CS319 Spring 2022
Design Patterns Assignment

Esad İsmail Tök

21801679

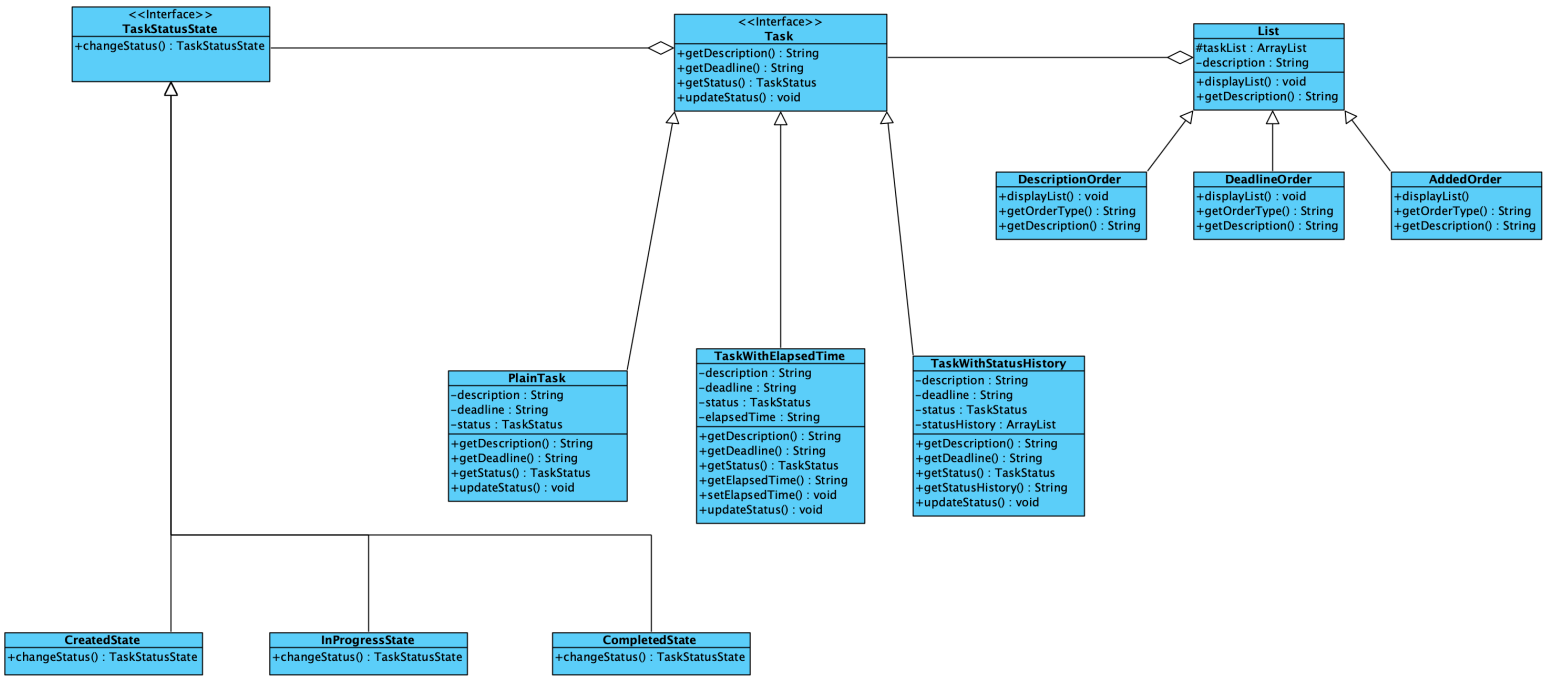

Figure 1: Class Diagram Representing the Used Design Patterns

- In the Task Management Application, Decorator Design Pattern and State Design Pattern are used in the implementation and Figure 1 represents the class diagram of the Task Management Application.

- Detailed explanations about the used design patterns, their reasons to be used, and the implementation of the design patterns are explained in the next page in detail.

Decorator Design Pattern:

- In the assignment description it is stated that there is a Task which has a common behaviors such as returning description of the task, deadline of the task, and the status of the task. However there are also additional options that can be selected such as *TrackElapsedTime* and *TrackStatusHistory.* Choosing these options, our Task will need to perform additional functionalities namely behaviors. Moreover, those additional behaviors will be added in the run time. Also there will be lists in the application that will store a list of tasks and other lists which can be displayed in 3 different orders such as according to description order, deadline order, and added order (ascending order in each of the displays). Therefore it can be understood that it is needed the Decorator Design Pattern in this part of the application.
- In terms of the implementation of the Decorator Design Pattern, we first need a Task interface that will determine the base functionalities of a Task object. Then we implement this interface in the relevant classes and add the additional behaviour in those classes when needed. For example PlainTask class implements the functions of the Task interface and has the necessary properties of its own but does not have any additional behaviour on the other hand TaskWithElapsedTime and TaskWithStatusHistory also have additional behaviours. The List abstract class will have the list of tasks and possibly other lists as a property and common functionality and the subclasses of this abstract class will implement the functions according to the ordering of the tasks by description order, deadline order, and added order in each class.

State Design Pattern:

- A task in the application can have three different statuses which are created, in progress, and completed. A task can switch between different statuses in accordance with specific rules. For example every task has a status of created initially, from the created status a task can move to in progress or completed statuses, from in progress status a task can move to completed status, and from the completed status a task cannot move to another status. All those states and transitions indicates that we need to use the State Design Pattern in this part of the application.
- In terms of the implementation of the State Design Pattern, we have an interface called TaskStatusState that defines the general behaviour of a state which is changing the state from one to another. There are 3 discrete states that implements our interface which are CreatedState, InProgressState, and CompletedState. Finally those discrete states overrides the change state functions according to the transaction rules that are defined in the assignment description.