МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні вказівки
до виконання лабораторних робіт для студентів
спеціальності 7.080401 „Інформаційні
управляючі системи та технології”

Київ 2009

УДК 681.3.06
ББК 32.973 – 01
О – 29

Укладач Г.В. Красовська, канд. техн. наук, доцент

Рецензент О.В. Федусенко, канд. техн. наук, доцент

Відповідальний за випуск В.Б. Задоров, канд. техн. наук, професор

Затверджено на засіданні кафедри інформаційних технологій, протокол № 20 від 27 квітня 2009 року.

Видається в авторській редакції.

Об'єктно-орієнтоване програмування: методичні вказівки
О – 29 до виконання лабораторних робіт / уклад. Г.В. Красовська. – К.: КНУБА, 2009. – 32 с.

Містять методичні вказівки до виконання лабораторних робіт для студентів, рекомендації щодо розробки та відлагодження програм, оформлення звітів, список рекомендованої літератури.

Призначено для студентів спеціальності 7.080401 "Інформаційні управляючі системи і технології" під час проведення лабораторних занять та виконання самостійних робіт з дисципліни „Об'єктно-орієнтоване програмування".

© КНУБА, 2009

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Застосування об’єктно-орієнтованої методології – найсучасніший та найпоширеніший підхід до процесу розробки програмного забезпечення, що може бути основою побудови програмних систем будь-якого рівня складності і для будь-якої предметної області. Оволодіння основами об’єктно-орієнтованої методології не тільки збільшує суму знань та вмінь з програмування, але також формує принципово новий спосіб мислення, необхідний на етапах аналізу, проектування і програмування сучасних складних програмних систем.

Цикл лабораторних робіт з дисципліни «Об’єктно-орієнтоване програмування» ставить за мету:

- закріплення теоретичних знань в галузі об’єктно-орієнтованої методології розробки програмного забезпечення (ПЗ);
- набуття практичних вмінь і навичок, необхідних для раціонального використання засобів об’єктно-орієнтованого програмування та об’єктно-орієнтованого проектування ПЗ;
- ознайомлення студентів з програмним інструментарієм створення складних програмних систем для різноманітних предметних областей та формування навичок об’єктно-орієнтованого програмування на мовах Delphi Pascal та C++.

Після виконання циклу лабораторних робіт студенти повинні

- **знати** основні можливості об’єктно-орієнтованих мов і засоби їх використання; будову і загальні підходи до використання засобів стандартних бібліотек.
- **вміти** виділити на етапі проектування і описати засобами C++ життєздатну абстракцію предметної області з використанням принципу інкапсуляції даних; побудувати ієрархію успадкування з використанням принципів поліморфізму; використовувати можливості узагальненого програмування для опису класів, функцій, при роботі зі стандартною бібліотекою;
- **мати уяву про** сучасні засоби автоматизації процесу об’єктно-орієнтованого проектування; системи візуального програмування; існуючі об’єктно-орієнтовані мови програмування і сфери їх використання.

Лабораторна робота №1. КЛАСИ ТА ОБ'ЄКТИ. ПОЛЯ, МЕТОДИ, КОНСТРУКТОРИ ТА ДЕСТРУКТОРИ

Завдання

1. Описати мовою DelphiPascal та C++ клас для відрізка TLine. Поля класу – координати кінців відрізку x1, y1, x2, y2; методи – конструктор, в якому проводиться ініціалізація полів класу та функція Length для визначення довжини відрізка.
2. Описати два екземпляри класу TLine – L1, L2.
3. Розробити програмний інтерфейс з користувачем, який надає змогу ввести дані для двох відрізків та вивести на екран всі розрахункові значення.

Хід роботи

1. Запустіть Delphi, відкрийте новий проект, додайте до проекту новий модуль (опція меню File / New / Unit).
2. Збережіть проект в своєму робочому каталозі, задавши імена для модуля форми – Line_fm.pas, для нового модуля – Line_cl.pas, для файлу проекту – Line_pro.dpr.
3. В модулі Line_cl.pas опишіть заголовок класу TLine в розділі interface та реалізацію класу в розділі implementation.
4. В розділі interface модуля також оголосіть екземпляри класу TLine – L1, L2.
5. Під'єднайте модуль класу Line_cl до модуля форми Line_fm.pas.
6. Розробіть інтерфейс з користувачем (рис. 1), складіть опис прецеденту (див. дод. А).

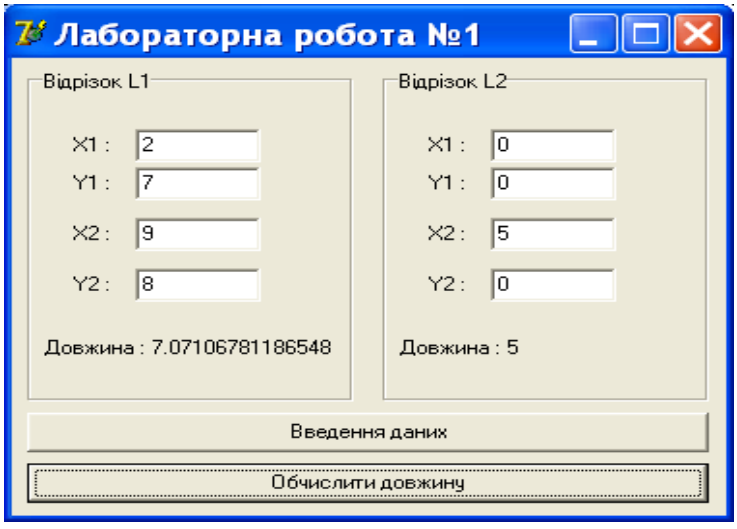


Рис. 1. Головне вікно програми

7. Задайте три обробники подій:
   - для кнопки «Введення даних» – обробник OnClick – створення екземплярів класу TLine L1 та L2, використовуючи дані з полів введення Edit;
   - для кнопки «Обчислення довжини» – обробник OnClick – обчислення довжин відрізків L1, L2;
   - для форми – обробник OnClose – руйнування екземплярів L1, L2 (виклик методів Free).

8. Збережіть проект та запустіть на виконання, проведіть тестування.

9. Додайте до класу метод для визначення координат центра відрізка. Внесіть необхідні зміни в опис самого класу, в інтерфейс з користувачем та в опис прецеденту.

10. Опишіть в звіті структуру програми у вигляді UML-діаграми компонентів (див. дод. Б).

11. Запустіть C++Builder (або будь-яке середовище розробки C++). Відкрийте новий проект.

12. Додайте до проекту новий модуль (cpp-файл та файл-заголовок). В h-файлі опишіть клас tline для відрізка. В cpp-файлі опишіть реалізацію цього класу.

13. Під'єднайте h-файл з описом класу до модуля форми (головного файлу).

14. Розробіть інтерфейс з користувачем аналогічно до програми на Delphi, задайте відповідні обробники подій та створіть два екземпляри класу.

15. Збережіть та протестуйте розроблену програму. Опишіть структуру C++-програми.

Самостійна робота

Оберіть варіант завдання згідно з останньою цифрою у вашому номері в списку групи, якщо остання цифра 0, то – варіант 10.

