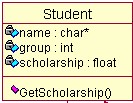
# Практична робота № 5 ПОБУДОВА ДІАГРАМ КЛАСІВ

**Теоретичні відомості** Діаграми класів (***Class diagrams***) – головний тип діаграм UML, які відображають логічну структуру програмної системи та суттєво впливають на процес генерації програмного коду. Основними елементами діаграми класів у Rational Rose є безпосередньо класи (***classes***) та відношення між ними *(****relations****)****.***

***Клас*** – сукупність логічних об’єктів, які мають схожі характеристики і відрізняються однаковою поведінкою. В ООП характеристики об’єктів певного класу представлені сукупністю атрибутів, а поведінка – сукупністю операцій.

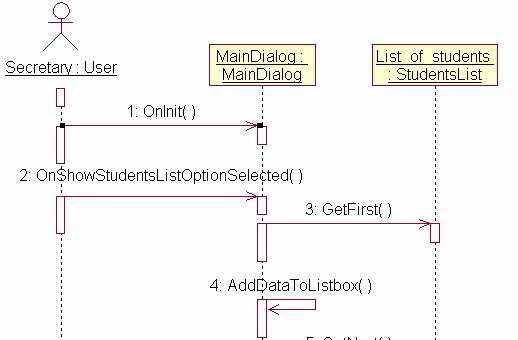


**Рис. 7.** Графічне представлення класу на діаграмі класів у Rational Rose У Rational Rose кожен клас графічно представлений прямокутником, що має три секції: ім’я класу, перелік атрибутів та перелік операцій (рис. 7). Для кожного атрибуту задається тип даних, для кожної операції тип даних для значення, що повертається, та перелік параметрів. Атрибути та операції можуть бути визначені для кожного об’єкта класу, чи для класу в цілому (***static*** attributes and operations). Також для всіх атрибутів та операцій можна визначити тип видимості (public, protected, private). Розрізняють декілька типів класів:

1. Конкретний клас – для даного класу можна створити об’єкт.
2. Абстрактний клас – клас, для якого не можна створити об’єкт. Даний клас виступає чистою абстракцією, від нього можна наслідувати конкретні класи.
3. Параметризований клас – клас, для визначення якого використовується список формальних параметрів, які впливають на атрибути та операції (аналог шаблона в С++). Найчастіше такі класи використовуються для створення абстрактних типів даних (списки, вектори, стеки, черги і т. п.).
4. Інтерфейс – клас, що містить тільки визначення набору операцій (без реалізації). У мові С++ аналогом даного класу є клас, всі операції якого є чистими віртуальними функціями. В мові Java класи можуть реалізовувати декілька інтерфейсів (інтерфейси використовуються для реалізації концепції множинного наслідування).

***Створення нових класів та операцій класів:***

У Rational Rose для кожного об’єкта на діаграмах взаємодії існує можливість призначити певний клас (чи створити новий). При цьому в прямокутнику об’єкта дані відображаються в наступному форматі <ім’я об’єкта>:<ім’я класу>. Також кожне повідомлення на діаграмі взаємодії представляє певну операцію класу-приймача. Для кожного повідомлення можна обрати існуючий метод класу-приймача чи створити новий. На рис. 8 наведено фрагмент модифікованої діаграми послідовності для варіанта використання «Переглянути список студентів».



**Рис. 8.** Фрагмент діаграми послідовності для варіанта використання

«Переглянути список студентів»

Після створення набору класів з операціями можна відобразити їх на діаграмі класів. При цьому кожен клас буде мати ім’я та перелік операцій, визначених користувачем. На рис. 9 показано відображення класу MainDialog з операціями, створеними для повідомлень 2 та 4 попередньої діаграми (повідомлення 1 буде показано пізніше на загальному вигляді діаграми класів, оскільки воно визначено для класу Dialog, який є базовим для даного класу).



**Рис. 9.** Відображення класу MainDialog на діаграмі класів

Також нові класи та операції класів можна створювати безпосередньо з діаграми класів.

***Додавання атрибутів класів:***

Після створення переліку класів із набором операцій необхідно для кожного класу створити набір атрибутів – даних, якими оперує програма у процесі виконання операцій класу. Наприклад, виходячи з діаграми послідовності, зображеної на рис. 10 можна визначити для класу MainDialog атрибути для кнопки (Button), яку натискає користувач для перегляду списку студентів та списку, до якого безпосередньо додається інформація про кожного студента.



**Рис. 10.** Клас MainDialog з повним набором атрибутів та операцій

***Повний формат відображення атрибутів на діаграмі класів:***

<attribute\_name>:<attribute\_type>

***Повний формат відображення операцій на діаграмі класів:***

<operation\_name>(<list\_of\_arguments>):<type\_of\_return\_value> ***Відношення між класами:***

Rational Rose для нотації Unified можна задати 5 основних типів *відношень* між класами:

1. Відношення асоціації (***association***) – визначає абстрактний зв’язок між класами, може представляти аналог відношення «m до n».

***Приклад:*** відношення між класами Student та Course (кожен студент відвідує декілька курсів (дисциплін), для кожної дисципліни визначений набір студентів, які її відвідують). ***Представлення в програмному коді:***

class Student

{ .............

