МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
Київський національний університет
будівництва і архітектури

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні вказівки
до виконання курсової роботи
для студентів, які навчаються за напрямом
підготовки 6.050101 «Комп'ютерні науки»
спеціалізації «Інформаційні
управляючі системи та технології»

Київ 2011

ББК 32.937.26-01
О-13

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

Рецензент І.М. Доманецька, канд. техн. наук, доцент

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

Затверджено на засіданні кафедри інформаційних технологій, протокол №17 від 11 квітня 2011 року

Видається в авторській редакції.

Об'єктно-орієнтоване програмування: методичні вказівки
О-13 до виконання курсової роботи / уклад.:
Г.В. Красовська. – К.:КНУБА, 2011. – 28 с.

Містять загальні положення, порядок виконання роботи, варіанти тем курсової роботи, методичні вказівки до виконання роботи, список літератури, додатки.

Призначено для студентів, які навчаються за напрямом підготовки 6.050101 «Комп'ютерні науки» спеціалізації «Інформаційні управляючі системи та технології».

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ 4
ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ 4
ЕТАПИ ВИКОНАННЯ КУРСОВОЇ РОБОТИ 5
ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ 6
МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ КУРСОВОЇ РОБОТИ 7

Опис предметної області 7

Опис функцій програми 7

Розробка концептуальної моделі предметної області 8

Розробка прецедентів (варіантів використання) 9

Розробка UML-діаграми класів 10

Опис системних операцій 11

Розробка UML-діаграми послідовностей 14

Розробка UML-діаграми компонентів (опис структури програми) 16
ПЕРЕЛІК ТЕМ КУРСОВИХ РОБІТ 16
РЕКОМЕНДАЦІЇ ЩОДО ДЕЯКИХ АСПЕКТІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТЕМ КУРСОВИХ РОБІТ 23

Створення довідників 23

Створення полів-списків 24

Перегляд файлів з зображенням, відео-файлів, прослуховування аудіо-файлів 24
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 25
ДОДАТОКИ 26

Загальні положення

Виконання курсової роботи є завершальним етапом вивчення курсу «Об'єктно-орієнтоване програмування». *Метою курсової роботи* є створення програмного застосування для обробки списків сталих об'єктів з використанням класів-контейнерів. Збереження та завантаження об'єктів відбувається з використанням потокового введення/виведення.

Під час виконання курсової роботи студентами на практиці застосовуються елементи об'єктно-орієнтованого аналізу предметної області та об'єктно-орієнтованого проектування програмного застосування.

Розробка програм повинна виконуватися з урахуванням вимог мінімізації об'єму інформації, що зберігається в класах, списках, бінарних деревах або інших структурах даних. Як тестові дані можуть використовуватися дані, що описані безпосередньо в програмі чи такі, що зберігаються на зовнішніх носіях.

Курсова робот вважається закінченою, якщо є готова налагоджена програма, отримано результат по всіх тестових прикладах роботи програми та оформлено звіт з виконання роботи, зміст якого регламентується даними методичними вказівками. Під час здавання курсової роботи студент має вільно орієнтуватися в теоретичному матеріалі, правильно відповідати на запропоновані запитання.

Робота виконується в термін, що встановлюється деканатом (тривалість виконання 8 тижнів). Під час виконання роботи студент кожного тижня доповідає викладачеві про хід виконання роботи.

Завдання до курсової роботи

Завдання до курсової роботи обирається студентом зі списку варіантів, що наведені нижче у методичних вказівках, самостійно або за рекомендацією викладача. Студент може

запропонувати власне формулювання теми курсової роботи, узгодивши це з викладачем.

Курсова робота виконується індивідуально кожним студентом. В окремих випадках з дозволу викладача допускається об'єднання студентів у групи (2-3 студенти) для роботи над складними чи комплексними темами.

Етапи виконання курсової роботи

Під час розробки курсової роботи можна виділити такі етапи:

Назва етапу	***Тривалість***
Вибір завдання, дослідження предметної області; розробка вимог до створюваної програми	2 тижні
Розробка класів та ескізів програмного інтерфейсу з користувачем	2 тижні
Розробка, налагодження, тестування програми та аналіз результатів тестування програми	3 тижні
Оформлення звіту та захист курсової роботи	1 тиждень
Загальна тривалість виконання КР	***8 тижнів***

Результатом дослідження предметної області є визначення основних функцій майбутньої програми та послідовності їх виконання, визначення основних сутностей предметної області та зв'язків між ними.

Виходячи з функцій майбутньої програми та побудованої концептуальної моделі предметної області, розробляються ієрархії класів (у вигляді UML-діаграми класів). Під час розподілу обов'язків між класами слід застосовувати шаблони проектування.

Також на основі аналізу предметної області розробляються варіанти використання майбутньої програми у вигляді UML-діаграми прецедентів з описом сценаріїв прецедентів.

Структура програми подається у вигляді UML-діаграми компонентів з описом призначення кожного компонента.

Під час розробки та тестування програми здійснюється підготовка набору тестів для перевірки правильності роботи програми, проводиться аналіз результатів тестування програми.

Після отримання позитивних результатів тестування програми та демонстрації її роботи викладачу оформлюється пояснювальна записка (звіт) до курсової роботи.

Завершальним етапом у виконанні курсової роботи є її захист, під час якого студент повинен вільно відповідати на запитання по змісту пояснювальної записки, тексту розробленої програми, а також на теоретичні запитання по курсу «Об'єктно-орієнтоване програмування».

