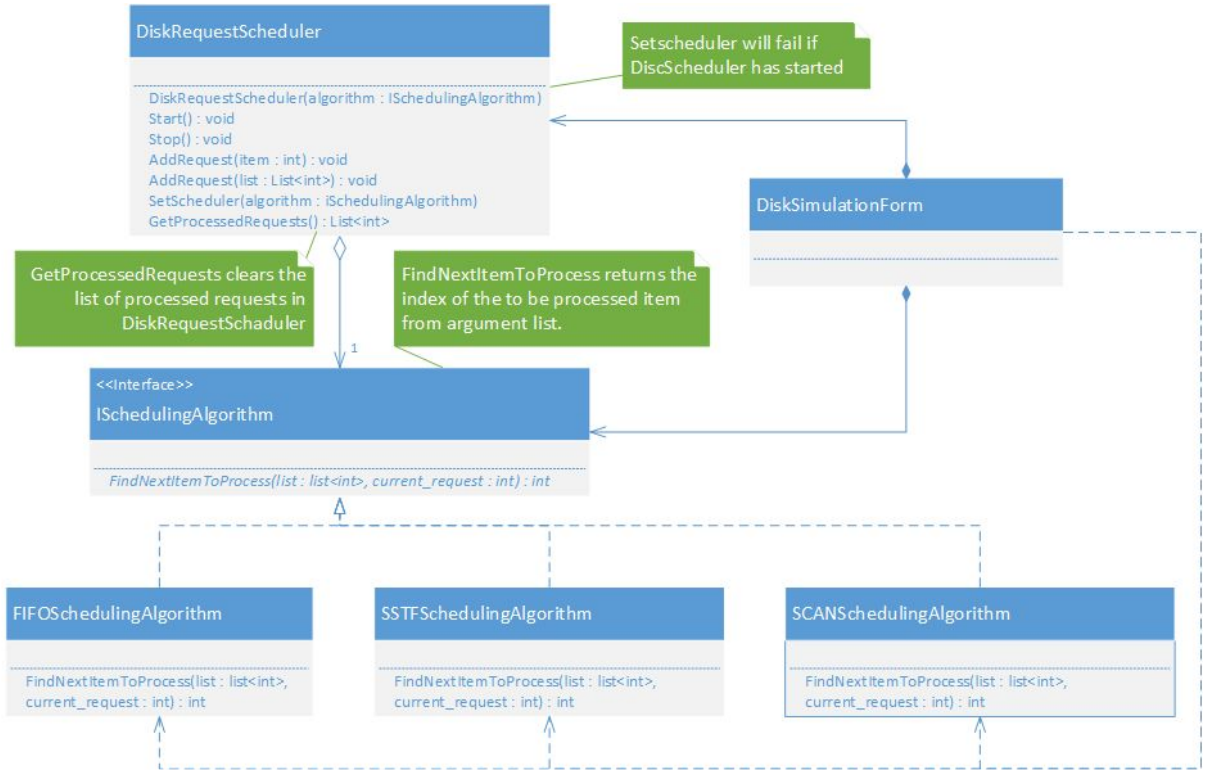
Design patterns assignment 1

Strategy Pattern

Marco Schriek & Jip Muskens T62

Make an object oriented class design for the Disk Scheduling of an (simplified) Operating System. An Operating System can have several different algorithms for handling the requests for a hard disk (like First Come First Serve, Shortest Seek Time First, etc.)

The pattern



The GUI is in charge of constructing all objects. The *DiskRequestScheduler* is highly dependent on a scheduling algorithm to find the next request to process.

By using an intermediate interface *ISchedulingAlgorithm*, it does not matter to the *DiskRequestScheduler* what the exact implementation of the method *FindNextItemToProcess* is. This allows the program to dynamically define different prioritisation behaviours and change it during runtime.

Advantages

- The context class does not have to know about the concrete strategy classes.
- Allow varying algorithms (behaviour) independent of the context.
- Lesser line breaking codes (while, for, if, switch).
- Adding new algorithm classes is easier.
- Avoid multi conditional statements for calling various behaviours. (Some statements can be moved to the strategy class)
- The context class' behaviour can have dynamic behaviour, changeable during runtime.

Disadvantages

- Only the methods that are defined in the interface can be called. Method extension is difficult. If the context must be aware of different strategies, the classes in question must be exposed to the context. The strategy pattern hides these classes in favor of flexibility.
- The varying strategies may not always use all the information that is defined using the strategy interface (using parameters or objects).
- Increased number of objects; one strategy will require at least one object to be defined in order to be used.