Assignment Portfolio 1

Kevin K. M. Petersen Eksamens nr: 480900

April 7, 2021

1 Introduktion

Link til kildekode github.com/kepet19/komponent_systemer
Formålet med opgaven er at komme igennem de forskellige komponent basseret arkitektur. De forskellige komponent arkitektur er:

- whiteboard modellen (Den der er standard i java)
- netbeans module system
- OSGi Declarative Services & Bundle Context API

2 Javalab

I denne opgave skulle man lave et ekstra komponent som kunne loades ind automatisk uden at blive tilføjet manuelt som dependencies i java filerne.

2.1 Register komponent

først lave jeg en ny "Enemy" komponent som er baseret på player, da mange af tingende er de samme.
Derefter tilføjede jeg Enemy til rodens/root "./pom.xml" filen, hvor den kom til at stå som et ekstra modul så maven vidste den skulle compiles med. Derefter tilføjede jeg "Enemy" til dependencies inde i "Core/pom.xml". så java kunne finde de interfaces som vi ledte efter. En af de key items er at man skal have en "resource" mappe med hvor er er en "META-INF" mappe og i den mappe har man en fil med navnet på det interface man siger man implementere. inde i filen siger man hvilke klasse der implementere dette interface.
Her har jeg en træ structur af hvordan filerne ser ud.

```
Enemy
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── dk.sdu.mmmi.cbse.enemysystem
    │           ├── Enemy.java
    │           ├── EnemyControlSystem.java
    │           └── EnemyPlugin.java
    └── resources
```

```
└── META-INF
    ├── navent på det interface man siger man implementere
    ├── dk.sdu.mmmi.cbse.common.services.IEntityProcessingService
    └── dk.sdu.mmmi.cbse.common.services.IGamePluginService
```

```
1 dk.sdu.mmmi.cbse.enemysystem.EnemyControlSystem
```

Listing 1: dk.sdu.mmmi.cbse.common.services.IEntityProcessingService

„„

```
1 dk.sdu.mmmi.cbse.enemysystem.enemyplugin
```

Listing 2: dk.sdu.mmmi.cbse.common.services.igamepluginservice

2.2 Hvordan systemet bruger den ny komponent

Java søger søger i Core modulet efter dependencies der implementere nogle besteme interfaces vi har valgt.

3 NetBeansLab1

I denne ogpave skal man lave et netbean module som kan loades ind via ”Netbeans module system”. Der er allered givet example projekt, som jeg har udvidet på.

3.1 Register komponent

For at registere et komponent skal man først lave et netbeans module som skal lægges ind som dependenci inde i ”application/pom.xml”, så netbeans module system ved at den skal loade modulet som der er blevet lavet.

```
Astroid6Shapes
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── dk.sdu.mmmi.cbse.astroid6shapes
    │           ├── AsteroidPlugin.java
    │           ├── AsteroidProcessor.java
    │           └── AsteroidSplitterImpl.java
    ├── nbm
    │   └── manifest.mf
    └── resources
        └── dk.sdu.mmmi.cbse.astroid6shapes
            └── Bundle.properties
```

I **Listing 3** ses implementation på klassen **AsteroidPlugin.java**. På linje 6 ses en decorator der fortæller netbeanas module system at dette er en klasse som provider IGamePluginService interfacet.

```
1  package dk.sdu.mmmi.cbse.astroid6shapes;
2
3  import dk.sdu.mmmi.cbse...*;
4  import org.openide.util.lookup.ServiceProvider;
5
6  @ServiceProvider(service = IGamePluginService.class)
7  public class AsteroidPlugin implements IGamePluginService{
8          @Override
9      public void start(GameData gameData, World world) {
10             ...
11     }
12
13     @Override
14     public void stop(GameData gameData, World world) {
15                 world.getEntities(Astroid6shapes.class).forEach(asteroid
                       -> {
16                         world.removeEntity(asteroid);
17                 });
18     }
19 }
```

Listing 3: AsteroidPlugin.java

3.2 Netbeans module system lille og stor

Netbeans module system har 2 forskellige systemer en slim uden noget grafisk, kun til at loade netbeans modules med og en stor hvor man får netbeans grafiske interface med.

4 NetBeansLab2

Netbeans lab 2 skal man vise at netbeans module system kan loade og undeloade moduler i runtime/kørslen af programmet. Der der allerede er lavet et modul "SilentUpdate" som kan søger for at installere og afinstallér moduler i runtime. Det modul smider man så som en dependecy inde i "application/pom.xml" filen.