Перелічені вище етапи є основою для складання календарного плану виконання роботи (див. бланк завдання у додатку Б).

Оформлення пояснювальної записки

Пояснювальна записка містить:

1. титульний лист (додаток А);
2. завдання до курсової роботи (додаток Б);
3. зміст;
4. вступ - опис предметної області та функцій програми у вигляді дерева функцій, концептуальна модель предметної області;
5. UML-діаграма прецедентів з описами сценаріїв прецедентів (реальні або ідеальні прецеденти).
6. UML-діаграма класів з описом полів та методів кожного класу (для кожного методу наводиться його специфікація або МПЗ у разі необхідності);
7. опис системних операцій та поведінки програми у вигляді UML-діаграм послідовностей для кожної системної операції;
8. опис структури програми у вигляді UML-діаграми компонентів з описом призначення кожного компонента;
9. опис тестових прикладів виконання програми, аналіз отриманих результатів;
10. список літератури, використаної при роботі над курсовою;

11. додатки з роздруком тексту програми та, у разі необхідності, роздруком файлів з вхідними/вихідними даними;

При оформленні звіту необхідно дотримуватися державних стандартів, що висуваються до оформлення технічної документації (див. [3])

Методичні рекомендації до виконання курсової роботи

Опис предметної області

Кожний програмний продукт розробляється для спрощення та (або) вдосконалення роботи спеціаліста в певній предметній області (СПО). Тому для того щоб програма дійсно була корисною, розробнику треба досконально вивчити, знати і розуміти ті дії (функції), які має виконувати СПО. Робота на цьому етапі неможлива без взаємодії з СПО, який і допомагає розробнику з'ясувати:

- які саме дії функції виконуються СПО;
- яка послідовність їх виконання;
- які особливості необхідно враховувати при реалізації кожної функції.

Враховуючи те, що виконання курсової роботи не передбачає досліджень роботи реальних підприємств, студент після вибору теми курсової узгоджує з викладачем та оформлює письмово опис предметної області за своєю темою. Опис предметної області обов'язково включає визначення ролей СПО та текстовій опис послідовності їх дій (функцій). Функції СПО можна подати в ієрархічній формі у вигляді дерева функцій з його текстовим описом.

Опис функцій програми

Для того щоб визначити функції майбутнього програмного застосування необхідно, спираючись на проведене дослідження предметної області, дати відповідь на просте запитання: що буде робити майбутня програма з тих дій (функцій) СПО, що були визначені?

Якщо X дійсно є функцією програми,
то має сенс таке речення:
Програма повинна виконувати X.

До функцій, які виконує СПО можуть бути додані функції, що не мають місця в предметній області але необхідні для роботи програми. Наприклад, збереження/ завантаження певної інформації в/з файл(у), функції пошуку даних за різними ознаками і т.п.

Отже в результаті формулюються функції програми, які також подаються у вигляді ієрархії функцій (текстовий опис багаторівневим списком та графічно у вигляді дерева функцій).

Розробка концептуальної моделі предметної області

На концептуальній моделі предметної області відображаються:

- поняття (сутності) предметної області;
- атрибути сутності, тобто ті дані, яки описують їх властивості;
- зв'язки між сутностями.

Для того щоб визначитись, які сутності є в предметній області рекомендується скласти словник термінів, в який заносяться всі іменники, які використовувалися під час аналізу предметної області. Всі ці іменники записуються в табличку такого виду:

ХТО?	***ЩО?***

Після цього треба провести ретельний аналіз словника і визначитись, чи всі перелічені іменники будуть сутностями предметної області. Існує декілька правил:

1. *Якщо деяке поняття X в реальному світі описується одним числом або рядком символів, то, скоріш за все, це буде атрибут, а не сутність.*
2. *Якщо деяке поняття X є підмножиною іншої множини, але ця множина в іншому стані, то це не є окремою сутністю предметної області.*

Наприклад, поняття «студент» описується набором атрибутів (полів): ПІБ, № заліковки і т.п. А поняття «оцінка», скоріш за все, не буде окремою сутністю, тому що описується одним числом. Поняття «список студентів» та «список відмінників»: «список відмінників» по суті є підмножиною списком студентів, тому (згідно правила 2) «список відмінників» не буде окремою сутністю.

В пошуку та виборі атрибутів понять знов допоможе складений словник термінів.

Ті терміни зі словника, які не стали згідно правилу 1 поняттями концептуальної моделі будуть атрибутами понять.

До концептуальної моделі необхідно додати зв'язки між поняттями. Назви зв'язків формулюються у дієслівній формі ("працює", "належить", "включає" і т.п.). На кінцях лінії, що позначає зв'язок, можуть розташовуватися вирази, що визначають кількісний зв'язок між екземплярами понять. Під час вибору зв'язків слід запобігати використанню зв'язків, що повторюються або є надлишковими.

Концептуальна модель предметної області подається в звіті в UML-нотації.

Розробка прецедентів (варіантів використання)

В звіті до курсової роботи подається UML-діаграма прецедентів та опис сценарію кожного прецеденту.

На UML-діаграмі прецедентів відображаються ролі СПО (актори), прецеденти, зв'язки між прецедентами (включення, розширення, генералізація) та акторами і прецедентами.

