Workshop

Inversion of Control and Dependency injection in Spring.

Your workshop is to build a console-based student management system in Spring framework by making the **Spring Container** build the *StudentManagement* bean for you. We could do this in a lot of different ways with XML, Java code or annotations. We are going to use a combination of annotations and Java configured beans.

Part 1

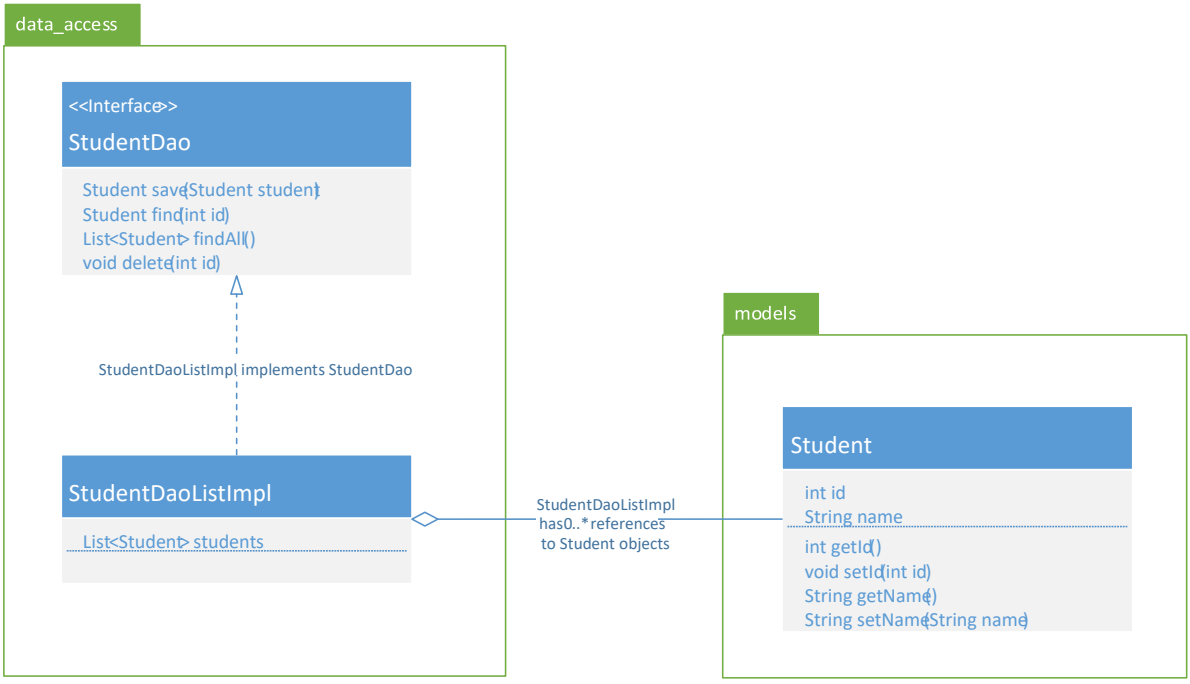
1. Create a package called *data_access*.
2. Create the interface *StudentDao* inside *data_access*.
3. Create the class *StudentDaoListImpl* inside *data_access*.
4. Make *StudentDaoListImpl* implement *StudentDao*.
5. Create a new package called models.
6. Create the *Student* class inside the models-package
7. Annotate *StudentDaoListImpl* class with **@Component**
8. Configure component scanning by doing the following:
    - Create a new configuration class. (“ComponentScanConfig” would be a good name).
    - Annotate your configuration class with **@Configuration**
    - Add **@ComponentScan**(“packagename”) to the configuration class

9. In your main method write this:

```
AnnotationConfigApplicationContext context =
        new AnnotationConfigApplicationContext(ConfigFileName.class);

StudentDao studentDao = context.getBean(StudentDao.class);
```

If you compile the code without Exceptions go ahead and make a commit to a Repository.



Part 2

Having created and configured a *StudentDao* bean we move on and configure another dependency:

ScannerInputService implements *UserInputService*. *ScannerInputService* need to have a *Scanner* object injected.

To make a *Scanner* object "injectable" you need to configure the object to be read in by the Spring Container. The way to do this is to create another configuration class annotated with **@Configuration** and in this class create a method annotated with **@Bean** that returns a *Scanner* object to be injected. You could also add a method to your already existing configuration class.

When you have defined your *Scanner* bean, a Scanner object is ready to be injected using Constructor injection, Setter injection or Field injection using **@Autowired**.

