Task Manager, OOAD Project

1. Nabil Ettachfini

2. Project Description:
    a. An application that allows a manager to login and create a task for their employees to complete. The employee can sign in and view the tasks and set the status of completion. The manager can then view all the tasks and the new status the employee set.

3. Features that were implemented

ID	Actor	Requirements
01	Manager and Employee	Employees/ Managers can sign up and then they can login once they sign-up
02	Employee	Employee enters what kind of employee they are after signing-up.
03	Manager and Employee	Managers/Employees can login, to access the necessary features
04	Manager	The manager can create a task. This task would have details accompany it. Such details would be completion time, difficulty, name of employee assigned to etc.
04-a	Manager	Managers can enter the name of the employee they would like to work on the task.
05	Manager	Manager can then add more specific tasks based off of the type of employees the task is for.
06	Employee	An employee can view task they manager has created.
07	Employee	The employee can set a completion status on the task.

08	Manager	Manager can view new completion status.
09	Manager	Manager can then print all tasks and see the completion on each task.
10	Employee	Can view all tasks and the completion status on each task
11	Manager	Manager can delete tasks.

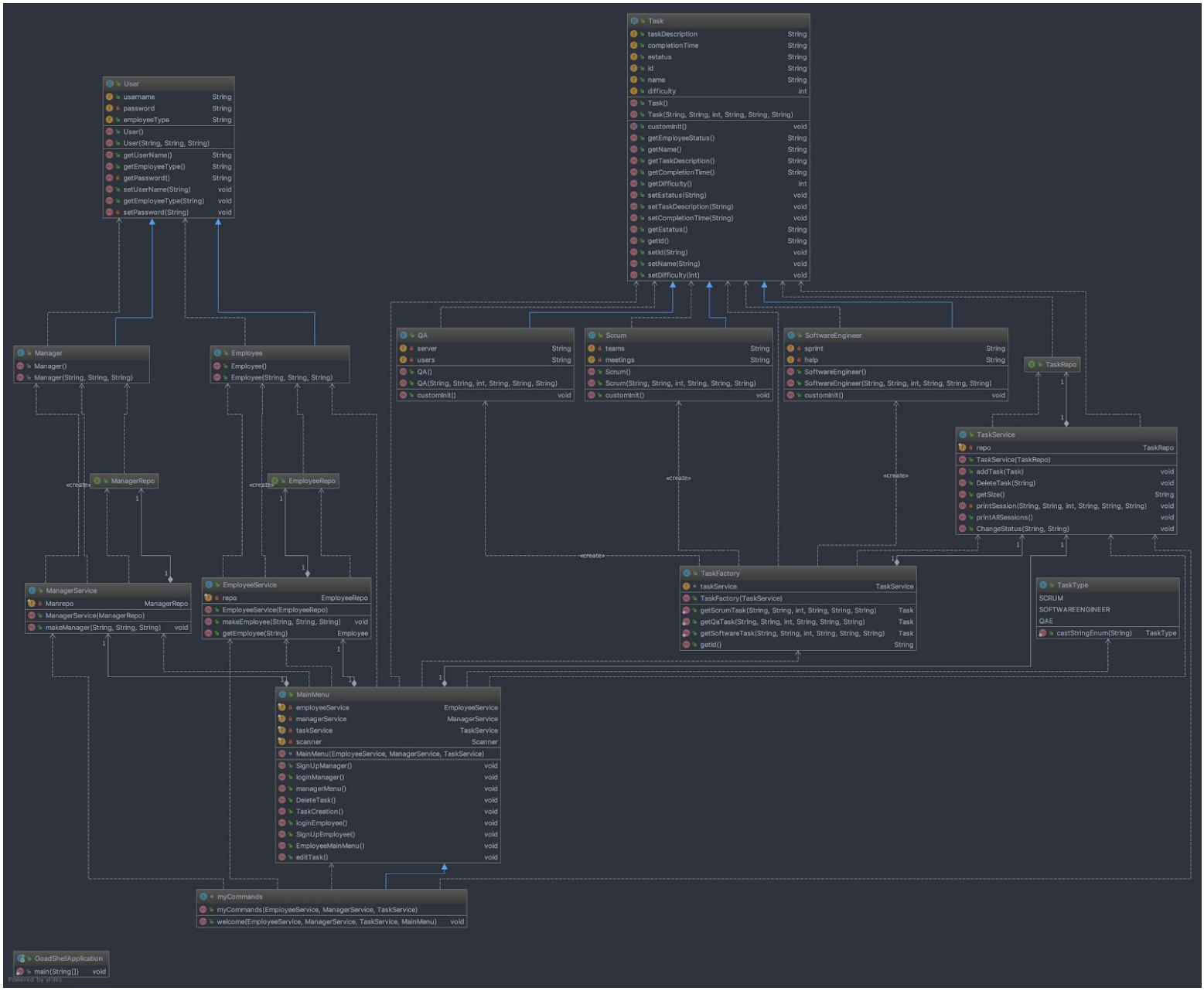
4.