Діаграма прецедентів будується на основі таких документів:

- **опис предметної області та функцій СПО**. Найчастіше як прецеденти розглядаються функції верхнього рівня «дерева функцій».

Прецедент (use case – варіант використання програми) – описує певну завершену послідовність дій СПО з використанням розроблюваної програми.

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:
прецедент починається, коли <Користувач (СПО)> <ініціює подію>.

На кожну подію користувача описується відгук системи – дії, що виконуються програмою як реакція на ці дії. В типовому ході подій описуються дії виконавця, що виконуються під гаслом: «Політ іде нормально!!!», 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 в [4]).

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

Опис прецеденту може складатися з підрозділів (див. [4]).

Розробка UML-діаграми класів

Діаграма класів (class diagram) відображає статичну структуру системи: класи, атрибути класів (поля та методи) та зв'язки між класами (асоціація, агрегація, композиція, узагальнення (генералізація)). Для зв’язків на діаграмі класів вказуються назва та напрямок зв’язку, кратність зв’язку (див. [4]).

Діаграма класів будується на основі таких документів:

♦ **концептуальна модель предметної області**. З її допомогою розробник може визначити основні програмні класи

та їх атрибути, що відповідають поняттям (сутностям) предметної області;

♦ **опис функцій програми («дерево функцій»)**. В добре спроектованій об'єктно-орієнтованій програмі всі функції повинні бути реалізовані класами програми. Отже всі функції, що увійшли в «дерево функцій» необхідно розподілити між сутностями концептуальної моделі. В результаті формуються класи програмних об'єктів. Під час розподілу обов'язків (функцій) між сутностями (класами) слід використовувати шаблони проектування Expert, Creator, Low Coupling, High Cohesion, Pure Fabrication (див. додаток В);

♦ **реальні (ідеальні) прецеденти**. З опису сценаріїв прецедентів розробник бере інформацію про те, виконанню яких функцій повинні задовольняти класи програми (можливо в результаті такого аналізу з'являться нові функції, що не були враховані в «дереві функцій»).

Опис системних операцій

Перш ніж розпочати розробку логіки роботи програми необхідно дослідити її поведінку як «чорну скриню». Однією зі складових такого дослідження є опис системних операцій.

Системна операція – це дія, виконання якої ініціюється користувачем, а обробка якої здійснюється програмою.

Системні операції визначаються для кожного прецеденту. При цьому розглядається, по-перше, типовий хід подій, а потім найбільш суттєві альтернативи. Для того щоб визначити системні операції необхідно:

1. зобразити програму як «чорну скриню», намалювавши для неї вертикальну пунктирну лінію (рис. 1);
2. ідентифікувати ролі кожного користувача (актора), що безпосередньо взаємодіє з програмою. Для кожної ролі також необхідно намалювати вертикальну лінію;
3. з опису типового ходу подій прецеденту з частини «Дії користувача» необхідно визначити події, які ініціюються кожним з акторів та відобразити їх горизонтальними

стрілками. Назви подій краще розпочинати з дієслова, що відображає сенс події: add…(додати), enter…(ввести), make…(зробити), end…(закінчити).

Події, що зображені на рис. 1 і є **системні операції**.

Опис системної операції (system operation contract) – це документ, що описує результати виконання операції. Він акцентує увагу на том, *що* має статися, а не на тому, *як* цього досягти.

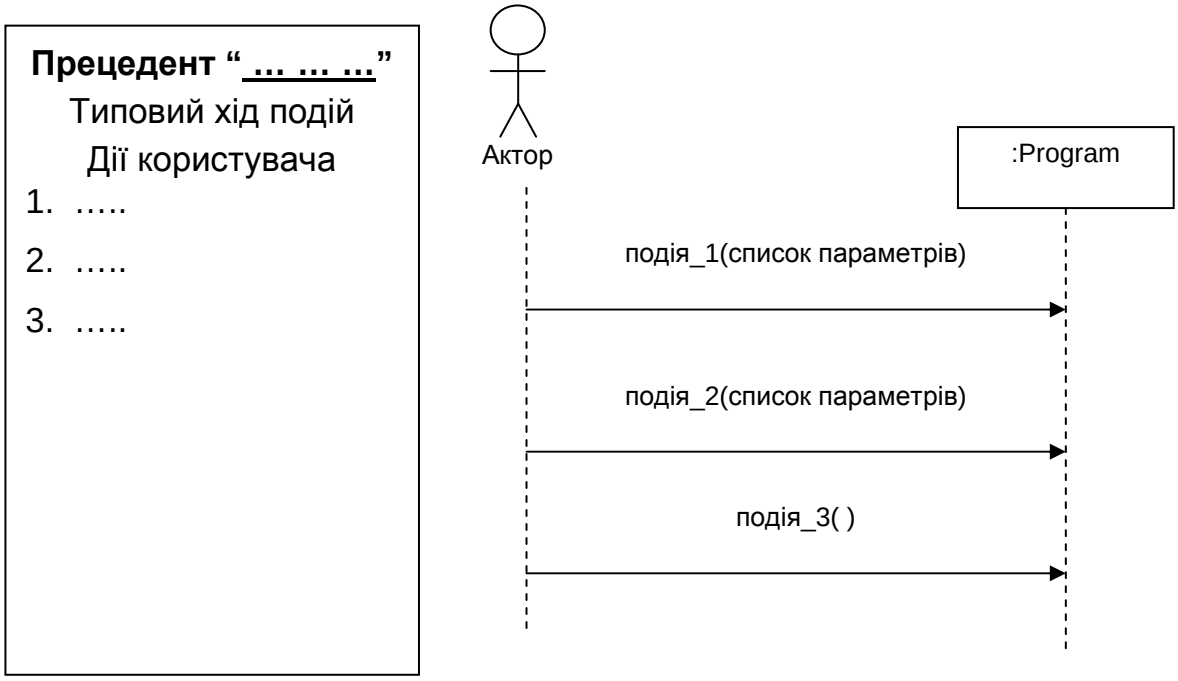

Рис. 1. Визначення системних операцій

Опис системної операції подається у вигляді таблиці:

