Course Title: Advanced Programming Laboratory
Course Code: 0714 02 CSE 2100

Field of Application: Club Management

Submitted by:
Kazi Rifat Morshed (Student ID: 230220)
Md. Rimon Islam (Stundent ID: 230236)

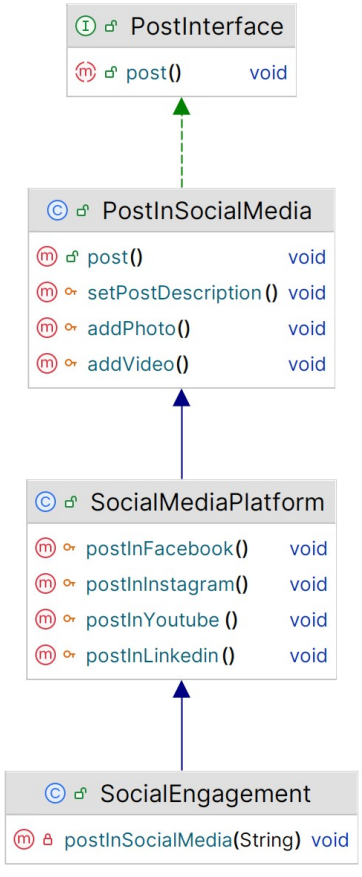
Implementation of SRP (Single Responsibility Principle):

All classes for HostContest, Meeting, SocialEngagement functionality are kept seperate so that any class bear only one responsibility. Which means, any class can provide only one type of service.

Implementation of OCP (Open/Closed Principle):

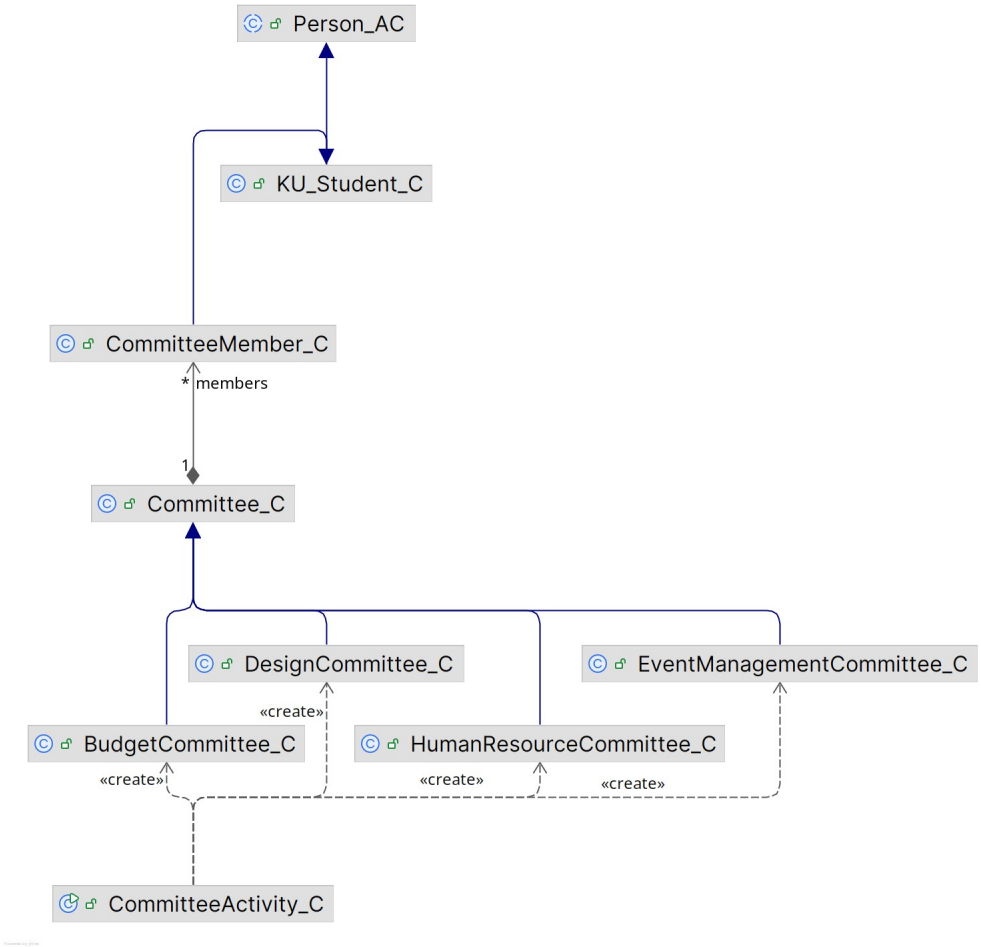
According to OCP principle, classes should be open for extenson, but closed for modification.