1. Визначити клас для прямокутника. Поля класу – координати лівого верхнього та правого нижнього кутів. Методи – конструктор, визначення периметру та площі.
2. Визначити клас для ромба. Поля класу – довжини діагоналей. Методи – конструктор, визначення довжини сторони та площі.

3. Визначити клас для трикутника. Поля класу – довжини сторін. Методи – конструктор, визначення периметру та площі.
4. Визначити клас для прямокутного трикутника. Поля класу – довжини катетів. Методи – конструктор, визначення гіпотенузи та площі.
5. Визначити клас для кола. Поля класу – координати центра та радіус. Методи – конструктор, визначення довжини та площі.
6. Визначити клас для квадрата. Поле класу – довжина діагоналі. Методи – конструктор, визначення сторони, периметру та площі.
7. Визначити клас для кулі. Поля класу – координати центра та радіус. Методи – конструктор, визначення площі поверхні та об'єму.
8. Визначити клас для конуса. Поля класу – радіус основи та висота. Методи – конструктор, визначення площі поверхні та об'єму.
9. Визначити клас для прямокутного паралелепіпеда. Поля класу – довжини ребер. Методи – конструктор, визначення площі поверхні та об'єму.
10. Визначити клас для циліндра. Поля класу – радіус основи та висота. Методи – конструктор, визначення площі поверхні та об'єму.

Оформлення звіту

Звіт оформлюють для лабораторної роботи (ЛР) і самостійної роботи (СР). Звіт включає:

- завдання;
- опис класів (UML- діаграму класів) з описом полів та методів класу. Для поля - назва поля, тип, призначення. Для методу - опис призначення, математична постановка задачі, тестові розрахунки;
- головне вікно програми з описом призначення елементів управління, заданих обробників подій для них та дій користувача під час роботи програми (у вигляді опису сценарію прецеденту);
- опис структури програм із зазначенням призначення кожного вхідного файлу (у вигляді UML-діаграми компонентів);
- роздрук тексту програми.

Контрольні запитання

1. Дайте визначення класу, об'єкту (екземпляру класу), поля, методу.
2. Дайте визначення конструктора та деструктора.
3. Яке ім'я мають конструктор та деструктор класу в Delphi Pascal ? А яке в C++?
4. Чим метод класів Delphi Free відрізняється від деструктора класу?
5. Який синтаксис створення екземплярів класу мовами Delphi Pascal та C++?

Лабораторна робота №2. ДИРЕКТИВИ ВИДИМОСТІ ТА ДОСТУП ДО ЗАХИЩЕНИХ ЕЛЕМЕНТІВ КЛАСУ

Завдання 1

1. Описати мовою DelphiPascal та C++ клас для кута TAngle. Поле класу FAngle – значення кута в радіанах, методи – конструктор, Cosinus для визначення відповідного значення.
2. Розробити програмний інтерфейс з користувачем, який надає змогу ввести дані та отримати значення косинусу (рис.2).

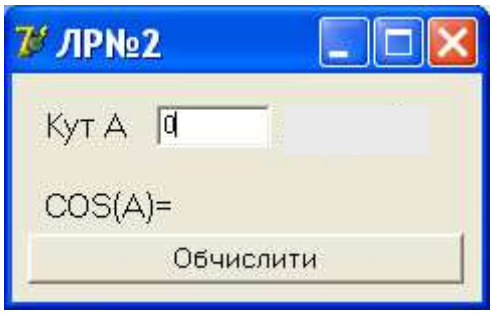


Рис. 2. Головне вікно програми

Хід роботи

1. Запустіть Delphi, відкрийте новий проект, додайте до проекту новий модуль, в якому опишіть клас.
2. Під'єднайте модуль класу до модуля форми. Оголосіть екземпляр класу A та створюйте його під час створення форми.
3. Під час закриття форми не забудьте зруйнувати екземпляр класу.
4. В обробнику OnClick кнопки «Обчислити» задайте такі дії:

- запис значення кута, що задане користувачем в рядку введення Edit, в поле класу (*Не забудьте, що тригонометричні функції потребують значення кута в радіанах!!!*);
- виклик методу класу для обчислення косинуса і виведення цього результату в Label.

5. Перевірте працездатність програми, задавши декілька значень кута.

Завдання 2

1. Описати поле класу FAngle як приватне, залишивши публічним метод Cosinus.
2. Описати в класі властивість-property Angle для доступу к захищеному полю FAngle.

Хід роботи

6. Додайте до опису класу відповідні директиви видимості, методи читання та запису для поля FAngle та опис property Angle.
7. В **модулі форми** змініть всі звернення до поля FAngle на властивість Angle
8. Перевірте працездатність програми, задавши декілька значень кута.

Завдання 3

Ніхто не буде сперечатися з тим, що введення значення кута в радіанах не дуже зручно. Історично ☺ так склалося, що ми звикли працювати зі значеннями кутів у градусах. Наступне завдання полягає в тому щоб надати можливість користувачу при введенні даних величину кута задавати в градусах, але в полі FAngle зберігати величину кута в радіанах, тому що метод визначення косинусу потребує все ж таки значення кута в радіанах. Виникає питання: коли здійснювати перетворення градусів в радіани? Це перетворення доцільно задати в методах читання/запису для property Angle.

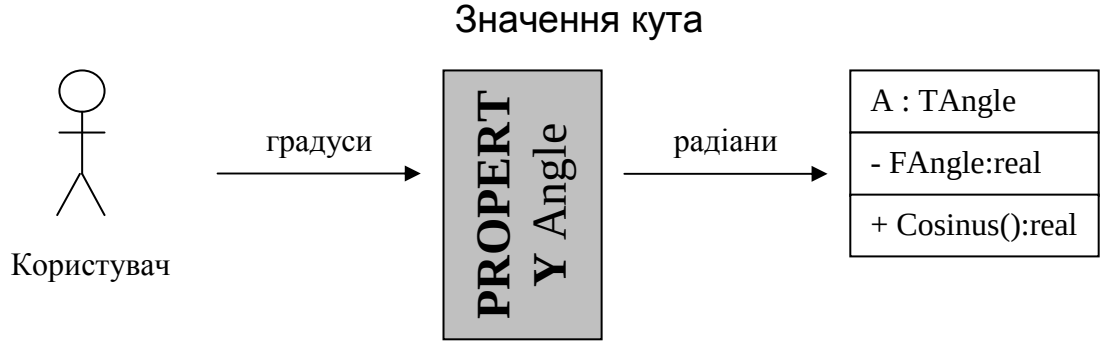


Рис. 3. Перетворення значення кута

Хід роботи

9. Змініть:
   - метод читання так, щоби значення поля FAngle перетворювалося в відповідне значення в градусах і поверталося як результат;
   - метод запису так, щоби вхідний параметр перетворювався в значення кута в радіанах і записувався в поле FAngle.
10. Внесіть зміни в інтерфейс з користувачем (рис. 4).