Ім'я	Ім'я_операції (список параметрів)
Обов'язки	Короткий опис змісту (обов'язків) операції або її цілей
Тип	Системна
Посилання	Функції системи: посилання на номера функцій. Прецедент: назва прецеденту
Примітки	Певні пропозиції щодо алгоритму, конструкторських рішень, деталей і т. п. виконання операції
Винятки	Опис виняткових (виключних) ситуацій, що можуть виникнути під час виконання операції (наприклад, введені помилкові дані).

Вивід	Вивід інформації, що не стосується інтерфейсу користувача (наприклад, повідомлення або запити, що відправляється за межі системи).
Передумови	Опис стану системи, що необхідний для виконання операції. До цього відносяться: ♦ фактори, існування яких необхідно перевірити в програмі до початку виконання операції; ♦ фактори, від яких залежить успішне виконання операції, але їх неможна перевірити програмно. Такі фактори носять інформативний характер для майбутніх користувачів системи.
Постумови	Опис змін, що відбулися в системі після виконання операції; опис змін стану об'єктів концептуальної моделі. В описі можна використовувати такі категорії постумов: • створення/ видалення екземпляру; • модифікація атрибута екземпляру; • формування/ розрив зв'язків між екземплярами.

Опис передумов та постумов виконується в контексті концептуальної моделі:

- екземпляри яких об'єктів утворюються? – об'єктів, що присутні на концептуальній моделі;
- які зв'язки можуть формуватися? – зв'язки, що відображені на концептуальній моделі і т. д.

Дуже часто під час опису системних операцій виникає необхідність внесення змін до концептуальної моделі: нові поняття, нові атрибути, нові асоціації.

Для того щоб скласти опис системних операцій доцільно виконати такі дії:

1. визначити системні операції;
2. розпочати опис системної операції з опису розділу "Обов'язки", в якому неформально викладаються цілі операції;
3. заповнити розділ «Постумови». Постумови бажано описувати в декларативній формі з використанням дієслів минулого часу в завершеній формі пасивного залогу, щоб підкреслити

факт зміни стану, а не спосіб його реалізації. Наприклад, краще сказати “Створений екземпляр класу …”, а не “Створюється екземпляр класу …”.

Не забувайте встановлювати відношення між існуючими і створеними екземплярами класів (сутностей концептуальної моделі). Наприклад, при створенні запису про студента не достатньо тільки створити екземпляр відповідного класу «Студент», а необхідно додати цей запис до списку. Тому однією з постумов буде “Створений екземпляр об’єкту …, він зв’язаний з об’єктом …”.

Найбільш типовою помилкою при описі системних операцій є невключення *факту формування зв’язків між екземплярами* в постумови операції.

4. Після заповнення розділу “Постумови” заповнити розділ “Передумови”, записавши в нього всі умови, які необхідні для коректного виконання системної операції.

Розробка UML-діаграми послідовностей

Діаграми послідовностей відноситься до групи UML-діаграм взаємодії. Вона ілюструють процес обміну повідомленнями між екземплярами (класами) в часі (рис. 2).

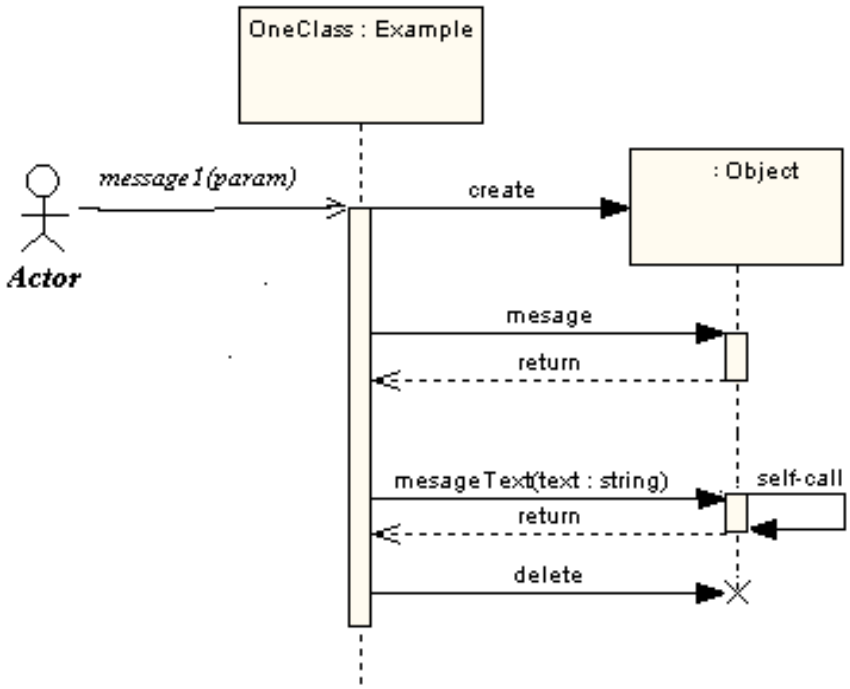

Рис. 2. Загальний вигляд UML-діаграми послідовностей

Діаграма послідовностей будується на основі таких документів:

