with only a TransportHandler (pushbased) or only a ProtocolHandler (pullbased)



Example Wrapper with PQL

```
#PARSER PQL
#RUNQUERY
wea ::= RECEIVE({
            source = 'wea',
            transport = 'tcpclient',
            datahandler = 'tuple',
            protocol = 'simplecsv',
            schema = [
              ['id', 'Integer'],
              ['timestamp', 'StartTimestamp'],
              ['load', 'Double'],
              ['location', 'SpatialPoint']
            ],
            options = [
              ['host', '123.0.1.2'],
              ['port', '1230']
            ]
          }
        )
```



Creating Transport Handler: ITransportHandler Interface

- Extends ITransportHandler and registered as OSGi Service
- Some notes:
  - The current framework is not optimal and needs some modifications
  - We plan to move to another framework where the creation of wrappers is much easier
  - We will keep this framework, too :-)
- If full implemented, a single TransportHandler class can be used for
  - pull-based reading from sources (e.g. files) (Access)
  - push-based receiving from sources (e.g. sockets) (Receive)
  - push-based writing to sources (Sender)
  - pull-based writing to sources (Sender)
- Every TransportHandler must provide a factory method to create an instance

```
ITransportHandler createInstance(IProtocolHandler<?> protocolHandler, OptionMap options);
```

  - will be called from TransportHandlerRegistry
  - needs an already initalized ProtocolHandler (special handler „None“ is available)
  - options (key,value pairs) are directly copied from query

Reading and Writing

- Methods are called from **ProtocolHandler**
- For **pull-based reading**, the transport handler needs to provide an InputStream

```
InputStream getInputStream();
```

- For **pull-based writing**, the transport handler needs to provide an OutputStream

```
OutputStream getOutputStream();
```

- For **push-based writing**, the transport handler provides some send-Messages

```
void send(byte[] message) throws IOException;
void send(String message, boolean addNewline) throws IOException;
void send(Object message) throws IOException;
```

- What about push-based receiving?
  - Must be handled inside and depends on the push based source!
  - E.g. a Netty-based TCP-Handler that receives input

Initializing and closing

- Similar to operators, the transport handler get initialized and closed
- Framework method `open` calls depending on `ExchangePattern` and `direction`:
  - InOnly, InOptionalOut, InOut and IN, INOUT: `processInOpen()`
  - OutOnly, OutOptionalIn, InOut and OUT, INOUT: `proessOutOpen()`
- Depending on type of handler, methods should be implemented
  - e.g. Opening a FileInputStream or a SocketInputStream
- Each TransportHandler instance will only be opened once (openCounter++ for each call)
- Framework method `close` calls depending on `ExchangePattern` and `direction`:
  - InOnly, InOptionalOut, InOut and IN, INOUT: `processInClose()`
  - OutOnly, OutOptionalIn, InOut and OUT, INOUT: `proessOutClose()`
- Close will be called, if transport handler is no longer needed (openCounter == 0)
- Again: Methods must be called from protocol handler (in their open-method)
  - Issue: if protocol handler overwrites open(), it must call `getTransportHandler().open();`

Done ...

- Typically, a query in a streaming system runs forever ...
- But sometimes (especially for evaluations), the query should be closed when all data is processed (e.g. from a file)
- For this, Odysseus uses a done flag
- TransportHandler:
  - done ist set to false, when open is called
  - can be set to true, if no more data is available
- Will lead to propagation of done in the query plan (process_done()) and this will lead to close call from query

TransportHandler

- Typically, a TransportHandler can be created by extending one of the following abstract base classes:
  - AbstractTransportHandler (one for all kinds)
  - AbstractPullTransportHandler
  - AbstractSimplePullTransportHaldler
  - AbstractPushTransportHandler
  - AbstractFileHandler (when reading from files)
- For cases, where this is not possible (e.g. Base class needs to be extended) the
  - AbstractTransportHandlerDelegate can be used
  - Example: ProtobufServerTransportHandler

Protocol Handler

- While transport handler are used to connect to out side sources, protocol handler handle the content
- Again, this handler can be used for reading (pull),

```
boolean hasNext() throws IOException;
T getNext() throws IOException;
```