4.1 Register komponent

Denne her gang fjerner jeg det module jeg lavede i netbeans lab 1 "Astroid6Shapes" dependecy fra "application/pom.xml" for at vise den kan bliver loadet ind med SilentUpdate.

```
SilentUpdate
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── org.netbeans.modules.autoupdate.silentupdate
    │           ├── UpdateActivator.java
    │           └── UpdateHandler.java
    ├── nbm
    │   └── manifest.mf
    └── resources
        └── org.netbeans.modules.autoupdate.silentupdate
            └── resources
                ├── Bundle.properties
                └── layer.xml
```

I filen "Bundle.properties" skal man pege på ens update center. jeg har valgt at bruge "target" mappen i application mappen som "update site".
Her under ses den bundle.properties. Linje 7 hvor man kan ændre hvor "netbeans site" peger til. Jeg har dog valgt at ændre navnet til kort et. da det ikke ville værer pænt med full path til filen.

```
1 #Tue Mar 10 14:13:59 CET 2015
2 Services/AutoupdateType/
    org_netbeans_modules_autoupdate_silentupdate_update_center.instance=
    Sample Update Center
3 OpenIDE-Module-Display-Category=Infrastructure
4 OutputLogger.Grain=VERBOSE
5 OpenIDE-Module-Name=Silent Update
6 OpenIDE-Module-Short-Description=Silent Update of Application
7 org_netbeans_modules_autoupdate_silentupdate_update_center=file:///target
    /netbeans_site/updates.xml
8 OpenIDE-Module-Long-Description=A service installing updates of your
    NetBeans Platform Application with as few as possible user's
    interactions.
```

Listing 4: Bundle.properties

4.2 Netbeans site

Netbeans site modulerne består af information om hvad de vil have, hvad de giver af service til andre og de indeholder også java byte code (jar filer).

4.3 Komponentdet bliver loadet og undloadet via silent update

Se video: youtu.be/H2Y0cPQ0fzc

5 OSGiLab

I OSGI lab skal der laves 2 ny moduler, fordi der skal laves en med OSGi Declarative Services og en med OSGi BundeContext API.

5.1 OSGi Declarative Services

I OSGi declarative services bruger vi en "osgi.bnd" til at fortælle OSGi hvilke activationpolicy der skal værer. Den fortæller også noget om hvor den kan finde noget om de service der bliver leveret fra modulet se **Listing 5**.

```
Player
├── osgi.bnd
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── dk.sdu.mmmi.cbse.playersystem
    │           ├── Player.java
    │           ├── PlayerControlSystem.java
    │           └── PlayerPlugin.java
    └── resources
        └── META-INF
            ├── entityprocessor.xml
            └── gameplugin.xml
```

```
1 Bundle-SymbolicName: Player
2 Bundle-ActivationPolicy: lazy
3 Service-Component: META-INF/entityprocessor.xml, META-INF/gameplugin.xml
```

Listing 5: osgi.bnd

I **Listing 6** ses på linje 3 at det er implementation klassen som den peger på og den implementere interfaces som er listest på linje 5.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="dk.
    sdu.mmmi.cbse.playersystem.plugin">
3     <implementation class="dk.sdu.mmmi.cbse.playersystem.PlayerPlugin"/>
4     <service>
5         <provide interface="dk.sdu.mmmi.cbse.common.services.
              IGamePluginService"/>
6     </service>
7 </scr:component>
```

Listing 6: gameplugin.xml

5.2 OSGI BundleContext API

```
Enemy2
├── pom.xml
└── src
    └── main
        └── java
            └── dk.sdu.mmmi.cbse.osgienemy2
                ├── Activator.java
                ├── Enemy2.java
                ├── Enemy2Plugin.java
                └── Enemy2Processor.java
```

5.3 Register komponent

I **Listing 7** ses den pom.xml som peger på den klasse som har implementeret bundle activator.

```
1   <build>
2       <plugins>
3           <plugin>
4               <groupId>org.apache.felix</groupId>
5               <artifactId>maven-bundle-plugin</artifactId>
6               <extensions>true</extensions>
7               <configuration>
8                   <instructions>
9                       <Bundle-Activator>dk.sdu.mmmi.cbse.osgienemy2.
                            Activator</Bundle-Activator>
10                  </instructions>
11              </configuration>
12          </plugin>
13      </plugins>
14  </build>
```

Listing 7: pom.xml

I **Listing 8** ses den klasse som har implementeret bundle activator.

```
1  package dk.sdu.mmmi.cbse.osgienemy2;
2
3  import dk.sdu.mmmi.cbse.common.services.IEntityProcessingService;
4  import dk.sdu.mmmi.cbse.common.services.IGamePluginService;
5  import org.osgi.framework.BundleActivator;
6  import org.osgi.framework.BundleContext;
7
8  public class Activator implements BundleActivator {
9
10     @Override
11     public void start(BundleContext context) throws Exception {
12
13         context.registerService(IGamePluginService.class, new
               Enemy2Plugin(), null);
14         context.registerService(IEntityProcessingService.class, new
               Enemy2Processor(), null);
15     }
```

```
16
17      public void stop(BundleContext context) throws Exception {
18      }
19
20  }
```

Listing 8: Activator.java

5.4 Komponentdet bliver loadet og undloadet via Apache gogo shell

Se video youtu.be/nB84hmut8W0