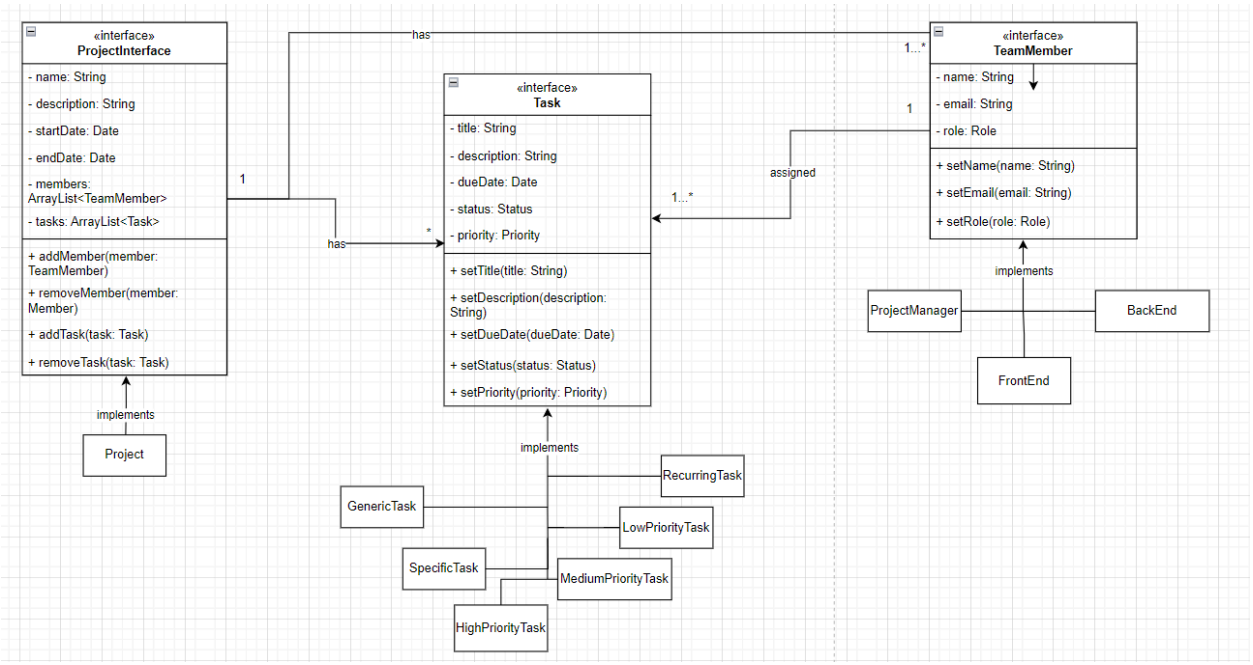
Emily Liew, Ria Nayar, Vivian Ngai, Anushya Shankar, Carly Yin, Ruby Yu
CS 2340 Team 3

SOLID & GRASP Assignment

Interfaces: Project, Tasks, Members
Classes: RecurringTask, LowPriority, HighPriority, MediumPriority

Incorporation of SOLID & GRASP Principles

- Single Responsibility Principle:
  We made sure that each of our class has their own responsibility so they would only change depending on the needed change for that class. For instance, within the RecurringTask class, the code would only be changed/altered depending on if we wanted to change some feature regarding the task that is recurring only.
- Interface Segregation Principle:
  We made sure that each interface created will be used by our users and each interface is focused and not generic. For example, the "tasks" interface is used by users to delegate different assignments for project management.
- Open/Closed Principle:
  We have accomplished this in our code through basic Project, Task, and Member interfaces that define methods/functions that all projects, tasks, and members should have. This ensures that more specific classes that implement these interfaces are able to add to the features provided by the interface but are also required to maintain the basic functionality without modifying it.
- Low Coupling:
  We have 3 interfaces that have minimum coupling by creating classes within the interfaces, like the priority classes within tasks, to ensure unnecessary couples between modules is as low as possible.
- High Cohesion:
  Within each interface, we ensured high cohesion by making sure that our classes and its features' code are highly related to one another. The functionalities are interrelated, an example is that for our RecurringTask and GenericTask all implements Task and their functions within the classes are correlated with one another and are similar.
- Liskov Substitution Principle:
  Objects of the Task superclass can be replaced with objects of its subclasses such as lowPriority, highPriority, and mediumPriority without breaking the application.


«interface»
ProjectInterface
- name: String
- description: String
- startDate: Date
- endDate: Date
- members: ArrayList<TeamMember>
- tasks: ArrayList<Task>
+ addMember(member: TeamMember)
+ removeMember(member: Member)
+ addTask(task: Task)
+ removeTask(task: Task)
has
1
1...*
«interface»
TeamMember
- name: String
- email: String
- role: Role
+ setName(name: String)
+ setEmail(email: String)
+ setRole(role: Role)
1
assigned
1...*
has
*
«interface»
Task
- title: String
- description: String
- dueDate: Date
- status: Status
- priority: Priority
+ setTitle(title: String)
+ setDescription(description: String)
+ setDueDate(dueDate: Date)
+ setStatus(status: Status)
+ setPriority(priority: Priority)
implements
Project
implements
ProjectManager
BackEnd
FrontEnd
implements
RecurringTask
GenericTask
LowPriorityTask
SpecificTask
MediumPriorityTask
HighPriorityTask

Interfaces:

- Project
    - Arraylist of members
    - Arraylist of tasks
- Task
    - Priority will already be set
- Member
    - Role within this for the classes (this should be a variable within that and its set for each type)
    - List of tasks too

Classes:

- GenericTask implements Task
- SpecificTask
- lowPriority implements Task
- highPriority implements Task
- mediumPriority implements Task
- RecurringTask implements Task
- ProjectManager implements Member
- FrontEnd implements Member
- BackEnd implements Member

Enum:

- Status
- Priority