Лекція 10. Відносини між класами

Жоден з об'єктів навколишнього нас світу не існує сам по собі. Птахи літають тому, що є повітря, на яке спираються їхні крила. Кожен з нас пов'язаний з масою інших людей різноманітними родинними, професійними та іншими зв'язками, які передбачають різні типи відносин.

Так само і класи пов'язані між собою. І щоб в повній мірі оволодіти ООП, необхідно зрозуміти суть цих відносин і навчитися їх ідентифікувати.

Спадкування (узагальнення) – це відношення між більш загальною сутністю, яка називається суперкласом, та її конкретним втіленням, що називається підкласом.

Спадкування – один із трьох фундаментальних принципів об'єктно-орієнтованого програмування, оскільки саме завдяки йому можливе створення ієрархічної класифікації.

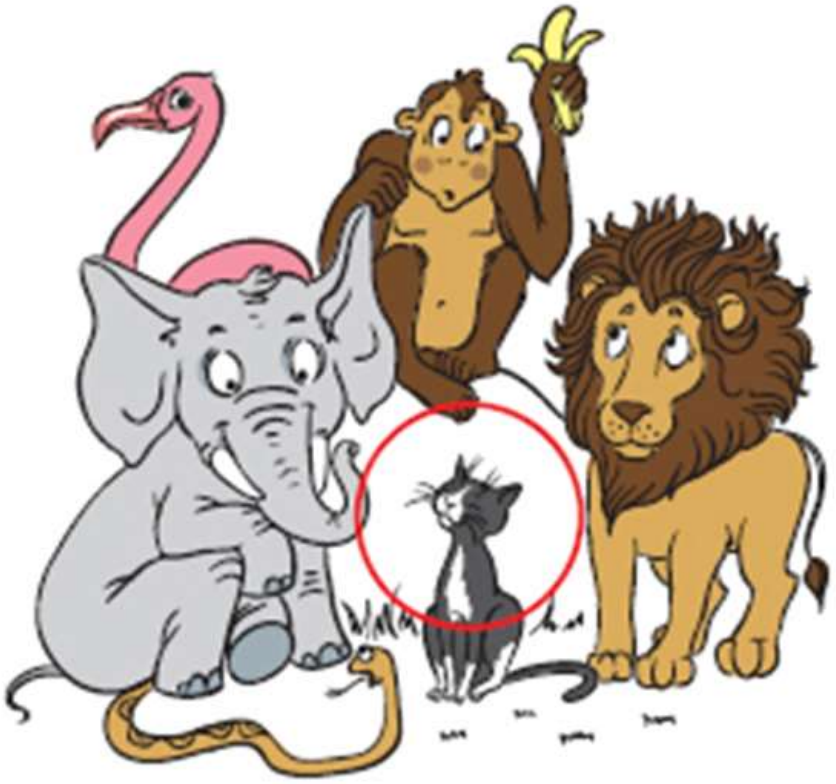


Рис. 1. Один об'єкт представляє конкретний екземпляр більш загального класу.

Узагальнення (або спадкування) на діаграмах позначається незафарбованою трикутною стрілкою, спрямованої на суперклас (базовий клас) (Рис. 2)

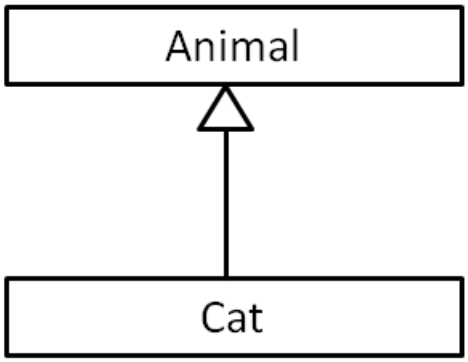


Рис. 2. Узагальнення або спадкування.

Для того щоб навчитися ефективно моделювати наслідування, звернемося до класиків, а саме до Г. Бучу. Він радить проводити цю процедуру в такій послідовності:

1. Знайдіть атрибути, операції і обов'язки, загальні для двох або більше класів з даної сукупності. Це дозволить уникнути непотрібного дублювання структури і функціональності об'єктів.
2. Винесіть ці елементи в певний загальний суперклас, а якщо такого не існує, то створіть новий клас.
3. Відзначте в моделі, що підкласи успадковуються від суперкласу, встановивши між ними відношення узагальнення.