Features that were not implemented

Id	Actor	Requirements
01	Manager/Employee	Did not implement a time tracker on tasks.
02	Manager	Did not implement the progress tracking on the project. This was removed from the scope of the project.

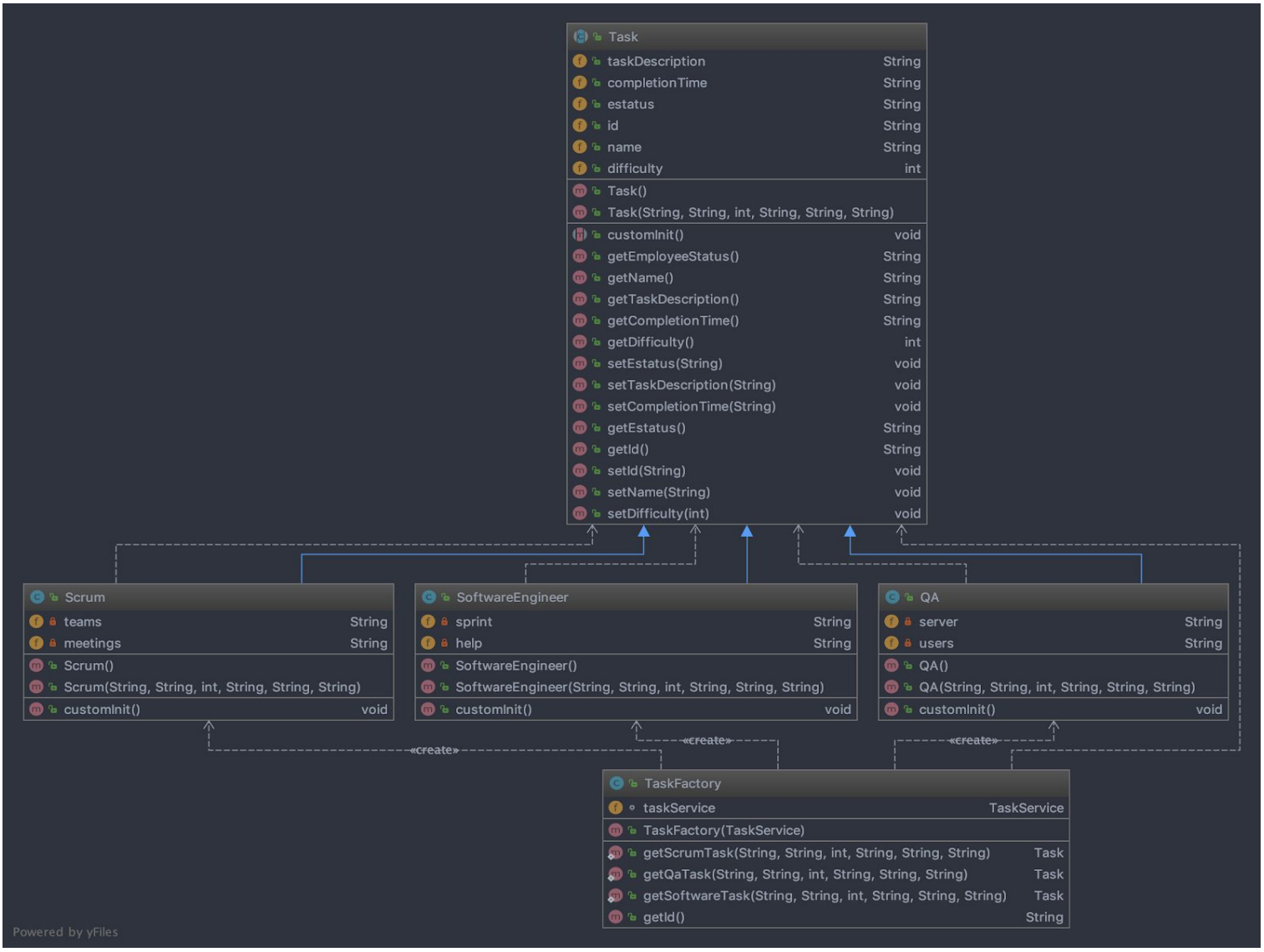
5.