♦ **опис системних операцій**. З їх допомогою розробник ідентифікує обов'язки та постумови, яким повинна задовольняти діаграма;

♦ **концептуальна модель та діаграма класів**. З їх допомогою розробник може визначити основні програмні класи, що відповідають поняттям (сутностям) предметної області. Екземпляри цих класів взаємодіють, що і відображається на діаграмі;

♦ **реальні (ідеальні) прецеденти**. З опису прецедентів розробник бере інформацію про те, виконанню яких задач повинні задовольняти діаграма послідовностей.

Діаграма послідовностей, фактично, є деталізацією опису системної операції. Якщо під час опису системної операції вказується, **що** змінюється в системі, то діаграма послідовностей відображає, **як** ці зміни здійснюються.

Для того щоб побудувати діаграму послідовностей доцільно виконати такі дії:

1. взяти до розгляду першу (другу, третю і т.д.) системну операцію;
2. відповісти на запитання:
    - який клас відповідає за обробку цієї операції?
    - який саме екземпляр цього класу?
3. зобразити на діаграмі послідовностей актора, стрілку системної операції, прямокутник екземпляру класу, його лінію життя, фокус управління для даної системної операції та зворотній зв'язок до актора;
4. відповісти на запитання:
    - чи будуть створюватися під час обробки даної системної операції нові екземпляри класів? яких класів?
    - чи необхідно буде під час обробки даної системної операції отримувати (передавати) інформацію іншим екземплярам класів? яких класів?

- чи необхідно буде під час обробки даної системної операції використовувати методи інших екземплярів класів? яких класів? які методи? і т.п.

5. зобразити на діаграмі прямокутники використовуваних (створюваних) екземплярів класів, їх лінії життя;
6. зобразіть стрілки методів, які використовуються, фокуси управління для цих методів та зворотній зв'язок;
7. визначите, чи необхідна перевірка певних умов під час виклику цих методів. Або виклик методів здійснюється в циклі. Додати до стрілок виклику методів відповідні позначки у разі необхідності.

Розробка UML-діаграми компонентів (опис структури програми)

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

На діаграмі компонентів відображаються компоненти (виконувані файли, файли вхідного коду, файли даних і т.п.), **стереотипи компонентів**, можливо, **класи**, які реалізує компонент, та зв'язки між компонентами (див. [4]).

В звіті до курсової роботи для кожного компонента наводиться текстовий опис його призначення. Для файлів вхідного коду наводиться опис основних класів, що реалізуються, або специфікації всіх підпрограм. Для файлів даних – опис структури інформації, що зберігається в файлі.

Перелік тем курсових робіт

1. «О спорт, ти – світ!!!»

Основні класи: член команди (ПІБ, фото, посада(амплуа), дата приходу/уходу в команду), список членів команди.

Основні функції: ведення списку членів команди, визначення складу команди на задану дату (рік), пошук членів команди за різними ознаками, ведення довідника посад членів команди (амплуа гравців).

2. Розробка програми ведення ділового щоденника

Основні класи: подія (назва, тип події, дата, час проведення, посилання на файл з описом події), список подій.

Основні функції: ведення списку подій, перевірка, чи не збігаються декілька подій в часі, пошук подій за різними ознаками, ведення довідника посад типів подій.

3. Розробка програми для ведення довідника про випускників університету

Основні класи: випускник (ПІБ, фото, факультет, спеціальність, № групи, рік вступу, рік закінчення), список випускників.

Основні функції: ведення списку випускників, визначення чисельності випускників за факультетами/спеціальностями, роками вступу/випуску, пошук випускників за різними ознаками, ведення довідника факультетів та спеціальностей.

4. Розробка програми формування сітки мовлення телевізійного каналу

Основні класи: програма (тип програми, час виходу в ефір тривалість), список програм.

Основні функції: ведення списку програм, формування сітки мовлення на день: призначення часу виходу в ефір програми, перевірка, щоби програми не накладались у часі, щоб не було «дірок», пошук програм за різними ознаками, ведення довідника видів програм.

5. Розробка програми для обліку пацієнтів ветеринарної клініки

Основні класи: пацієнт (кличка, вид тварини, вік, фото, посилання на файл з примітками), список пацієнтів.

Основні функції: ведення списку пацієнтів, пошук пацієнтів клініки за різними ознаками, ведення довідника видів тварин.

6. Арт-афіша

Основні класи: подія (назва, тип події, дата/час проведення, місце проведення, файл-афіша), список подій.

Основні функції: ведення списку подій, пошук подій за різними ознаками, ведення довідника типів подій (виставка, презентація, концерт, вистава і т.п.) та місць проведення.

7. «Мій прекрасний сад»

Основні класи: рослина (назва рослини, тип рослини, фото, посилання на файл з описом, температурний режим, режим поливу, режим освітлення, період цвітіння), список рослин.