Залежність

Розглянемо ще один вид відносин між класами – залежність.

Залежність виникає тоді, коли реалізація класу одного об'єкта залежить від специфікації операцій класу іншого об'єкта. І якщо зміниться специфікація операцій цього класу, нам неминуче доведеться вносити зміни і в залежний клас.

Наприклад, операція «Відтворення», що реалізується класом-медіаплеєром, залежить від операції «Декомпресія», що реалізовується

класом кодек. Якщо специфікація операції «Декомпресія» зміниться, доведеться змінювати код медіаплеєра.

Ось так залежність між класами зображується в UML (Рис. 3)

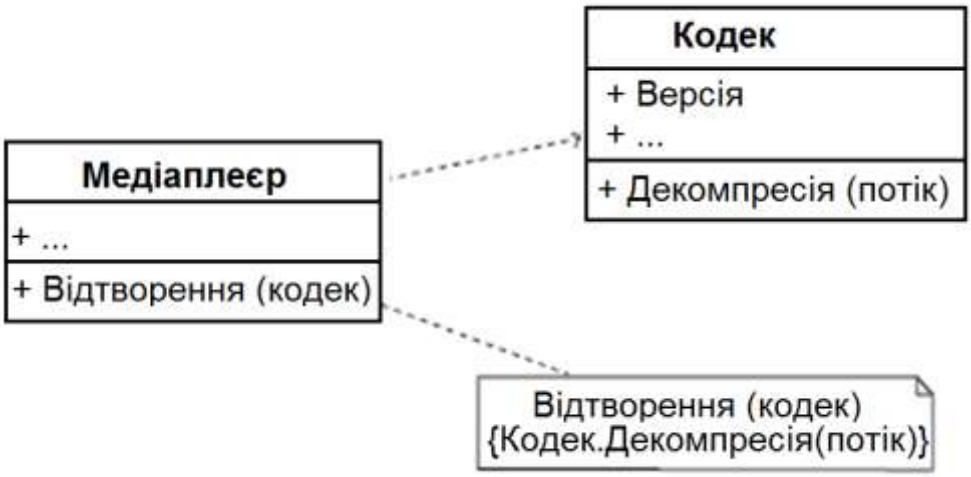


Рис. 3. Залежність між класами.

Варто відзначити, що залежності на діаграмах зображують далеко не завжди, а тільки в тих випадках, коли їх відображення є важливим для розуміння моделі. Часто залежності лише маються на увазі, бо логічно випливають з природи класів.

Асоціація. Агрегація і композиція

Інший вид відносин між об'єктами – це асоціація. Це просто зв'язок між об'єктами, по якому можна між ними переміщатися (Рис. 4).

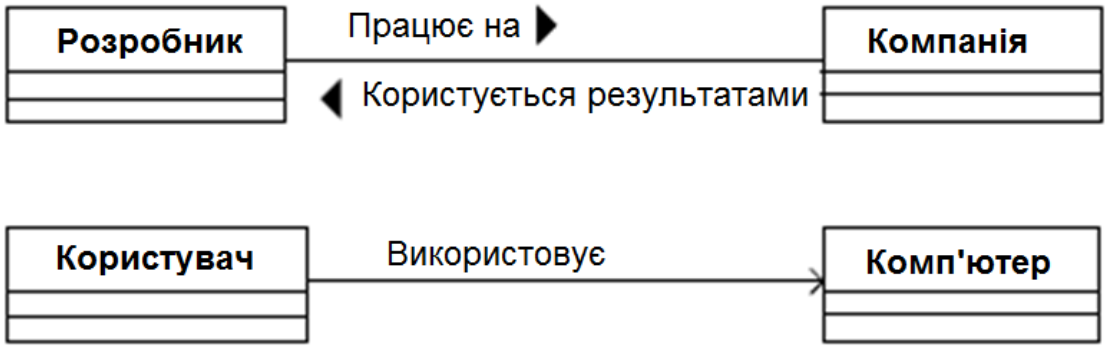


Рис. 4. Зв'язок між об'єктами типу асоціація.