- receiving (push) as ITransportHandlerListener

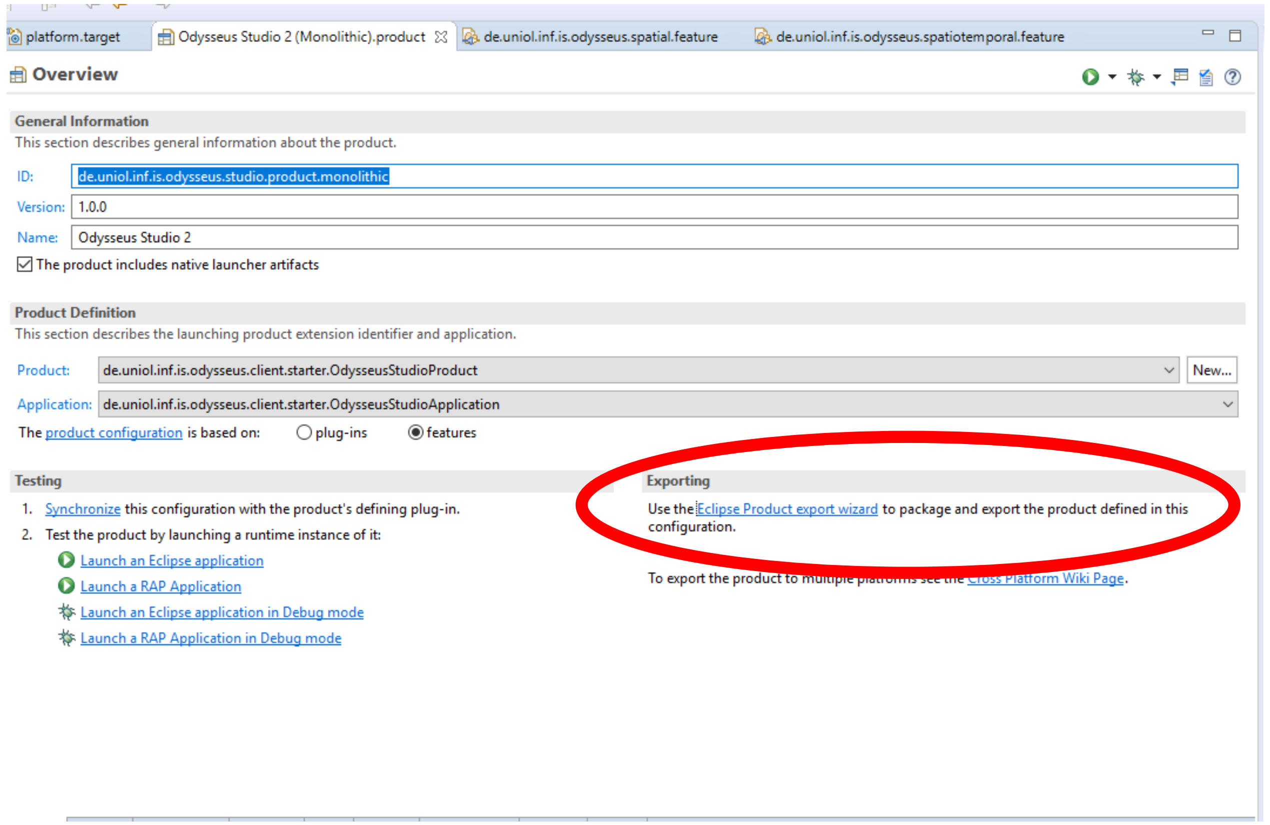
```
void process(long callerId, ByteBuffer message);
void process(InputStream message);
void process(String[] message);
void process(String message);
void process(T m, int port);
void process(T m);
```

- and writing (push)

```
void write(T object) throws IOException;
void writePunctuation(IPunctuation punctuation) throws IOException;
```

Deployment

- Maven/Tycho
  - Currently, we are trying to switch from pure eclipse to maven
  - First success this week … but takes some times ;-)
- Eclipse
  - If you only want to deploy for the current operating system:



platform.target | Odysseus Studio 2 (Monolithic).product | de.uniol.inf.is.odysseus.spatial.feature | de.uniol.inf.is.odysseus.spatiotemporal.feature

Overview

General Information
This section describes general information about the product.

ID: de.uniol.inf.is.odysseus.studio.product.monolithic
Version: 1.0.0
Name: Odysseus Studio 2
☑ The product includes native launcher artifacts

Product Definition
This section describes the launching product extension identifier and application.

Product: de.uniol.inf.is.odysseus.client.starter.OdysseusStudioProduct New...
Application: de.uniol.inf.is.odysseus.client.starter.OdysseusStudioApplication
The product configuration is based on: ○ plug-ins ● features

Testing
1. Synchronize this configuration with the product's defining plug-in.
2. Test the product by launching a runtime instance of it:
   - Launch an Eclipse application
   - Launch a RAP Application
   - Launch an Eclipse application in Debug mode
   - Launch a RAP Application in Debug mode

Exporting
Use the Eclipse Product export wizard to package and export the product defined in this configuration.