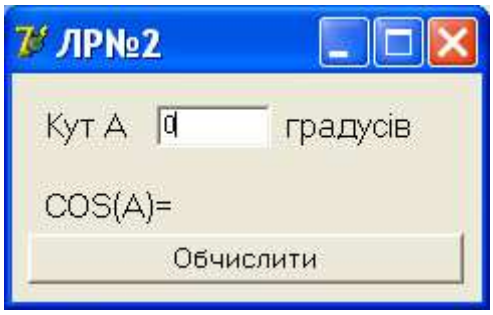


Рис. 4. Головне вікно програми

11. Протестуйте роботу програми.
12. Задайте для кута значення 90°. Математично Cos(90)=0, але ж на формі ви побачите не 0, а дуже мале значення. Додайте до функції Cosinus додаткову перевірку для уникнення втрати точності.

Завдання 4. Опишіть цей клас мовою C++ та оформіть звіт так, як описано в ЛР№1.

Контрольні запитання

1. Дайте визначення поняттю інкапсуляція в ООП.
2. Які директиви видимості застосовуються при описі класів в Delphi Pascal? Які в C++?
3. Які обмеження накладають директиви видимості на доступ до полів та методів класів?
4. Поняття властивості в об’єктній моделі Delphi.
5. Синтаксис опису простих властивостей в Delphi.
6. Як має бути описаний метод читання поля, як метод запису?
7. Чи може властивість бути тільки для читання або тільки для запису? Наведіть приклади, коли доцільно використовувати такі властивості?

8. Синтаксис опису властивостей - масивів та індексованих властивостей в Delphi.
9. Чи є властивості в мові C++? Як в мові C++ здійснити доступ до захищених членів класу?

Лабораторна робота № 3. ОДИНИЧНА СПАДКОВІСТЬ

Завдання 1

1. Опишіть ієрархію класів мовою DelphiPascal: батьківський -- точка TPoint, дочірній – відрізок TLine (можна використати клас з ЛР№1). Опис елементів класу (полів та методів) подані на рис. 5. Елементи класу опишіть як загальнодоступні (public). Кожен клас опишіть в окремому модулі, створивши проект, опис структури якого подано на рис. 6.
2. Опишіть та створіть один екземпляр класу TPoint та один екземпляр класу TLine, ввівши на формі відповідні значення полів для кожного екземпляру класу (недоступні елементи управління у вікні активізуються після створення екземплярів класу (рис. 7)).

Хід роботи

1. Запустіть Delphi, відкрийте новий проект, додайте до проекту один новий модуль, а другий з ЛР№1.
2. Збережіть файли проекту з відповідними іменами (рис. 6).
3. Утворіть зв’язки міх модулями проекту за схемою (рис. 6).
4. Опишіть клас TPoint в модулі TPoint_cl.
5. Задайте для TLine батьківськім класом клас TPoint. Видаліть з опису класу TLine ті властивості, що успадковуються від батьківського класу TPoint.
6. Опишіть екземпляри P: TPoint та L:TLine в інтерфейсних частинах модулів з описом відповідних класів.
7. Задайте обробники подій для кнопок головного вікна, задавши в них створення екземплярів класу та виведення необхідних даних у вікно.
8. Під час закриття форми не забудьте зруйнувати екземпляри класів.
9. Запустивши програму на виконання, перевірте її працездатність.

Завдання 2

1. Зробіть поля класу TPoint захищеними (protected), а методи відкритими (public). Зробіть команду з головного меню Delphi – Project | Build, спробуйте запустити програмне застосування на виконання та поясніть причину помилки, що ви отримаєте. Оформіть звіт. Дайте письмові відповіді в звіті на такі запитання:

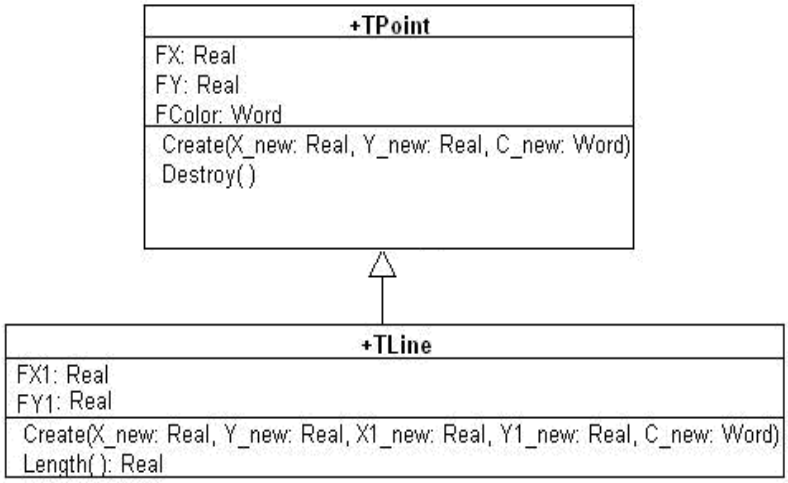


Рис. 5. Опис ієрархії класів (UML-діаграма класів)

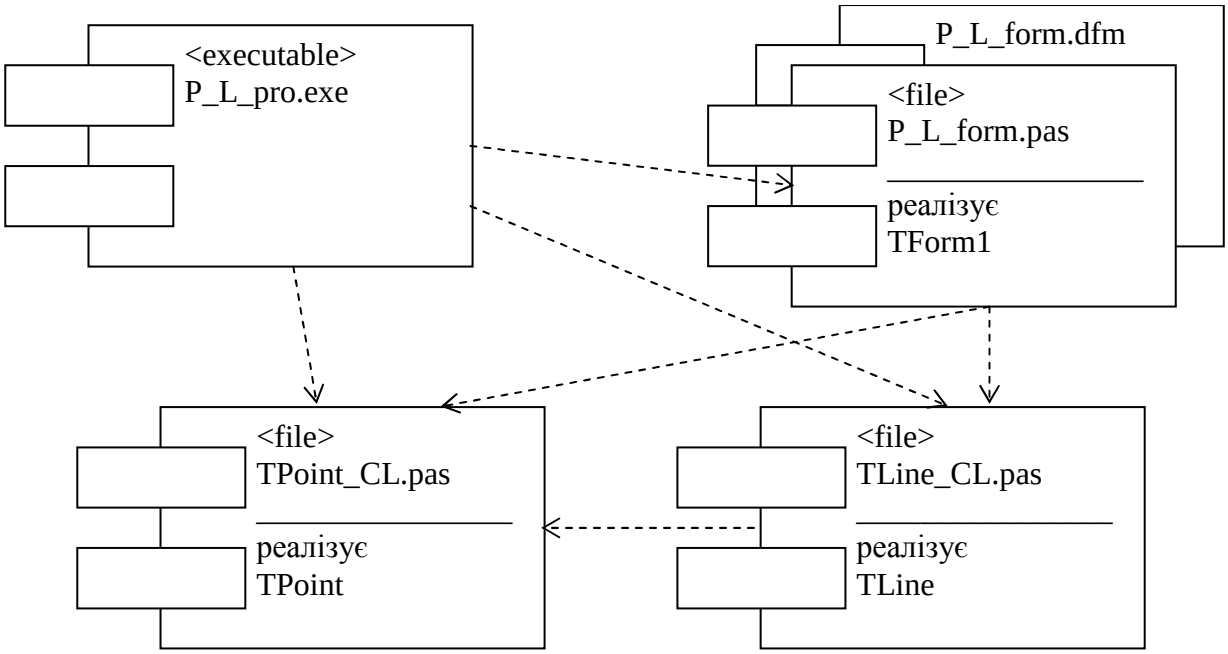


Рис. 6.Структура проекту в Delphi (UML-діаграма компонентів)