Основні функції: ведення списку рослин, пошук рослини за різними ознаками, ведення довідника типів рослин, типів режимів поливу та освітлення.

8. «Мої кінофайли»

Основні класи: фільм (назва, жанр фільму, рік випуску, країна, режисер, список головних акторів, посилання на файл з фільмом, особистий рейтинг фільму), список фільмів.

Основні функції: ведення списку фільмів, пошук фільмів за різними ознаками, перегляд файлу з фільмом, ведення довідників жанрів фільмів, країн.

9. «Мій альбом»

Основні класи: фотографія (назва, посилання на файл зображення, дата/час створення, посилання на текстовий файл з описом), список фотографій.

Основні функції: ведення списку фотографій, перегляд альбому, пошук фотографій за різними ознаками.

10. «Навколо світу!»

Основні класи: день подорожі (номер дня, опис програми екскурсій (посилання на текстовий файл), список фото), список днів подорожі (як поле містить назву подорожі).

Основні функції: ведення списку днів, перегляд інформації по кожному дню подорожі, пошук за різними ознаками.

11. «Аукціон»

Основні класи: лот (назва, тип виробу, рік створення, автор, стартова/кінцева(якщо вже відома) ціна, посилання на файл з зображенням), список лотів.

Основні функції: ведення списку лотів, визначення загальної вартості лотів, що виставлені на аукціон, вартості проданих лотів, вартості за типами лотів, пошук лотів за різними ознаками, ведення довідника типів лотів.

12. «Музична скринька»

Основні класи: пісня (назва, група-виконавець, посилання на аудіо- або відео-файл, особистий рейтинг пісні (кількість прослуховувань)), список пісень.

Основні функції: ведення списку пісень, визначення топ-списку, пошук пісні за різними ознаками, ведення довідника виконавців.

13. «Наша група»

Основні класи: студент (ПІБ, фото (список фото), рік вступу, рік закінчення навчання (відрахування), посилання на файл з примітками), список студентів (як поле містить офіційну/неофіційну назву групи, список групових фото).

Основні функції: ведення списку студентів групи, визначення складу групи на заданий рік навчання, пошук студентів за різними ознаками.

14. «Моя родина»

Основні класи: родич (ПІБ, фото, дати життя).

Основні функції: побудова генеалогічного дерева, пошук родичів за різними ознаками, ведення довідника родинних ролей (мати, батько, брат, сестра і т.п.).

15. Кафе-кондитерська (2 частини)

Основні класи:

1) кондитерський виріб (назва, тип виробу, посилання на фото з зображенням, вага, кількість порцій, ціна, посилання на файл з описом, наприклад, інгредієнтів виробу), список виробів;

2) замовлення на вироби (дані замовника, список замовлених виробів з зазначенням кількості).

Основні функції:

1) ведення списку виробів, пошук виробів за різними ознаками, ведення довідника типів виробу;

2) ведення списку замовлень на вироби, визначення вартості замовлення та загальної вартості всіх замовлень.

16. Арт-кафе (2 частини)

Основні класи:

1) арт-програма (назва, тип програми, дата/час проведення, посилання на файл з афішею, список учасників (див. 2 частину)), список арт-програм;

2) учасник арт-програми (ПІБ/назва групи, амплуа учасника, фото, опис), список учасників.

Основні функції:

1) ведення списку арт-програм, пошук програм за різними ознаками, ведення довідника типів програм;

2) ведення списку учасників програм, пошук учасників за різними ознаками, ведення довідника амплуа учасників.

17. Книжкова полиця (2 частини)

Основні класи:

1) книжка (назва, автор, жанр, посилання на електронній файл, посилання на файл з зображенням, особистий рейтинг книги), список книг;

2) автор (ПІБ, країна, дати життя, стать, фото, посилання на файл з біографією), список авторів.

Основні функції:

1) ведення списку книг, визначення улюблених книг, пошук книги за різними ознаками, ведення довідника жанрів;

2) ведення списку авторів, визначення улюбленого автора, пошук авторів за різними ознаками, ведення довідника країн.

18. Розробка програми обліку товарів, що зберігаються на складах

Основні класи: товар (назва, од. виміру, кількість), список товарів.

Основні функції: ведення списку товарів, визначення загальної вартості товарів на складі, прийом/відпуск товарів, пошук товару за різними ознаками, ведення довідника одиниць виміру.

19. Розробка програми обліку ліків в аптеці

Основні класи: ліки (назва, вартість, тип ліків, кількість в наявності, кількість проданих за тиждень), список ліків.

Основні функції: ведення списку ліків, визначення інтенсивності споживання ліків та формування замовлення ліків на тиждень, пошук ліків за різними ознаками, ведення довідника типів ліків.

20. Розробка програми для контролю виконання студентами програми певної дисципліни (2 частини)

Основні класи:

1) студент (ПІБ, № заліковки), список студентів; список результатів успішності «студент-дисципліна».

2) дисципліна (назва, список форм контролю по семестрах (№ семестру, форми контролю (залік, іспит, курсова)), список дисциплін.

Основні функції:

1) ведення списку студентів та списку результатів успішності, визначення відмінників, двієчників і т.п., пошук студентів за різними ознаками;

2) ведення списку дисциплін, визначення форм контролю навчання по номеру семестру, пошук дисциплін за різними ознаками.

