Group Activity - SOLID & GRASP

Team 13

Participating Team Members:

- Justin Cruz
- Varun Giridhar
- Rohit Hair
- Abhinaya Karthana

Applying SOLID and GRASP Principles

1. Single Responsibility Principle (SOLID)
   The Single Responsibility Principle states that a class should have only one reason to change, meaning it should have only one job or responsibility.
   In our design:
   - The **Project** class handles the creation, removal, and management of both tasks and team members. It is solely responsible for managing a project's data and associated members.
   - The **Task** class and its subclasses (e.g., HighPriorityTask, RecurringTask) are responsible for defining task-specific properties and behaviors (such as due dates, priorities, and status management).
   - The **TeamMember** and its subclasses (Developer, ProjectManager) focus on the execution of tasks and interaction with the project, ensuring clear separation of concerns.

2. Open/Closed Principle (SOLID)
   The Open/Closed Principle states that software entities (classes, modules, functions) should be open for extension but closed for modification. This principle promotes extensibility without altering existing code, reducing the risk of introducing bugs.
   In our design

   - The **Task** class is abstract, allowing for the creation of specialized task types like HighPriorityTask and RecurringTask without altering the original Task class. This allows us to extend the system with new types of tasks without changing the existing code.
   - Similarly, the **TeamMember** class is abstract, and different types of team members (e.g., Developer, ProjectManager) can be added by extending this class without modifying it.

3. Information Expert (GRASP)
   The Information Expert principle in GRASP states that responsibility for handling a piece of information should be assigned to the class that has the information necessary to fulfill that responsibility.
   In our design:

   - The **Project** class manages information related to tasks and team members. Since it contains lists of tasks and team members, it is responsible for adding/removing them and managing their state.

- The **Task** class is responsible for managing its own status, due date, and priority. It provides methods for setting and retrieving this information (setStatus(), getDue_date()), making it the expert on task-related details.
- The **TeamMember** class is responsible for executing tasks, and it contains the necessary information (e.g., team member's name and email) to join or leave a project.

4. Creator (GRASP)
   The Creator principle states that a class should be responsible for creating instances of another class if it contains, aggregates, or closely uses those objects.
   In our design:

   - The **Project** class is responsible for creating and managing Task and TeamMember objects because it aggregates both tasks and team members. It creates tasks via the addTask() method and manages team members via the addMember() method.
   - This principle is also reflectd in how the **Project** class manages the lifecycle of these objects, including adding and removing them from the project.