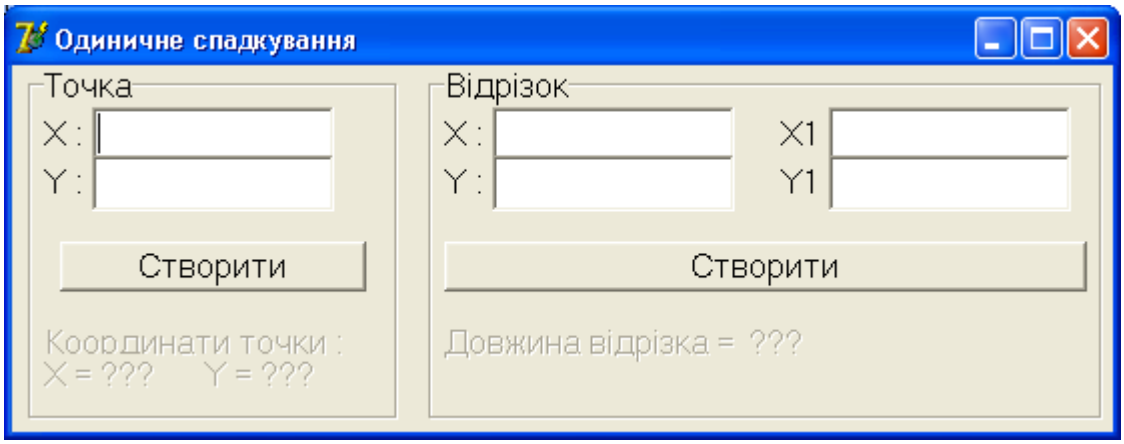


Рис. 7. Головне вікно програми

- чому при зміні директиви видимості полів класу TPoint на protected значення цих полів стає неможливим вивести на форму?
- чому клас TLine, хоча описаний в іншому модулі, все ж таки має доступ до успадкованих полів X та У?

2. Змінивши директиву видимості полів класу TPoint з protected на private, перезапустіть програму, поясніть причину «нової» помилки.

Завдання 3

1. Опишіть цю ієрархію класів мовою C++, вказавши ключ доступу до батьківського класу public.
2. Застосуйте різні директиви видимості при описі полів батьківського класу. Як це вплине на можливість використання членів батьківського класу класом-нащадком?
3. Застосуйте під час спадкування різні ключі доступу до батьківського класу. Як це вплине на можливість використання членів батьківського класу класом-нащадком?
4. Побудуйте для C++ застосування UML-діаграму компонентів.

Самостійна робота

1. Додайте на форму вікно ColorDialog для вибору під час створення екземплярів класу кольору точки та відрізка. Використайте ці кольори під час виведення властивостей точки та відрізка на форму.
2. Задайте поля точки та відрізка як власні, опишіть властивості-property для доступу до цих полів.

Контрольні запитання

1. Дайте визначення поняттю спадковість.
2. Що таке одинична спадковість? Що таке множинна спадковість?
3. Який клас є спільним предкам за замовчуванням для всіх класів мови Delphi Pascal?
4. Як директиви видимості, що застосовуюся під час опису класів впливають на можливість використання успадкованих властивостей в Delphi Pascal?
5. Як директиви видимості, що вказуються під час спадкування в C++, впливають на видимість успадкованих членів класу?

Після завершення ЛР№3 можна розпочинати виконання індивідуального завдання з дисципліни ООП!!!

Лабораторна робота № 4. ПЕРЕВИЗНАЧЕННЯ (ЗАМІЩЕННЯ) МЕТОДІВ КЛАСУ

Завдання 1

1. До класу точки Delphi та C++ з ЛР№3 додайте віртуальний метод Draw (Canvas:TCanvas), який виводить зображення точки. Параметром цього методу є канва, на якій буде відтворюватися це зображення.
2. Перевизначте цей метод в класі TLine.
3. На головне вікно програми додайте два компоненти PaintBox (вкладка VCL System): один – для виведення зображення точки інший - відрізка.
4. Додайте до обробників OnClick кнопок «Створити» виклики відповідних методів Draw, передавши в них як параметр відповідні канви.

Завдання 2. Динамічний поліморфізм

1. Опишіть в проекті Delphi два екземпляри класу-точки.
2. Створіть перший екземпляр класу через виклик конструктора класу точки, а другий через виклик конструктора класу-лінії.
3. Викличте методи Draw для обох екземплярів класу. Які зображення будуть виведені на екран? Чому, на вашу думку, спричинена ця різниця?
4. Опишіть в C++ програмі два покажчики на клас точки.
5. Виділіть пам’ять операцією new для першого покажчика як для екземпляру класу точки, а для другого – як для екземпляру лінії.
6. Викличте через покажчики метод Draw. Які зображення будуть виведені на екран?

Завдання 3

1. Оформіть звіт. Побудуйте для даної ієрархії класів UML-діаграму класів, а для програмного застосування UML-діаграму компонентів.
2. Визначте, які виключні ситуації можуть виникнути під час роботи застосування. Передбачте їх обробку в програмі, та внесіть їх в опис сценарію прецедента.

Самостійна робота

1. Відкрийте новий проект, додайте до нього новий модуль.

2. Опишіть в новому модулі клас, в якому один віртуальний метод VirtMethod, а інший – звичайний RegMethod (для спрощення роботи нехай ці методи просто виводять повідомлення виду: «Ім’я класу – ім’я методу»).
3. Викличте з методу RegMethod віртуальний VirtMethod.
4. Створіть екземпляр класу, викличте для цього методу RegMethod.
5. Після перевірки працездатності програми додайте нащадок для описаного вище класу, в якому перевизначте віртуальний метод, вивівши на екран «ім’я класу-нащадка – ім’я методу».
6. Створіть екземпляр цього класу та викличте для нього метод RegMethod. Які повідомлення ви побачите на екрані?
7. Дайте визначення динамічного поліморфізму.

Контрольні запитання

1. Дайте визначення поняття поліморфізм.
2. Як описуються заміщувані методи класів в Delphi та C++?
3. Як перевизначити в класі-нащадку заміщувані методи?
4. Для чого використовувати заміщувані методи в класах?
5. Поняття раннього та пізнього зв’язування.
6. Для заміщуваних методів класу використовується раннє чи пізнє зв’язування? Для яких методів класу використовується раннє зв’язування?
7. Як визначаються адреси заміщуваних методів класу?
8. Чим відрізняються принципи організації таблиць віртуальних та динамічних методів в Delphi?
9. Чи є в C++ таблиці віртуальних методів, якщо так, то як вона називається?
10. Які переваги (чи недоліки) в використанні віртуальних та динамічних методів: коли доцільно робити метод віртуальним, а коли динамічним?
11. Що таке динамічний поліморфізм?
12. На вашу думку, в чому полягає доцільність використання динамічного поліморфізму?