[Full class Diagram](#) (link)

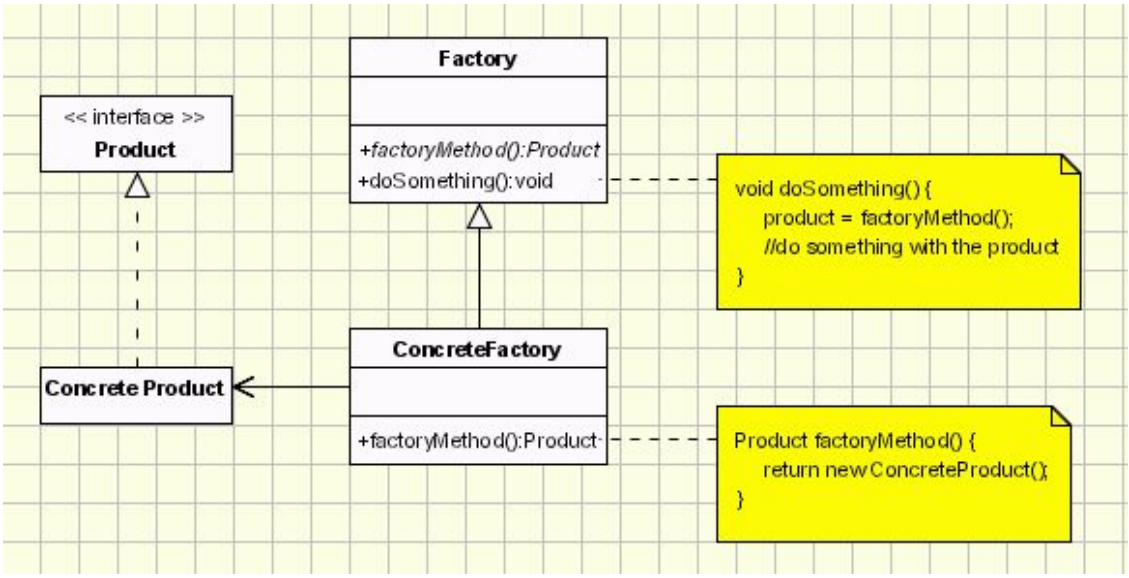


- There are many things that have changed. One of the main things that changed is the addition of more classes and dependencies within those classes. This was due to not understanding what would go into the task creation and the objects associated with it. Also, there are more methods and attributes that allow for better functionality when trying to accomplish the task generation. This is because the classes for the different employees(for the tasks) and service classes were not incorporated. In the initial mock-up the class diagram did not have a design pattern implemented, so that is now in the diagram. Which alone added more classes. I would say overall I underestimated the scope and amount of classes for this project.

6. **My class diagram for the Factory Pattern**



Generic class diagram for Factory:



- I applied this pattern because it allows a clean way to create tasks for the different users. This allows these task objects to be created but allows TaskFactory to decide which subclass to instantiate. This pattern is also good, as it allows all of this creation to be done based of the user input, and creates the task objects with the respective input and goes to the correct subclasses accordingly. I also used this because, Task Class cannot anticipate the type of object it needs to create beforehand, with this pattern it eliminates that ambiguity.

7.

- I have learned many things in this project. When pertaining to analysis and design I learned smarter ways to set-up and execute code. As I was stepping through the code I understood using the design pattern where and why to place classes in certain places, which is better than just writing code as you go and hoping for the best. Especially when implementing the design of a system. When implementing the design of the system you are forced to think thoroughly through how things will connect as we saw in the class diagrams for these projects. So using what we learned I could step through the class diagram and then from there go carefully through the code creation.