Асоціація може мати ім'я, яке показує природу відносин між об'єктами, при цьому в імені може вказуватися напрям читання зв'язку за допомогою трикутного маркера. Односпрямована асоціація може зображуватися стрілкою.

Крім напряму асоціації, ми можемо вказати на діаграмі ролі, які кожен клас відіграє в даному відношенні, і кратність, тобто кількість об'єктів, пов'язаних відношенням (Рис. 5).

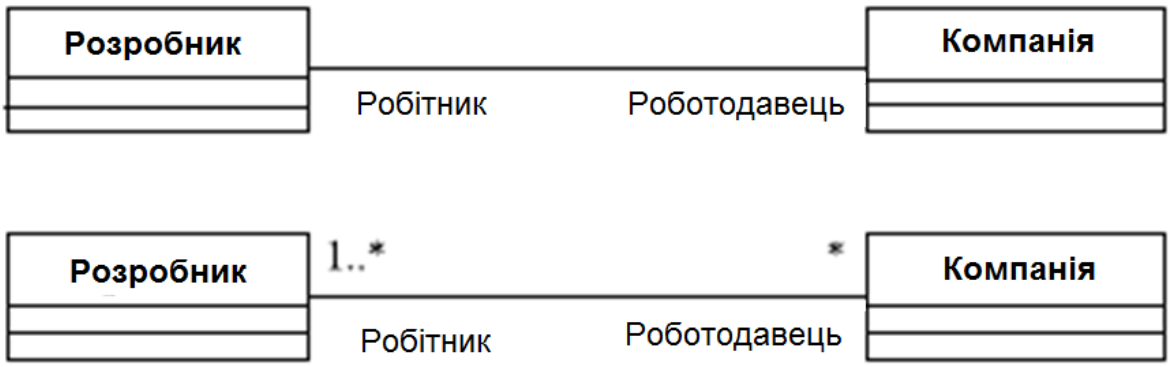


Рис. 5. Асоціація. Ролі і кратність.

Насправді, в реальності зв'язки бувають «просто зв'язками» вкрай рідко. Зазвичай при найближчому розгляді під асоціацією розуміється більш складне відношення між класами, наприклад, зв'язок типу «частина-ціле».

Такий вид асоціації називається асоціацією з агрегуванням. У цьому випадку один клас має більш високий статус (ціле) і складається з нижчих за статусом класів (частин).

При цьому виділяють просте і композитне агрегування і говорять про агрегацію і композицію.

Проста агрегація передбачає, що частини, відділені від цілого, можуть продовжувати своє існування незалежно від нього.

Під композитним ж агрегуванням розуміється ситуація, коли ціле володіє своїми частинами і їх час життя відповідає часу життя цілого, тобто незалежно від цілої частини існувати не можуть.

На рисунку 6 показано відношення типу агрегація і композиція з використанням зафарбованого та не зафарбованого ромбика.

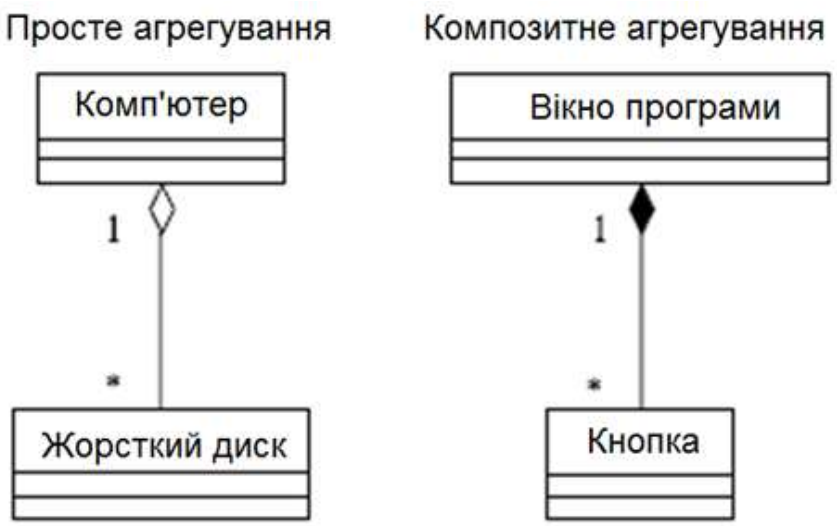


Рис. 6. Просте і композитне агрегування (агрегація і композиція).