Лабораторна робота № 5. АБСТРАКТНІ ТА КЛАСОВІ МЕТОДИ DELPHI. ЧИСТІ ТА СТАТИЧНІ МЕТОДИ C++

Завдання 1

Додайте до ієрархії класів точка-відрізок батьківський клас для точки TGraphicFigure, в якому задаються поля – координати точки прив'язки та колір; методи – конструктор та абстрактний віртуальний метод Draw.

Хід роботи

1. Відкрийте проект з ЛРN№4. Додайте до модуля з класом точки опис класу TGraphicFigure, перемістивши до нього поля з класу точки, властивості, конструктор та заголовок методу Draw.
2. Додайте до заголовку методу Draw в класі TGraphicFigure директиву abstract.
3. В класі точки опишіть цей метод директивою override.
4. Перевірте працездатність програми.
5. Спробуйте, створивши екземпляр класу TGraphicFigure, викликати метод Draw. Запишіть в звіт, яка виключна ситуація виникне при цьому.
6. Дайте відповіді на запитання: чому в класі TGraphicFigure метод Draw має бути абстрактним?
7. В звіті побудуйте UML-діаграму класів для класів TGraphicFigure, TPoint та TLine.

Завдання 2.

Опишіть зазначену ієрархію класів мовою C++, задавши абстрактний метод Delphi як чистий C++.

Контрольні запитання

1. Дайте визначення абстрактних Delphi (чистих C++) методів.
2. Обґрунтуйте доцільність використання абстрактних методів?
3. Який синтаксис опису абстрактних (чистих) методів?
4. Що таке абстрактний клас? Чи доцільно створювати екземпляри абстрактних класів?

Завдання 3

Опишіть для визначених класів класові функції, які повертають інформацію про назви та типи полів класів.

Завдання 4

Побудуйте для даної ієрархії класів UML-діаграму класів, а для програмного застосування UML-діаграму компонентів.

Хід роботи

1. Для класу TGraphicFigure додайте класову функцію Info, яка повертає рядок з інформацією про назви і типи полів цього класу. Опишіть цю функцію як віртуальну.
2. Подумайте, чи доцільно перевизначати цю функцію в класі точки?
3. В класі відрізка перевизначте функцію Info, при цьому під час формування рядка-інформації спочатку викличте успадковану функцію від класу-предка, застосувавши директиву inherited, а потім до її результатів додавайте інформацію про поля, що набуті відрізком.
4. Додайте до головного вікна компоненти, в які буде виводитися інформація про класи до створення екземплярів цих класів (за синтаксисом *ім'я_класу.класова_функція*).
5. Перевірте працездатність програми.
6. Перевірте, чи можна викликати класові функції після створення екземплярів класу (за синтаксисом *екземп_класу.класова_функція*).

Завдання 4.

1. Опишіть класові методи Delphi як статичні методи мовою C++.
2. Перевірте виклик цих методів до створення екземплярів класів та після їх створення.
3. Опишіть, наприклад, в класі TGraphicFigure одне статичне поле.
4. Змініть значення цього поля через один екземпляр класу.
5. Перевірте, чи вплинуло це на значення цього поля в інших екземплярах класу. Який висновок можна зробити за результатами перевірки?

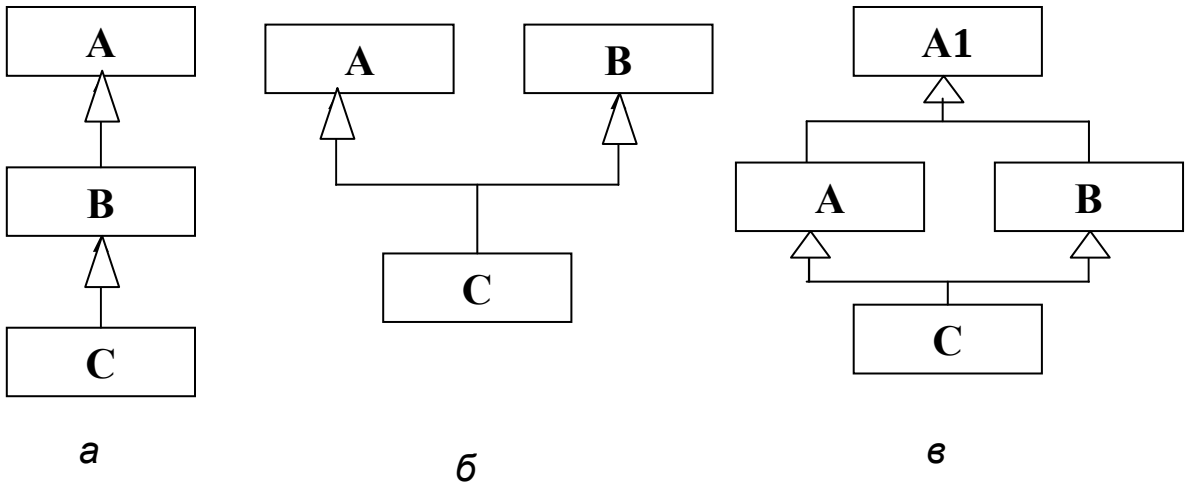
Контрольні запитання

1. Дайте визначення класовим Delphi (статичним C++) методам? Яка доцільність їх використання?
2. Які класові методи надає своїм нащадкам клас TObject Delphi?
3. Який синтаксис опису класових (статичних) методів?
4. В чому особливості використання статичних полів класу в C++?
5. Для чого використовується директива inherited в мові Delphi Pascal?
6. Який синтаксис виклику успадкованих методів, що були перекриті нащадком, в C++?

Лабораторна робота № 6. ОСОБЛИВОСТІ СПАДКОВОСТІ В C++

Завдання 1. Виклик конструкторів та деструкторів при спадкуванні C++

1. Опишіть ієрархію класів (використовуючи при спадкуванні ключ доступу до батьківського класу public), що подана на рис. 8,*а* мовою C++. Для кожного класу задайте по одному відкритому полю (для класу A – поле int a; для класу B – int b, для C – int c), а також конструктори та деструктори, в яких проводиться ініціалізація полів та виводиться повідомлення про створення (або знищення) екземпляру класу (наприклад, «Hello from A!!!» для конструктора та «Bye from A!!!» для деструктора класу A).



а *б* *в*

Рис. 8. Ієрархія спадковості класів

2. Створивши екземпляр класу C, визначити послідовність виклику конструкторів та деструкторів для ієрархії класів (опишіть цю послідовність в звіті).

Завдання 2. Множинне спадкування

Перетворіть ієрархію класів в ту, що задана на рис. 8,*б*. Чи зміниться щось при виклику конструкторів та деструкторів (опишіть цю послідовність в звіті)?

Завдання 3. Віртуальні предки при множинному спадкуванні

1. Додайте опис класу A1 та перетворіть ієрархію класів в ту, що задана на рис. 8,***в***. Опишіть в класі A1 відкрите поле a1, конструктор та деструктор за аналогією з класом A.
2. Створіть екземпляри кожного класу. Дослідить послідовність виклику конструкторів та деструкторів.
3. Зверніться до поля a1 з екземпляру класу C.