For Social Engagement, interface 'Post' holds only 'post()' method. 'PostInSocialMedia' class implements this interface and defines 'post()' method. 'SocialMediaPlatform' class extends 'Social Platform' class. 'SocialEngagement' class uses 'postInSocialMediia()' method of 'SocialMediaPlatform' class. We have extended functionality of parent class without modifying it.



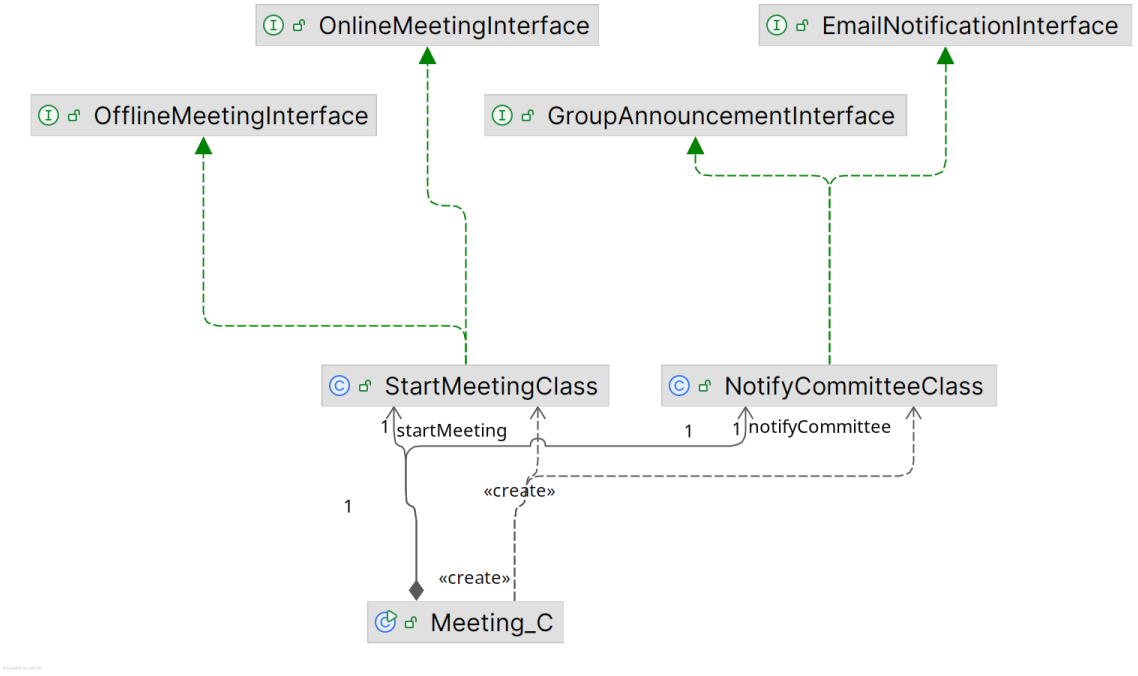
Implementation of LSP (Liskov Substitution Principle):

According to LSP, subclass should remain compatible with the behavior of the superclass. Here, 'Committee_C' is superclass and 'EventManagementCommittee_C' class, 'DesignCommittee_C class', 'BudgetCommittee_C' class, 'HumanResourcesCommittee_C' class are subclasses of 'Committee_C' class. These classes does not replace functionality of parent class.



Implementation of ISP (Interface Segregation Principle):

ISP states that Clients shouldn't be forced to depend on methods they do not use. To achieve this goal, different interfaces can be created that holds not more than one method signature. Since multiple interface implementation is supported, our classes can extend only necessary interfaces and override necessary methods.



Implementation of DIP (Dependency Inversion Principle):

According to Dependency Inversion Principle, High-level classes shouldn't depend on low-level classes. Both should depend on abstractions.
In this implementation, 'ResultClass', 'DecisionMakerClass', 'JudgeClass', 'SubmissionClass', 'PrizeClass', 'AnnouncementClass' do not depent on low level classes, rather depends on interfaces respectively 'ResultInterface', 'DecisionMakerInterface', 'JudgeInterface', 'SubmissionInterface', 'PrizeInterface', 'AnnouncementInterface'. In this implementation, instead of creating instances of concrete class, instances of interfaces were created.