Наступний програма показує приклад агрегації і композиції:

```
// Приклад агрегації і композиції
class Brain
{
public:
 int Volume;
 void Think()
 {
      cout << "я мислю, отже, я існую" << endl;
 }
};

class Human
{
public:
 int year_of_birth;
 Brain brain;   // містить мозок
 void Think()
 {
 brain.Think(); // людина сама по собі думати не може, думає мозок
 }
};
class Student: public Human
{
public:
 string pib;
 string arrdess;
 void Print()
 {
      cout<<"Студент "<<pib<<" каже: ";
      brain.Think();
 }
};
```

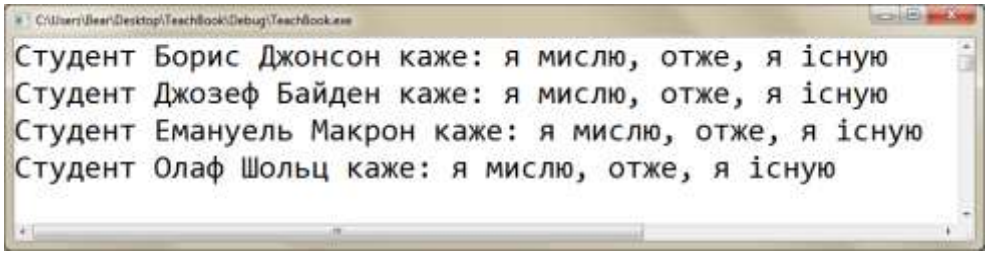
```
class Group
{
 Student *p;
 int kilkist;
public:
 Group(int kilkist)
 {
      this->kilkist = kilkist;
      p = new Student[kilkist];
 }

 ~Group()
 {
      delete []p;
 }
 Student & operator [] (int i)
 {
      return p[i];
 }

};

int main()
{
 setlocale(0, "");
 Group group(4);
 group[0].pib="Борис Джонсон";
 group[1].pib="Джозеф Байден";
 group[2].pib="Емануель Макрон";
 group[3].pib="Олаф Шольц";
 for(int i=0; i<4; i++)
      group[i].Print();
}
```

Результат роботи програми:



Кожна людина має мозок, але мозок не може функціонувати без людини. Тому клас Brain (мозок) є композитним для класу Human (людина).

Людина сама по собі думати не може, думає мозок. Коли викликається метод Think (думати) для об’єкта класу Human або успадкованого від нього класу Student, цей метод викликає метод Think (думати) у мозку (в класі

Brain). Така ситуація називається «делегування» і це, до речі, один із паттернів проектування.

Кожен студент є людиною, тобто нащадком класу Human. Це відношення типа спадкування. Крім того, студент є членом групи, де може бути кілька студентів, але студент може існувати і без групи. Тому клас Student є агрегованим для класу Group (група).

На рис. 7. показані відносини між цими класами.