Course\* theCourses;

....................};

class Course

{.................

Student\* theStudents;

.................};

При цьому Course\* в класі Student може вказувати на масив дисциплін, а Student\* у класі Course – на перелік студентів.

1. Відношення агрегації (***aggregation***) – відношення типу «частина – ціле», при якому час життя класа частини не співпадає з часом життя класа цілого.

***Приклад:*** відношення між класами Student та Ticket (у час переоформлення студентського квитка студент не має його). ***Представлення у програмному коді:*** class Student

{ .............

Ticket\* ticket;

............};

Атрибут ticket при цьому може створюватися та видалятися (за допомогою операторів new та delete) в коді будь-яких методів класу Student.

1. Відношення композиції (***composition***) – відношення типу «частина – ціле», при якому час життя класа частини співпадає з часом життя класа цілого.

***Приклад:*** відношення між класами Student та Date\_of\_Birth (кожен студент у будь-який момент часу характеризується датою народження).

***Представлення у програмному коді:*** class Student

{ .............

Date\_of\_Birth theDate\_of\_Birth;

...............}

Атрибут theDate\_of\_Birth існує протягом усього часу життя об’єкта класу Student.

1. Відношення наслідування (***generalization***) – відношення типу «загальне – часткове».

***Приклад:*** відношення між класами Student та Astudent (студент відмінник є частковим випадком студента). ***Представлення в програмному коді:*** class Student ; class Astudent : public Student{......

float GetScholarBonus();

.........};

Клас Astudent має додатковий метод GetScholarBonus(), який розраховує надбавку до стипендії для студента відмінника.

1. Відношення інстанціювання (instantiation) – визначає інстанціювання нового класу з параметризованого класу шляхом підставлення фактичних параметрів у формальні параметри шаблона.

***Приклад:*** відношення між класами List та StudentsList (з абстрактного списку інстанціюється список студентів). ***Представлення в програмному коді:***

template <class T> class List {………….} // List – абстрактний список typedef List <Student> StudentsList; // List<Student> – список студентів

Для класу, що інстанціюється, генерація програмного коду відсутня, оскільки він генерується компілятором автоматично при зверненні до параметризованого класу (підставлення фактичних параметрів у формальні параметри шаблона).

На рис. 11 наведено приклад діаграми класів для системи обліку успішності студентів у деканаті, яка містить перелік класів, пов’язаних різними типами відношень.



**Рис. 11.** Діаграма класів для системи обліку успішності навчання студентів у деканаті

## Завдання

1. Для всіх об’єктів на діаграмах взаємодії призначити (створити) певний клас; для кожного повідомлення призначити (створити) відповідний метод (операцію) для класу об’єкта-приймача.
2. Розташувати створені класи з переліком операцій на діаграмі класів.
3. Для кожної операції визначити атрибути, які вона використовує та при необхідності додати їх до списку атрибутів класу.
4. Для кожного атрибуту задати логічний тип даних, для кожної операції логічний тип даних для return value та для переліку аргументів, якщо вони присутні.
5. Пов’язати класи на діаграмі класів, використовуючи різні типи відношень (асоціацію, агрегацію, композицію, наслідування, інстанціювання).

## Вимоги

1. Діаграма класів повинна містити не менше 10 класів.
2. Для кожного класу визначити не менше 5 атрибутів та 5 операцій.
3. По можливості використати всі типи відношень між класами.

**Контрольні запитання** 1. Для чого призначена діаграма класів (class diagram)? Назвіть її основні елементи.

1. Дайте визначення класу. Які типи класів ви знаєте?
2. Чим відрізняється абстрактний клас від конкретного?
3. Чим відрізняється абстрактний клас від інтерфейсу?
4. Що таке параметризований клас? Дайте визначення процесу інстанціювання. Наведіть приклад.
5. Назвіть основні характеристики класу.
6. Що таке атрибути класу? Які типи атрибутів ви знаєте?
7. Що таке операції класу? Які типи операцій ви знаєте?
8. Дайте визначення статичних атрибутів та операцій класу.
9. Дайте визначення наслідування. Назвіть основні типи наслідування.
10. Яким чином реалізується інкапсуляція для атрибутів та операцій класу? Назвіть типи модифікаторів доступу до елементів класу.
11. Яким чином реалізується поліморфізм для класів, що знаходяться в єдиній ієрархії? Наведіть приклад.
12. Що таке віртуальні функції, для чого вони призначені?
13. Яким чином можна створити класи та їх операції з діаграм взаємодії?
14. В якому форматі відображають атрибути та операції на діаграмі класів?
15. Перелічіть можливі відношення між класами на діаграмі класів.
16. Які характеристики можна визначити для відношення асоціації?
17. Як відношення асоціації можна перетворити в агрегацію/композицію?
18. У чому відмінність між відношеннями агрегації та композиції?
19. Який клас є базовим, а який похідним при визначенні відношення наслідування?
20. Які типи класів поєднує відношення інстанціювання?
21. Яким чином кожний тип відношень між класами впливає на генерацію програмного коду?