**Лабораторна робота №16. Розробка програм із застосуванням типу відношення has-a між класами**

**Мета:** Засвоєння навичок розробки програм з використанням типу відношення has-a між класами

**Порядок виконання роботи**

1. Ознайомитися з теоретичними основами розробки віртуальних функцій та застосуванням поліморфізму.
2. Створити класи BusStation (Автостанція) містить масиви екземплярів класів Bus (Автобус), Car (Автомобіль). Кількість елементів у масивах класів Bus та Car не повинна перевищувати максимально-можливу кількість транспортних засобів на автостанції, яка задається константою.

* Клас Car має один член класу – текстове поле, що описує модель автомобіля та методи: конструктори (визначте потрібні), селектор, модифікатор та відображення інформації про автомобіль.
* Клас Bus має два члени класу – текстове поле, що описує модель автобусу та ціле – кількість місць для пасажирів, а також методи: конструктори (визначте потрібні), селектор, модифікатор та відображення інформації про автомобіль.
* Клас BusStation має такі члени класу: масив об'єктів класу Bus і поточна кількість автобусів, масив об'єктів класу Car і поточна кількість автомобілів, а також методи доступу, що обробляють масив автобусів та автомобілів (відповідно, окремо):
* отримати інформацію про транспортний засіб за його номером (індексом)з перевіркою, чи коректний номер;
* додати новий транспортний засіб з перевіркою, що максимально-можливу кількість транспортних засобів на автостанції не перевищена;
* вивести інформацію про розміщені на даний момент транспортні засоби на автостоянці.

В головній програмі оголосити екземпляр класу BusStation, створити декілька екземплярів класів Bus та Car, причому щоб отримати повідомлення про перевищенн максимально-можливої кількості транспортних засобів на автостанції, вивести інформацію про розміщені на даний момент транспортні засоби на автостоянці.

Оформити однією програмою, до якої включити опис класу.

1. Продовжити працювати над програмою лабораторної роботи №11, закінчити попередні завдання з урахуванням наданих зауважень (для тих, хто не здав). Результати надсилати разом зі сформованими файлами.
2. В першому рядку програми та заголовкового файлу повинні бути записаними в коментарі номер групи та прізвище, а також номер ЛР (через кому до попередньої).
3. Результати надсилати на електронну адресу викладача

[**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді заголовкового файлу та cpp-файлу з іменем у форматі

Для заголовкового файлу

**<Прізвище англійською>.h**

Наприклад, Ivanov.h

Для cpp-файлу:

**<Номер групи> <Номер лабораторної><Прізвище англійською>.cpp**

Наприклад, 31-01Ivanov.cpp.

Тему в заголовку листа записати

**ООП<Номер групи>-ЛР <Номер лабораторної>-<Прізвищеанглійською>**

**Строк відсилки ЛР ІПЗ-31 08.12.2020**

**ІПЗ-32 07.12.2020**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**ООП<Номер групи>-Запитання-<Прізвище англійською>**.

**Контрольні запитання**

1. Який вид зв’язку має бути реалізований у завданні?

**Теоретичні відомості.**

##### Типи відношень між класами. Класифікація

Між класами можливі два типи відношень:

* 1. Відношення типу is-a (є, являється) при якому один клас є підвидом іншого класу. При такому співвідношенні один клас розширює (деталізує) можливості іншого класу. Розширення можливостей класу здійснюється завдяки використанню спадковості.
* 2. Відношення, при якому існує взаємозв’язок між двома класами. Тут виділяють два підвиди взаємозв’язку між класами:
  + 2.1. Відношення типу has-a (клас містить інший клас). У цьому випадку в класі оголошується один або декілька екземплярів іншого класу. При даному відношенні можливі два випадки взаємодії. Перший випадок, це коли об’єкт (екземпляр), що оголошений в класі, не є складовою частиною класу (агрегація) і його використання не впливає на функціональну роботу класу. Другий випадок, коли об’єкт, що оголошений в класі, є складовою частиною цього класу (композиція).
  + 2.2. Відношення типу uses (клас “використовує інший клас). У цьому випадку клас містить програмний код іншого вкладеного класу, до якого він має доступ.

При відношенні has-a клас містить один або декілька об’єктів (екземплярів) іншого класу. Існує два види відношення has-a:

* агрегація. Це випадок, коли один або декілька вкладених об’єктів не є частиною класу, тобто клас може містити будь-яку кількість таких об’єктів (навіть 0).
* композиція. У цьому випадку один або декілька вкладених об’єктів є частиною класу, тобто без цих об’єктів неможливе логічне існування самого класу.

##### Приклад 1.Клас Figures містить екземпляри класів Triangle (трикутник) та Circle (коло)

У даному скороченому коді демонструється агрегація на прикладі класів Triangle, Circle, Figures. Клас Figures може містити різну кількість різних фігур (навіть 0). При можливості до класу Figures може бути додано масиви інших фігур, наприклад, Rectangle (прямокутник). У будь-якому випадку клас Figures буде повноцінним функціонально, отже, це є агрегація.

// Клас трикутник

class Triangle

{

// Методи та поля класу Triangle

// ...

};

// Клас, що реалізує коло

class Circle

{

// Методи та поля класу Circle

// ...

};

// Клас, що реалізує різні геометричні фігури.

// Використовується тип відношення - агрегація.

class Figures

{

Triangle tr[10]; // масив трикутників

unsigned int n\_tr; // к-сть трикутників у масиві tr

Circle cr[10]; // масив кіл

unsigned int n\_cr; // к-сть кіл у масиві cr

// Інші поля та методи класу

// ...

};

##### Приклад 2. Клас Bike (Велосипед) містить екземпляри класів Wheel (Колесо) та Saddle (сідло)

Клас Bike (Велосипед) містить екземпляри класів Wheel (Колесо) та Saddle (сідло), які є його складовою частиною. Велосипед не може повноцінно функціонувати без коліс чи сідла, тому це є композиція.

// Клас Колесо

class Wheel

{

// Поля та методи класу

// ...

};

// Клас Велосипед, містить колеса, сідло

// Клас сідло

class Saddle

{

// Поля та методи класу

// ...

};

// Клас Велосипед, містить об'єкти класів,

// які є частиною Велосипеду (сідло, колеса) - це є композиція

class Bike

{

Saddle sd; // одне сідло - частина, яка доповнює велосипед

Wheel whl[2]; // два колеса - частина велосипеду

// Поля та методи класу Bike

// ...

};