To export the product to multiple platforms see the Cross Platform Wiki Page.

Tycho/Maven

- Tycho is a maven plugin to allow builds for eclipse based artifacts (plugins, features, etc.)
- No need to define pom for each artifact → pomless build
  - Create folder .mvn and add file extensions.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
  <extension>
    <groupId>org.eclipse.tycho.extras</groupId>
    <artifactId>tycho-pomless</artifactId>
    <version>1.2.0</version>
  </extension>
</extensions>
```

- Typical, a releng folder for configuration, e.g. to declare tycho plugin and some other things

Releng pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>de.uniol.inf.is.odysseus_core</groupId>
        <artifactId>de.uniol.inf.is.odysseus_core.tycho.configuration</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <packaging>pom</packaging>
        <properties>
                <tycho.version>1.2.0</tycho.version>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        </properties>
```

Releng pom.xml

```
<build>
	<plugins>
		<plugin>
			<groupId>org.eclipse.tycho</groupId>
			<artifactId>tycho-maven-plugin</artifactId>
			<version>${tycho.version}</version>
			<extensions>true</extensions>
		</plugin>
		<plugin>
			<groupId>org.eclipse.tycho</groupId>
			<artifactId>tycho-packaging-plugin</artifactId>
			<executions>
				<execution>
					<phase>package</phase>
					<id>package-feature</id>
					<configuration>
<finalName>${project.artifactId}_${unqualifiedVersion}.${buildQualifier}</finalName>
					</configuration>
				</execution>
			</executions>
		</plugin>
```

Releng.pom

```
<plugin>
                                        <groupId>org.eclipse.tycho</groupId>
                                        <artifactId>target-platform-configuration</artifactId>
                                        <version>${tycho.version}</version>
                                        <configuration>
                                                  <target>
                                                            <artifact>
                                                                      <groupId>de.uniol.inf.is.odysseus_core</groupId>
                                                                      <artifactId>platform</artifactId>
                                                                      <version>1.0.0-SNAPSHOT</version>
                                                            </artifact>
                                                  </target>

                                                  <environments>
                                                            <environment>
                                                                      <os>linux</os>
                                                                      <ws>gtk</ws>
                                                                      <arch>x86</arch>
                                                            </environment>
                                                            …
                                                  </environments>
                                        </configuration>
                              </plugin>

                    </plugins>
          </build>
</project>
```

Poms for artifacts

- Depending on the folder structure, different poms are necessary
- For each subfolder without a project, something like

```
<project>
        <modelVersion>4.0.0</modelVersion>
        <groupId>de.uniol.inf.is.odysseus_core</groupId>
        <artifactId>de.uniol.inf.is.odysseus_core.releng</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <packaging>pom</packaging>

        <parent>
                <groupId>de.uniol.inf.is.odysseus_core</groupId>
                <artifactId>de.uniol.inf.is.odysseus_core.root</artifactId>
                <version>1.0.0-SNAPSHOT</version>
        </parent>

        <modules>
                <module>de.uniol.inf.is.odysseus_core.update</module>
        </modules>
</project>
```