21. Розробка програми формування меню ресторану (3 частини)

Основні класи:

1) рецепт страви (назва, тип страви, список інгредієнтів (продуктів), опис рецепту приготування, вихідна кількість порцій, фото готової страви), список рецептів;

2) страва (назва страви, необхідна кількість порцій), список страв – меню;

3) продукт (назва, одиниці виміру, кількість), список продуктів, що є в наявності.

Основні функції:

1) ведення списку рецептів, пошук рецепту за різними ознаками, ведення довідника типу страв (перша страва, десерт і т.п.);

2) ведення списку страв - меню, визначення загальної кількості продуктів, що необхідні для приготування страв з меню, пошук страви за різними ознаками;

3) ведення списку продуктів, пошук продуктів за різними ознаками, ведення довідника одиниць виміру продуктів.

22. Розробка програми обліку зайнятості авто в агенції по прокату автомобілів (3 частини)

Основні класи:

1) авто (марка, тип, рік випуску, категорія, фото), список авто;

2) замовник (дані замовника, категорія), список замовників;

3) замовлення (замовник, авто, дата, час, тривалість замовлення), список замовлень.

Основні функції:

1) ведення списку авто, пошук авто за різними ознаками, ведення довідника типів авто (вантажні, легкові і т.п.) та категорій (VIP, бізнес-клас і т.п.);

2) ведення списку замовників, пошук замовників за різними ознаками, ведення довідника категорій замовників (постійні клієнти, VIP і т.п.);

3) ведення списку замовлень, визначення вартості замовлення, пошук замовлення за різними ознаками, ведення тарифів вартості години замовлення в залежності від категорії та типу авто.

23. Розробка програми продажу путівок туристичної агенції (2 частини)

Основні класи:

1) тур-агент (назва, країна, місто, список фото, категорія («зірочки»)), список тур-агентів;

2) путівка (тур-агент, дата, тривалість, кількість осіб), список путівок.

Основні функції:

1) ведення списку тур-агентів, пошук тур-агентів за різними ознаками, ведення довідників категорій, країн, міст;

2) ведення списку путівок, пошук путівки за різними ознаками.

Рекомендації щодо деяких аспектів програмної реалізації тем курсових робіт

Створення довідників

Створення довідників – потужний засіб запобігти зайвого повторного збереження інформації. Розглянемо приклад, ведеться список осіб: ПІБ особи, країна громадянства:

№	ПІБ особи	Країна
1	Іванов І.І.	Росія
2	Петренко П.П.	Україна
3	СамойловС.С.	Росія
4	Федорчук Ф.Ф.	Україна
5	Остапчук О.О.	Україна
6	…	…

Вже навіть в цьому невеличкому списку назви країн повторюються (назва країна – це рядок, найскоріше, його довжина буде біля 20 символів, а, відповідно, обсяг біля 20 байтів). Доцільно створити окремий список назв країн, а в списку осіб зберігати не назву країни, а її індекс зі списку країн:

№	ПІБ особи	Країна
1	Іванов І.І.	0
2	Петренко П.П.	1
3	СамойловС.С.	0
4	Федорчук Ф.Ф.	1
5	Остапчук О.О.	1
6	…	…

	Назва країни
0	Росія
1	Україна

Тепер назви країн зберігаються без повторень, а індекс країни в списку осіб займає набагато менше пам’яті. Крім того, під час пошуку осіб за країною набагато швидше порівнювати індекси країн ніж рядки-назви.

В курсових роботах довідники можна реалізовувати як прості списки рядків (наприклад, TStrings в Delphi) з можливістю збереження/завантаження даних.

Увага!!! Під час роботи з довідниками необхідно бути обережними з видаленням (заміною) даних всередині довідника тому, що це призведе до втрати достовірності інформації в основному списку. Наприклад, якщо видалити з довідника країн першу країну (при цьому Україна отримає індекс 0), в основному списку громадяни Росії «стануть» українцями, а українці взагалі втратять громадянство (країни з індексом 1 в довіднику вже не буде).

Створення полів-списків

В деяких темах курсових робіт в якості поля певного класу необхідно формувати списки (тема 8 – список головних акторів, теми 10, 13, 23 – список фото,). Такі списки також доцільно реалізовувати як списки рядків.

Перегляд файлів із зображенням, відеофайлів, прослуховування аудіофайлів

Для перегляду файлів з графічними зображеннями у разі розробки програми в середовищах Delphi або C++Builder доцільно використовувати компонент TImage з вкладки Additional.

Інформація про зображення, що містить цей компонент, зберігається у властивості Picture. Властивість-об’єкт Picture має методи файлового читання і запису LoadFromFile и SaveToFile. Для того щоб, наприклад, завантажити до TImage фото-файл, можна написати такий код:

```
Image1.Picture.LoadFromFile(‘myfoto.bmp’);
```

Якщо property Stretch: Boolean встановити в True, то зображення «натягується» на робочу область TImage, за необхідністю зменшення або збільшення свого розміру.

Перегляд/прослуховування медіа-файлів реалізується за допомогою компонента TMediaPlayer вкладки System. Основні властивості компонента TMediaPlayer:

- Name — Ім’я компонента
- DeviceType — тип пристрою. Визначає конкретний пристрій, яким є MediaPiayer: dtAutoSelect – тип пристрою визначається автоматично; dtVaweAudio – програвач звуку; dtAVIvideo – відеопрогравач; dtCDAudio – CD-програвач
- FileName — Імя файлу, в якому знаходиться звуковий фрагмент або відеоролик
- AutoOpen — Признак автоматичного відкриття одразу після запуску програми файлу для програвання
- Display — Визначає компонент, на поверхні якого відтворюється відеоролик (зазвичай як екран використовується компонент Panel)
- VisibleButtons — Складена властивість, що визначає видимі кнопки компонента

Докладну інформацію про роботу з цими компонентами дивися в [1; 2].