Для уникнення неоднозначності при зверненні до членів класу A1 з класу-нащадка C використовуйте оголошення віртуального предка при описі класів A та B.

Контрольні запитання

1. Що таке одинична спадковість? Як в C++ описати клас-нащадок?
2. Для чого при описі спадковості вказуються ключі доступу до батьківських класів? На що вони впливають?
3. Чому при описі конструктору дочірнього класу необхідно явно викликати конструктори батьківських класів. Який синтаксис їх виклику?
4. Що таке множинна спадковість? Як в C++ описати клас-нащадок від декількох класів-предків?
5. Які проблеми можуть виникнути при множинній спадковості?
6. Як уникнути неоднозначності, що виникає при ромбоподібний множинній спадковості C++?

Лабораторна робота № 7. ПЕРЕВАНТАЖЕННЯ МЕТОДІВ КЛАСУ

Завдання 1. Перевантажені методи в Delphi

Опишіть клас для одновимірного масиву дійсних чисел. Клас містить два приватних поля – поточна кількість елементів масиву FN та безпосередньо сам масив FA (загальна кількість задається певною константою, наприклад, Nmax = 100, причому обов’язково FN<=Nmax).

Опишіть дві властивості (property) – просту властивість для доступу до поля FN та властивість масиву для доступу до FA.

Опишіть метод, який зчитує елементи масиву зі StringGrid та метод, що обчислює такі значення (оберіть варіант завдання згідно з останньою цифрою у вашому номері в списку групи, якщо остання цифра 0, то – варіант 10).

1. Середнє арифметичне елементів масиву.
2. Середнє геометричне елементів масиву.
3. Кількість від'ємних елементів.
4. Кількість додатних елементів.
5. Максимальне значення серед значень елементів масиву.
6. Мінімальне значення серед значень елементів масиву.
7. Кількість змін знаку в масиві.
8. Кількість парних елементів.
9. Суму елементів масиву.
10. Добуток елементів масиву.

Для класу, що був описаний додайте перевантажені конструктори:

- перший конструктор без параметрів під час створення ініціалізує елементи масиву 0;
- другий конструктор зчитує з текстового файлу кількість елементів в масиві з першого рядка файлу та значення цих елементів з наступних рядків. Як параметр в цей конструктор передається ім'я файлу.

В звіті опишіть зміни, які відбулися в Delphi при обробці файлів. Які причини викликали ці зміни?

Завдання 2. Опишіть мовою C++ аналогічний клас

Контрольні запитання

1. Дайте визначення перевантаженим методам?
2. Яка директива застосовується під час перевантаження методів в Delphi? Чи необхідно використання директив в C++?
3. Чим розрізняються перевантажені підпрограми? Чи повинні вони мати однакове ім'я? Чим можуть відрізнятися списки параметрів? Чи повинні перевантажені підпрограми реалізовувати однаковий алгоритм роботи?
4. Наведіть приклади неоднозначності, яка може виникнути під час перевантаження підпрограм.
5. Як під час виклику перевантажених підпрограм визначається, яка саме підпрограма буде викликана фактично?

Лабораторна робота № 8. ПЕРЕВАНТАЖЕНІ ОПЕРАЦІЇ МОВИ СІ

Завдання 1. Перевантаження бінарних та унарних операцій

Для С++ класу точки з Л.Р.№3 перевантажити:

- бінарні операції додавання та віднімання;
- унарні інкрементні (++) та декрементні (--) операції в префіксній та постфіксній формах запису.

Завдання 2. Перевантаження операції присвоєння та індексування. Використання конструктору копії

Описати на С++ клас для рядка. Поля класу – покажчик на рядок, який формується динамічно, та поточна довжина рядка. Методи – конструктори, для створення порожнього рядка та рядка заданого користувачем, конструктор копії для класу-рядка.

Перевантажити для класа-рядка операції:

- присвоєння рядків;
- індексування;
- конкатенації рядків, порівняння == (дорівнює) та != (не дорівнює);
- логічну операцію ! (not) так, щоби послідовність букв в рядку змінювалася на зворотну;
- перевантажте операцію конкатенації рядків таким чином, щоби до об’єкта-рядка можна було додати константу-рядок, а також до константи-рядка додати об’єкт-рядок.

Контрольні запитання

1. Для чого в С++ застосовується перевантаження операцій?
2. Який синтаксис перевантаження бінарних операцій?
3. Для якого з об’єктів obj1 чи obj2 буде викликана перевантажена операція додавання у виразі obj1+obj2? Який з цих об’єктів стане фактичним параметром цієї операції?
4. У чому особливість перевантаження операції віднімання? Наведіть ще приклад бінарної операції, при описі якої необхідно стежити за послідовністю операндів у виразі?
5. Який синтаксис перевантаження унарних операцій?
6. Що таке this? Чому унарні операції повертають this як результат? Яким ідентифікатором в Delphi позначається покажчик this?

7. Назвіть випадки, коли в С++ програмах створюється «нелегальна» копія об’єкта? В яких випадках це може зашкодити коректній роботі програми?
8. Який синтаксис опису конструктора копії? Чи є він перевантаженим конструктором?
9. Для чого використовуються дружні функції? Чи можуть бути дружніми класи?
10. Якщо клас А оголошений другом класу В, чи є ця дружність «взаємною», тобто, чи отримає клас В доступ до захищених членів класу А?

Лабораторна робота № 9. БІБЛІОТЕКА СТАНДАРТНИХ ШАБЛОНІВ STL МОВИ С++

Завдання 1. Використання класу-шаблону vector

1. Створити вектор нульового розміру для цілих.
2. Вивести на екран початковий розмір вектора (size()).
3. Задавши в діалоговому режимі кількість елементів вектора, ввести дані, додаючи їх в кінець вектора (push_back()), вивести на екран новий розмір вектора, та всі його елементи.
4. Додати до вектора ще декілька елементів та вивести дані про розмір та сам вектор на екран.
5. Описати ітератор на елементи вектора та провести виведення елементів на екран через ітератор.

```
vector<int>: : iterator p = v.begin();    // доступ через ітератор
while (p != v.end()) { cout<< *p <<" "; p++; }
```

6. Провести вставку всередину вектора нових елементів (в діалоговому режимі та вставку заданої кількості однакових значень). Оновити дані про розмір і вміст вектора на екрані.
7. Виконати аналогічні п.6. дії по видаленню елементів з вектора.

Самостійна робота

Створити вектор для збереження екземплярів класу-студент, з полями прізвище, номер заліковки, оцінка. Не забудьте в цьому класі описати конструктор за замовчуванням та перевантажити операції присвоєння =, порівняння <, ==.

Завдання 2 Використання класу-шаблону list

1. Створити два списки, що складаються з дійсних значень.
2. Один список заповнити з кінця, використовуючи метод push_back(), а інший з початку, використовуючи push_front().
3. Відсортувати елементи списків, використавши метод sort().
4. Об’єднати елементи другого списку з першим, вивести на екран.

Самостійна робота

Опишіть нащадок списку-шаблону, надавши йому додаткову можливість збереження даних зі списку в файл. Створити список для збереження екземплярів власного класу (див. попередню самостійну роботу) і перевірте його працездатність.