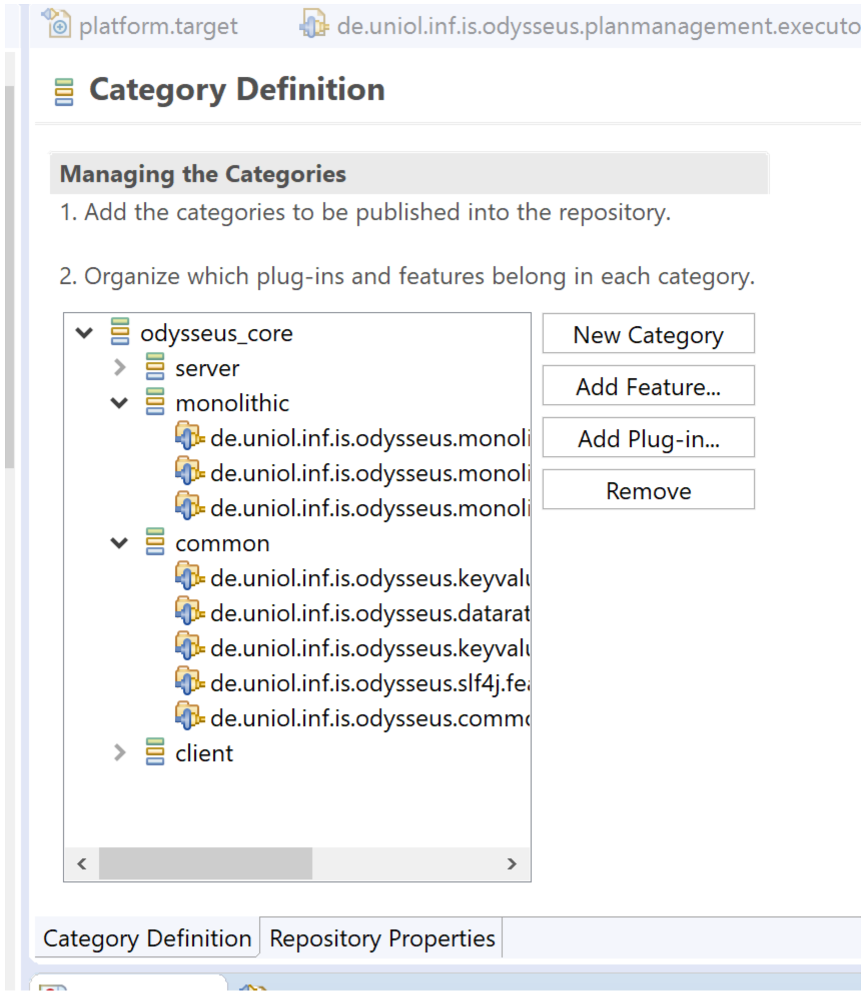
Especially, on top level

```
<project>
	<modelVersion>4.0.0</modelVersion>
	<groupId>de.uniol.inf.is.odysseus_core</groupId>
	<artifactId>de.uniol.inf.is.odysseus_core.root</artifactId>
	<version>1.0.0-SNAPSHOT</version>
	<packaging>pom</packaging>
	<parent>
		<groupId>de.uniol.inf.is.odysseus_core</groupId>
		<artifactId>de.uniol.inf.is.odysseus_core.tycho.configuration</artifactId>
		<version>1.0.0-SNAPSHOT</version>
		<relativePath>./releng/de.uniol.inf.is.odysseus_core.tycho.configuration</relativePath>
	</parent>

	<modules>
		<module>common</module>
		<module>targetplatform</module>
		<module>resource</module>
		<module>client</module>
		<module>server</module>
		<module>monolithic</module>
		<module>releng</module>
		</modules>
</project>
```

Creating update site

- Category.xml necesary
- Create default project (with eclipse) and create a new category definition
- Add features (and optionally plugins)
- Only features that are defined here, will be part of the update site

- See odysseus_core repository for an example (not working yet …)



platform.target de.uniol.inf.is.odysseus.planmanagement.executo

Category Definition

Managing the Categories

1. Add the categories to be published into the repository.

2. Organize which plug-ins and features belong in each category.

```
odysseus_core
  server
  monolithic
    de.uniol.inf.is.odysseus.monoli
    de.uniol.inf.is.odysseus.monoli
    de.uniol.inf.is.odysseus.monoli
  common
    de.uniol.inf.is.odysseus.keyvalu
    de.uniol.inf.is.odysseus.datarat
    de.uniol.inf.is.odysseus.keyvalu
    de.uniol.inf.is.odysseus.slf4j.fe
    de.uniol.inf.is.odysseus.comm
  client
```

Creating update site pom

```
<project>
		<modelVersion>4.0.0</modelVersion>
		<parent>
			<groupId>de.uniol.inf.is.odysseus_core</groupId>
			<artifactId>de.uniol.inf.is.odysseus_core.releng</artifactId>
			<version>1.0.0-SNAPSHOT</version>
			<relativePath>../pom.xml</relativePath>
		</parent>

		<artifactId>de.uniol.inf.is.odysseus_core.update</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>eclipse-repository</packaging>

</project>
```

Extensions

- Language extensions
  - Create a new language ✓
  - Create a new Odysseus Script Command (#....) ✓
  - Create a new PQL Operator ✓
- Processing function extensions
  - Create new datatypes ✓
  - Create new stream object types ✓
    - Data Handler ✓
  - Create a new wrapper ✓
    - Transport handler ✓
    - Protocol handler ✓
  - Create new functions for expressions and predicates ✓
  - Create new aggregationsfunctions ✓
  - Create new operators ✓
- Create new schedulers and scheduling strategies
- Create new meta data
- …

Time for questions and discussion

Quelle: http://www.entspannt-lernen.de/fragen-antworten.html

Odysseus
the event processing system