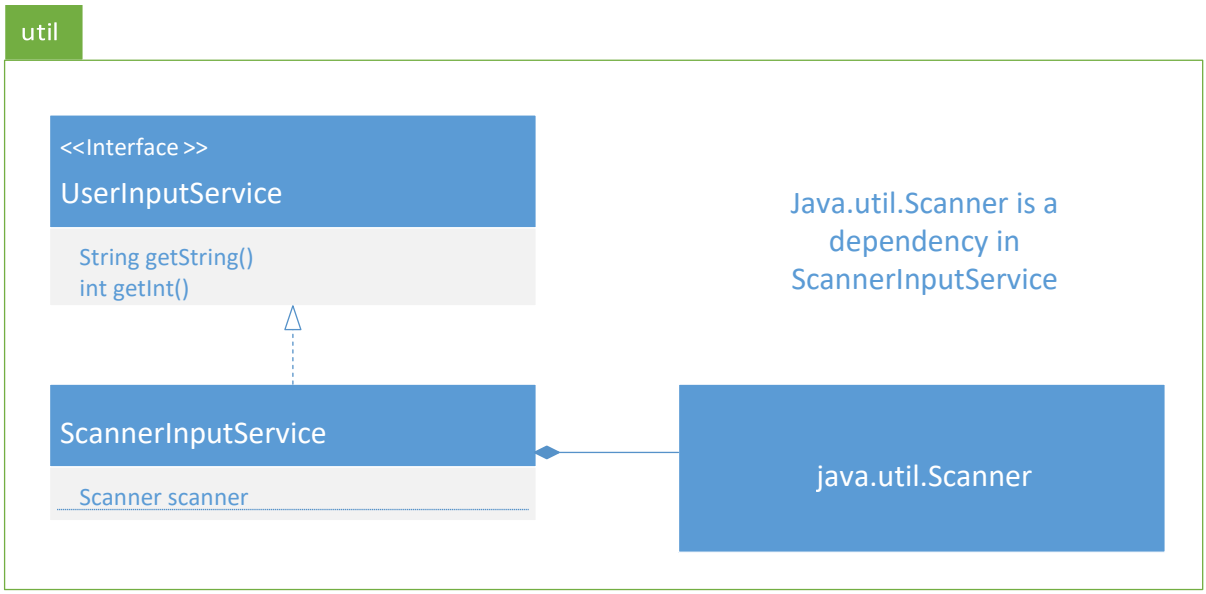
Don't forget to annotate *ScannerInputService* class with **@Component**.

Test your dependency by writing this in your main method:

```
AnnotationConfigApplicationContext context =
        new AnnotationConfigApplicationContext(ConfigFileName.class);

UserInputService userInputService =context.getBean(UserInputService.class);
```

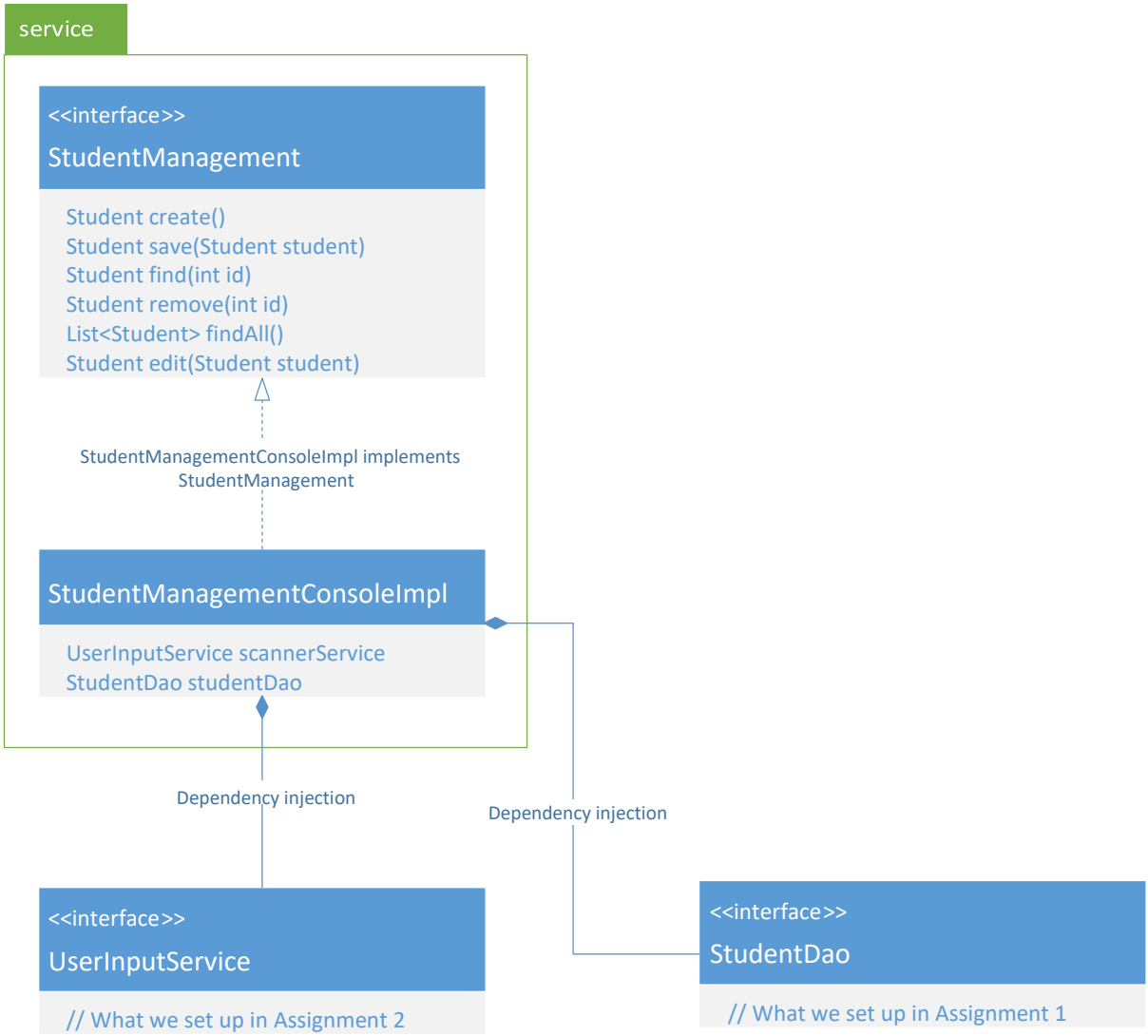
If no exception is throwed commit to your repo.



Part 3

Now when you have created StudentDao dependency in part 1 and UserInputService in part 2, you are ready to inject both dependencies in another class.

You need to create StudentManagementConsoleImpl that implements the interface StudentManagement first, then I think you can manage the rest. 😉



Complete Class diagram