Завдання 3 Використання класу-шаблону map

1. Створити асоціативний список для збереження десяти пар ключ/значення, де ключем є цифра від 0 до 9, а значенням символ.
2. Заповнити асоціативний список парами, використовуючи для утворення пар клас-шаблон pair<> (перший випадок) та метод make_pair() (другий випадок).
3. Задати пошук значення в списку по заданому ключу, використовуючи метод find().

Самостійна робота

Враховуючи те, що номер заліковки для кожного студента унікальний, він може вважатися ключем. Переробіть клас-студет для збереження його в асоціативному списку

Контрольні запитання

1. Що таке класи-контейнери? Які класи – контейнери ви знаєте?
2. Що таке ітератори? Назовіть п’ять видів ітераторів? В чому особливості їх використання?
3. Що таке шаблони С++? В чому доцільність використання шаблонів функцій (родових функцій) та класів-шаблонів (родових класів)?
4. Чим створення, наприклад, шаблонів функцій відрізняється від перевантаження функцій, адже шаблони також застосовуються для обробки даних різних типів?
5. Чому під час збереження в контейнерах власних класів необхідно перевизначити для цих класів операції порівняння та присвоєння?

Лабораторна робота № 10. ЗАСТОСУВАННЯ АЛГОРИТМІВ БІБЛІОТЕКИ STL

Завдання

До вектору цілих застосувати такі алгоритми:

- count() для визначення кількості нулів у векторі цілих;
- count_if() для визначення кількості додатних значень у векторі, задавши предикат для перевірки додатне число чи ні;
- remove_copy_if() заміни у векторі всіх парних значень на удвічі менше значення;
- transform() для довільної (за вибором студента) модифікації елементів вектору.

Самостійна робота

1. Застосувати алгоритми для підрахунку в списку студентів відмінників, хорошистів, трійочників та формування окремого списку для двійочників.
2. Переробити лабораторну роботу зі створення списку студентів на базі TList Delphi на C++ з використанням власного класу-шаблону списку (нащадка від класу-шаблону list).

Контрольні запитання

1. Що таке алгоритми STL?
2. Які алгоритми ви знаєте?
3. Що таке предикат? Які види предикатів ви знаєте?

Список літератури

1. *Дарахвелидае П. Г., Марков Е. П.* Программирование в Delphi 7. – СПб.: БХВ-Петербург, 2003. – 784 с.
2. *Ларман К.* Применение UML и шаблоны проектирования: учеб. пос. – М.: Вильямс, 2001. – 496 с.
3. *Методичні* вказівки до лабораторних робіт з дисципліни “Об’єктно-орієнтоване програмування” для студентів спеціальності ІУСТ. уклад. Красовська А.В. – К.: КНУБА, 2001. 58 с.
4. *Шилдт Г.* Самоучитель C++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003. – 688 с.

Додаток А

Опис прецеденту

Прецедент (use case - варіант використання системи) – документ, що описує послідовність дій користувача програмної систем (ПС). Можна сказати, що за допомогою прецедентів описують певний процес.

> *Процес (process)* від початку і до кінця визначає (задає) послідовність дій, необхідних для досягнення певного результату або надання певного знання користувачу ПС.
> *Прецедент* – це опис відносно великого завершеного процесу, в який зазвичай входить багато дій (кроків), які самі по собі не є самодостатніми.

Наприклад, якщо розглядати як прецедент регістрацію користувача в системі, то окремі кроки введення логіну та паролю недоцільно розглядати як окремі прецеденти, тому що само по собі введення логіну недостатньо для завершення регістрації.

Прецедент описується в таблиці, що зазвичай складається з двох частин: типового ходу подій та альтернатив.

Типовий хід подій	
Дії виконавця	*Відгук системи*
…	
Альтернативи	
Дії виконавця	*Відгук системи*
…	

Опис прецеденту слід починати за схемою:

цей прецедент починається, коли <Користувач> <ініціює подію>.

На кожну подію виконавця описується відгук системи або дії, що виконуються системою як реакція на ці дії. Якщо В типовому ході подій описуються дії виконавця, що виконуються під гаслом: «Політ іде нормально!!!» ☺, тобто без розгляду можливих виключних (аварійних) ситуацій. Все, що може бути «не так», описується в частині «Альтернативи». При цьому кожен пункт таблиці «Альтернативи» має номер, що відповідає пункту типового ходу подій (див. приклад).

До опису прецеденту додається загальний вигляд робочого вікна, на якому зазначаються елементи програмного інтерфейсу з користувачем.

Типовий хід подій	
Дії виконавця	**Відгук системи**
1. Користувач запустив на виконання програму «The Best»	*2. Виводиться вікно для введення логіну та паролю*
3. Користувач вводить в поле введення А логін.	*4. Під час введення в поле А системою відображаються списки автозаповнення.*

Продовження дод. А

Типовий хід подій	
Дії виконавця	**Відгук системи**
5. Після завершення введення логіну, користувач натискає клавішу <Enter> на клавіатурі, або клацає мишею в полі Б.	*6. Курсор переводиться в поле Б.*
7. Користувач вводить в поле Б пароль.	*8. Під час введення в поле Б символи маскуються.*
9. Користувач закінчує введення, натиснувши кнопку В «Вхід».	*10. Перевіряється правильність введення логіну та паролю. Якщо вірні, виводиться головне вікно системи.*
Альтернативи	
5. Після завершення введення логіну, користувач натискає кнопку В «Вхід».	*6. Подається звуковий сигнал, курсор переводиться в поле Б.*
5. Після завершення введення логіну, користувач натискає кнопку Г «Відмова».	*6. Програма закінчує свою роботу (зверніть увагу на нумерацію пунктів частини таблиці «Альтернативи»)*
9. Користувач закінчує введення, натиснувши кнопку В «Вхід».	*10. Виведення форми-анкети для регістрації даних користувача.*
9. Користувач закінчує введення, натиснувши кнопку Г «Відмова».	*10. Програма закінчує свою роботу*

Зверніть увагу на нумерацію пунктів частини таблиці «Альтернативи». Її рядку слугують якби заміною рядків типового ходу подій.

Опис прецеденту може містити *точки прийняття рішень* або розгалуження. Якщо одне з цих рішень є типовим, а інші рідкісними, незвичайними або винятковими, то тільки типові випадок описується в типовому ході подій, а інші в альтернативному. Але в момент прийняття рішень можуть з’явитися рівноважні, рівноімовірні альтернативи, тоді використовується така схема опису прецеденту:

- в основному розділі типового ходу подій вказуються всі можливі підрозділи;
- кожний підрозділ описується як типовий хід подій. Нумерація подій в кожному розділі починається з одиниці;
- якщо підрозділи також мають точки прийняття рішень, то вони записуються як альтернативи для кожного підрозділу.

Головний розділ	
Типовий хід подій	
Дії виконавця	*Відгук системи*
…	
Альтернативи	
…	
Підрозділ «Назва підрозділу»	
Типовий хід подій	
…	
Альтернативи	
…	

Наприклад, регістрацію нового користувача можна оформити як підрозділ головного прецеденту: пункт 10 типового ходу подій може бути сформований так: «*Перевіряється правильність введення логіну та паролю. Якщо вірні, виводиться головне вікно системи. Інакше див. підрозділ «Регістрація нового користувача»*».