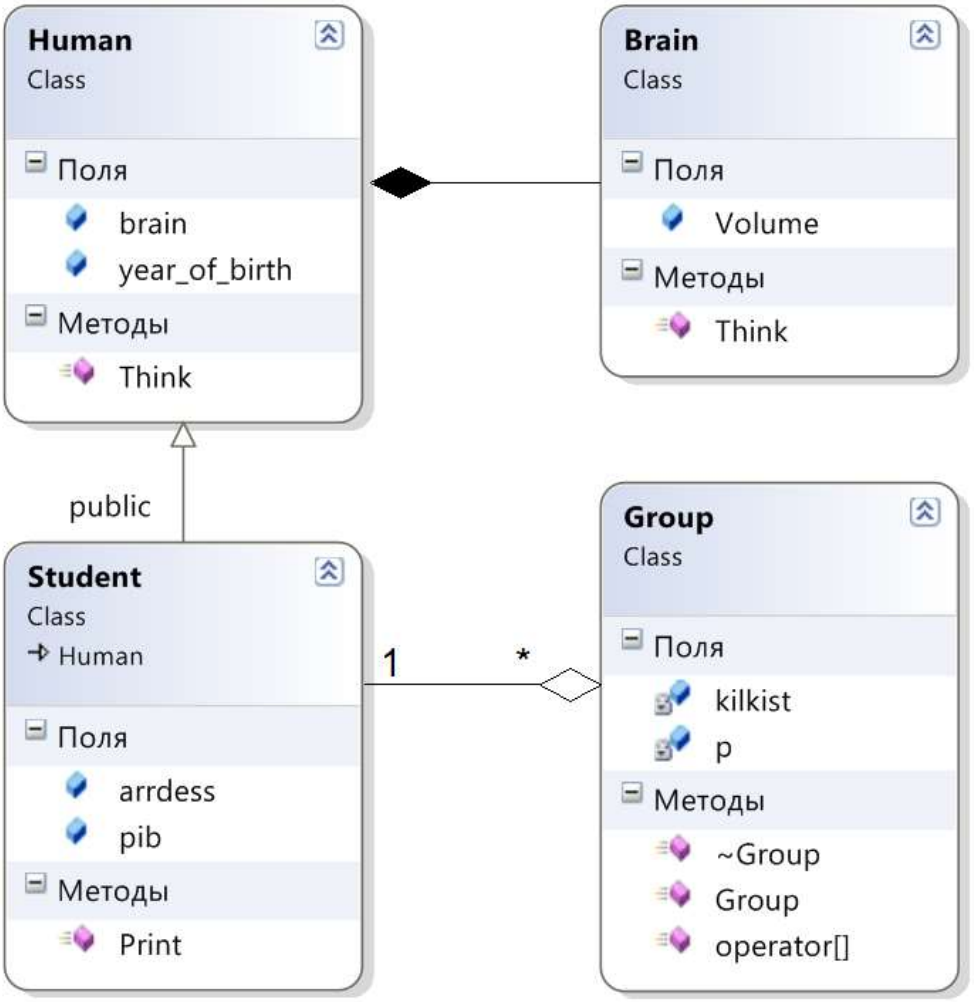


Рис. 7. Відносини між класами типа спадкування, агрегація і композиція.

Резюме

1. Використовуючи спадкування, можна створити клас, який визначає найбільш загальні характеристики всіх пов'язаних з ним елементів. Цей клас потім може бути успадкований іншими, спеціалізованими класами з додаванням в кожен з них своїх, унікальних особливостей.

2. Якщо один клас успадковує інший, члени базового класу стають членами похідного. Статус доступу членів базового класу в похідному класі визначається модифікатором доступу, що використовуються для спадкування базового класу.

3. Використовуючи модифікатор protected, можна створити члени класу, які закриті в рамках свого класу, але які може успадкувати похідний клас, причому з отриманням доступу до них.

4. Якщо базовий клас успадковується з використанням ключового слова public, його public-члени стають public-членами похідного класу, а його protected-члени – protected-членами похідного класу.

5. Якщо базовий клас успадковується з використанням модифікатора protected, його public і protected-члени стають protected-членами похідного класу.

6. Якщо базовий клас успадковується як private-клас, його public-члени стають private-членами похідного класу.

7. Похідний клас може успадковувати два або більше базових класів (так зване множинне спадкування).

8. Якщо базовий клас оголошується віртуальним, то тільки один його екземпляр буде включений в об'єкт похідного класу. Різниця між звичайним базовим і віртуальним класами стає очевидною тільки тоді, коли цей базовий клас успадковується більше одного разу.

9. Зв’язок між класами називається асоціацією з агрегуванням, коли один клас має більш високий статус (ціле) і складається з нижчих за статусом класів (частин).

10. Проста агрегація передбачає, що частини, відділені від цілого, можуть продовжувати своє існування незалежно від нього.

11. Під композитним агрегуванням розуміється ситуація, коли ціле володіє своїми частинами, які незалежно від цілої частини існувати не можуть.