Список літератури

1. *Архангельский А.Я.* Программирование в C+Builder 5. – М.: Бином, 2005. – 1152 с.
2. *Дарахвелидзе П.Г., Марков Е.П*. Программирование в Delphi 7. – СПб.:БХВ-Петербург, 2003. – 784 с.
3. *Об’єктно-орієнтоване* програмування: методичні вказівки до виконання індивідуального завдання / Г.В. Красовська. – К.: КНУБА, 2011. – 24 с.
4. *Об’єктно-орієнтоване* програмування: методичні вказівки до виконання лабораторних робіт / Г.В. Красовська. – К.: КНУБА, 2009. – 29 с.
5. *Ларман, Крег*. Применение UML и шаблонов проектирования: учебн. пос /пер. с англ. – М.: Издат. дом “Вильямс”, 2001. – 496 с.
6. *Леоненков А.* Самоучитель UML. – СПб: BHV Санкт-Петербург, 2001. – 304 с.
7. *Шилдт Г.* Самоучитель C++ / пер. с англ. – СПб: БХВ-Петербург, 2003. – 687 с.
8. *Хаймен М., Арнсон Б*. Visual C++ .NET для "чайников" / пер. с англ. – М.: Издательский дом "Вильяме", 2002. – 288 с.

Додаток А

Титульний лист до курсової роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Кафедра інформаційних технологій

КУРСОВА РОБОТА
З ДИСЦИПЛІНИ
"Об'єктно-орієнтоване програмування"

за темою_____

Виконав (ла) студент(ка) II курсу факультету АІТ
спеціальності ІУСТ

_____
(прізвище, ім'я, по батькові)

Керівник роботи _____(прізвище, ініціали)

Київ - рік

Додаток Б

Бланк завдання до курсової роботи

Київський національний університет будівництва і архітектури

Кафедра _____**інформаційних технологій**____
Дисципліна ____**»Об'єктно-орієнтоване програмування»**____
Спеціальність ____**ІУСТ**____
Курс_____**2**_____Група_____Семестр___**4**____

ЗАВДАННЯ
на курсову роботу студента(ки)

(прізвище, ім'я, по батькові)

1. Тема роботи

2. Загальні вимоги до роботи

3. Дата видачі завдання

4. Дата захисту курсової роботи

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу розробки курсової роботи	Термін виконання етапу	Дата консультації/ відмітка про виконання

Студент _____ / _____ /
(підпис) (прізвище, ім'я, по батькові)

Керівник _____ / _____ /
(підпис) (прізвище, ім'я, по батькові)

"_____"_____20__р.

Додаток В

Шаблони проектування GRASP

Шаблон	**Опис**
Expert	Який клас зазвичай повинен відповідати за виконання обов'язків? Обов'язок призначається інформаційному **експерту** – класу, що має найбільшу кількість інформації для виконання цього обов'язку.
Creator	Хто повинен відповідати за створення класу? Класу B призначається обов'язок створювати клас A, якщо виконуються одна з умов: ♦ клас B містить об'єкти класу A; ♦ клас B агрегує (складається з) об'єкти A; ♦ клас B має дані для ініціалізації об'єкту A; ♦ клас B записує екземпляри об'єкту A; ♦ клас B активно використовує об'єкти A.
Low Coupling (оцінювальний)	Як забезпечити низьку залежність класів та підвищити можливість повторного використання? Обов'язки розподіляються таким чином, щоб ступень пов'язаності була найменшою. Ступень пов'язаності показує наскільки один клас жорстко пов'язаний з іншим класом або певним набором даних іншого класу. Наприклад, якщо один клас в своїй роботі використовує значення певного поля іншого класу, то зміна в типі цього поля (або назві і т.п.) призводить до внесення змін до першого класу.
High Cohesion (оцінювальний)	Як забезпечити можливість управління складністю? Обов'язки розподіляються таким чином, щоб ступень зачеплення була найбільшою. Вважається, що клас має високий ступень зачеплення, якщо його обов'язки щільно пов'язані між собою та він не виконує робіт надмірного об'єму.
Pure Fabrication	Якщо розробник близький до відчаю, хто забезпечить реалізацію шаблонів Low Coupling та High Cohesion. Створіть штучний клас, що не являє конкретне поняття предметної області, тобто синтезується штучна сутність для підтримки високого зачеплення, слабкого пов'язання та повторного використання.

Навчально-методичне видання

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні вказівки
до виконання курсової роботи
для студентів, які навчаються за напрямом
підготовки 6.050101 «Комп’ютерні науки»
спеціалізації «Інформаційні
управляючі системи та технології»

Укладач КРАСОВСЬКА Ганна Валеріївна

Комп’ютерне верстання *Т.І. Кукарєвої*

Підписано до друку 2011. Формат $60 \times 84\ {}_{1/16}$
Ум. друк. арк.1,63. Обл.-вид. арк. 1,75.
Тираж 40 прим. Вид. № 60/III-11. 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.