Додаток Б

UML-діаграма компонентів

Діаграма компонентів (component diagram) призначена для візуалізації загальної структури програмної системи (архітектури програмної системи).

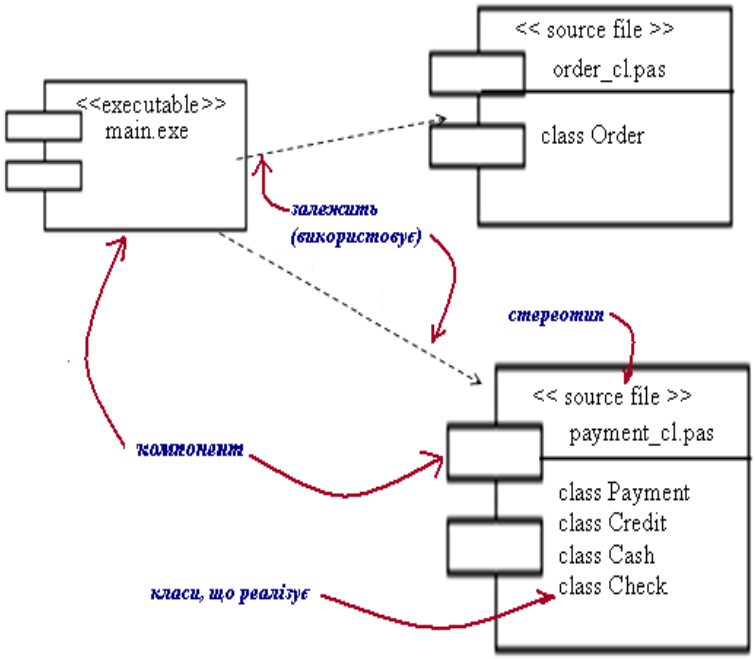


Рис. 1. Приклад діаграми компонентів

Мова UML виділяє такі види **компонентів**:

- виконуваний файл (executable).
- файл (file). Це може бути файли з вхідними текстами програми, файли заголовки (h-файли) та інш.;
- динамічна або статична бібліотека (library);
- документ (document);
- таблиця бази даних (table);
- також можуть виділятися Web-сторінки, файли довідки.

Для компонента може бути вказаний його **стереотип**, а також **класи**, які він реалізує.

Пунктирні стрілки відображають наявність **взаємозв'язку** між компонентами.

UML-діаграма класів

Діаграма класів (class diagram) відображає статичну структуру системи: класи, атрибути класів (поля та методи) та зв'язки між класами.

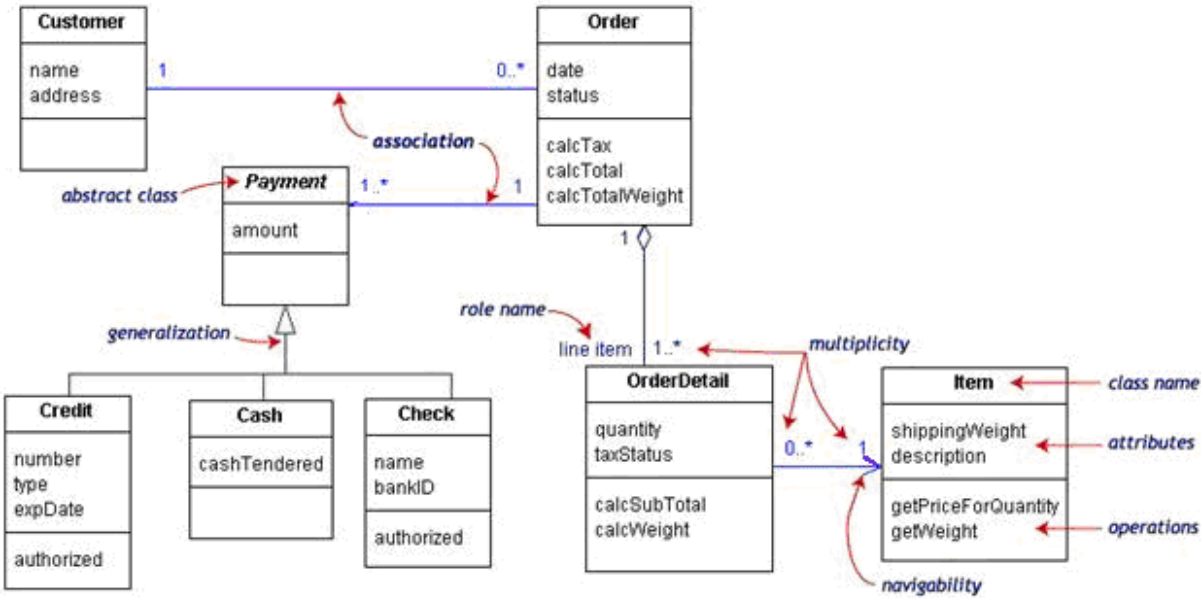


Рис. 2. Приклад UML-діаграма класів

Позначення класу – прямокутник, розділений на 3 частини: назва класу (class name), атрибути (attributes), операції (operation). **Відношення між класами** (association):

- асоціація – зв'язок між екземплярами класу (один об'єкт має знати про інший для успішної роботи);
- агрегація – клас належить (є частиною) іншого класу. Наприклад, клас Order (Замовлення) включає в себе декілька рядків-замовлень (Order has a collection of OrderDetails);
- композиція (окремий випадок агрегації) виникає, коли частина без цілого існувати не може;
- узагальнення (generalization) – відношення спадкування, зв'язок класу з суперкласом (Payment is a superclass of Cash, Check, and Credit).

Стрілка вказує **напрямок зв'язку** (navigability). Якщо стрілки немає – зв'язок працює в двох напрямках. Для зрозумілості кінець асоціації можна підписати **назвою ролі** (role name).

Кількість асоційованих об'єктів (multiplicity):

- 0..1 – нуль або 1 об'єкт;
- n.. m – від n до m об'єктів;
- 0..* або * – будь-яка кількість (включаючи відсутність об'єктів);
- 1 – точно один об'єкт;
- 1..* – не менше одного об'єкта.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчально-методичне видання

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні вказівки
до виконання лабораторних робіт для студентів
спеціальності 7.080401 „Інформаційні
управляючі системи та технології”

Укладач **КРАСОВСЬКА** Ганна Валеріївна

Комп’ютерне верстання *І.С. Черненко*

Підписано до друку 2009. Формат 60 × 84 1/ 16
Ум. друк. арк. 1,86. Обл.-вид. арк. 2,0.
Тираж 50 прим. Вид. № 55/ІІІ-09. Зам. №

КНУБА, Повітрофлотський проспект, 31, Київ, Україна, 03680

E-mail: red-isdat@knuba.edu.ua

Віддруковано в редакційно-видавничому відділі
Київського національного університету будівництва і архітектури

Свідоцтво про внесення до Державного реєстру суб’єктів
Видавничої справи ДК № 808 від 13.02.2002 р.