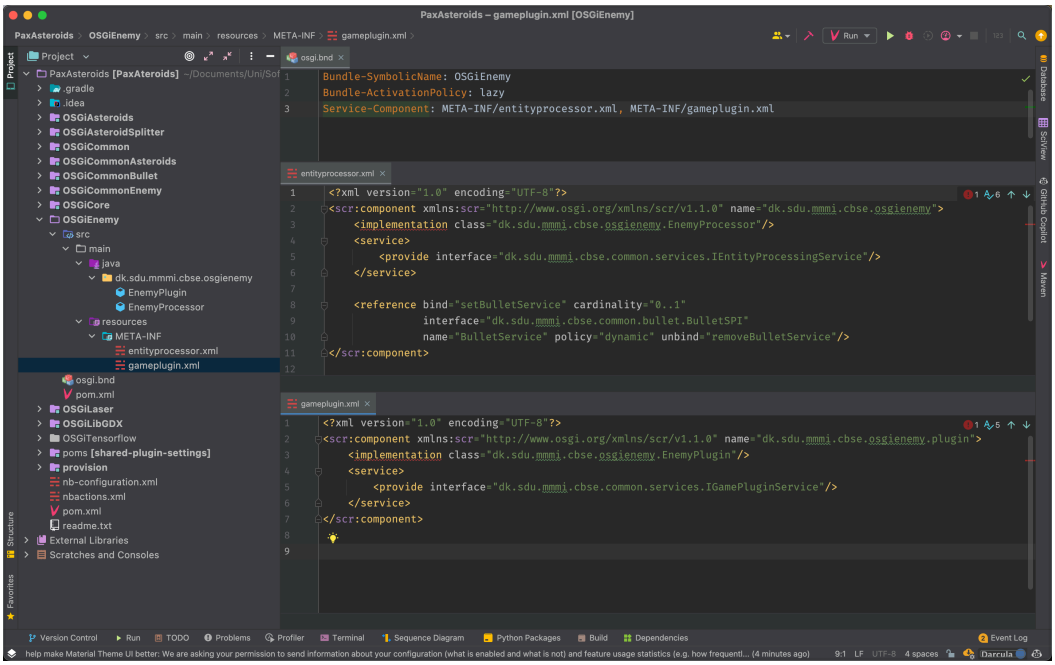
# Component-based systems - Assignment 2

You should also create a more through explanation on how you have created your test.

## OSGiLab with PaxAsteroids

Modules are stated and declared in the projects pom.xml. OSGi components are then declared using the manifest file sometimes being helped with the OSGi bnd file for each module. In the case of the enemy bundle, the osgi.bnd points to service components in the meta-inf folder that specifies the service components entityprocessor.xml and gameplugin.xml. Bnd helps create the required manifest from the classes and service components. We can see that the enemy has a dynamic dependency injection with the setBulletService from the declared services in the entityprocessor.xml.




The game can then be run by creating a maven configuration in IntelliJ and writing install pax:provision in the command line parameter.
In order for PaxAsteroids to run I had to out-comment a line in the pom.xml, namely:

```
<param>&#45;&#45;vmOptions=-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=5005</param>
```

To unload and load the modules while the game is running we use the following commands: lb, start ID, stop ID.

g! lb lists all the bundles and we can start and stop bundles with their respective IDs as shown below. Stopping bundle number 11 removes the enemy from the game screen, and starting it again starts the

```
g! lb
ID|State      |Level|Name
    0|Active     |    0|System Bundle (5.4.0)|5.4.0
    1|Active     |    1|Apache Felix Gogo Command (0.16.0)|0.16.0
    2|Active     |    1|Apache Felix Gogo Runtime (0.16.2)|0.16.2
    3|Active     |    1|Apache Felix Gogo Shell (0.12.0)|0.12.0
    4|Active     |    5|Apache Felix Declarative Services (1.8.0)|1.8.0
    5|Active     |    5|OSGiLaser (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
    6|Active     |    5|OSGiLibGDX (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
    7|Active     |    5|osgi.core (4.3.0.201102171602)|4.3.0.201102171602
    8|Active     |    5|OSGiCore (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
    9|Active     |    5|OSGiAsteroidSplitter (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
   10|Active     |    5|OSGiAsteroids (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
   11|Active     |    5|OSGiEnemy (1.0.0.SNAPSHOT)|1.0.0.SNAPSHOT
   12|Resolved   |    5|com.diffplug.osgi.extension.sun.misc (0.0.0)|0.0.0
   13|Resolved   |    5|com.diffplug.osgi.extension.sun.reflect (0.0.0)|0.0.0
g! stop 11
g! start 11
```

A module can also be started with an Activator class specified in the manifest file, for example, the Bundle-Activator in the OSGiLaser module.

## SpringLab with AsteroidsServiceLoader

Started with a new JavaLab project with AsteroidsServiceLoader. The following beans files that we register are:

AsteroidBeans.xml
CollisionBeans.xml
CoreBeans.xml
PlayerBeans.xml

In the XML files we have the following IDs that are tied to their class path.

asteroidControlSystemBean
asteroidPluginBean
colliderBean
playerControlSystemBean
playerPluginBean

They are then used in Game.java with application context to call upon the modules.

```
public class Game implements ApplicationListener {
  ...
ApplicationContext context = new ClassPathXmlApplicationContext("CoreBeans.xml");
  ...
  }


  public void create( ) {
  ...
    IGamePluginService asteroidPlugin = (AsteroidPlugin) context.getBean("asteroidPluginBean");
    IEntityProcessingService asteroidService = (AsteroidControlSystem)
context.getBean("asteroidControlSystemBean");
    pluginProcessors.add(asteroidPlugin);
    entityProcessors.add(asteroidService);

    IPostEntityProcessingService collidorPlugin = (Collider) context.getBean("colliderBean");
    postEntityProcessors.add(collidorPlugin);

    IGamePluginService playerPlugin = (PlayerPlugin) context.getBean("playerPluginBean");
    IEntityProcessingService playerService = (PlayerControlSystem)
context.getBean("playerControlSystemBean");
    pluginProcessors.add(playerPlugin);
    entityProcessors.add(playerService);

    ...
```

## TestLab with AsteroidsServiceLoader

A simple JUnit test on the collision module was made in the AsteroidsServiceLoader. Right clicking the module and saying new JUnit test on an existing class gives us a nice template to start from. A new entity needed both a position part and radius and then the test ran without issue.