МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні рекомендації до виконання курсового проєкту

для здобувачів вищої освіти спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2024

УДК 004.4(072.034)
О-29

Укладачі: С. Г. Удовенко
О. О. Тютюник
О. В. Гороховатський
Ю. Е. Парфьонов

Затверджено на засіданні кафедри інформатики та комп'ютерної техніки.
Протокол № 1 від 29.08.2023 р.

Самостійне електронне текстове мережеве видання

Об'єктно-орієнтоване програмування [Електронний ресурс] : О-29 методичні рекомендації до виконання курсового проєкту для здобувачів вищої освіти спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня / уклад. С. Г. Удовенко, О. О. Тютюник, О. В. Гороховатський, Ю. Е. Парфьонов. – Харків : ХНЕУ ім. С. Кузнеця, 2024. – 43 с.

Подано мету та завдання виконання курсового проєкту з об'єктно-орієнтованого програмування. Наведено зміст і оформлення текстової частини курсового проєкту, порядок організації захисту та критерії оцінювання курсового проєкту, професійні компетентності, якими має володіти здобувач вищої освіти після виконання курсового проєкту.

Рекомендовано для здобувачів вищої освіти спеціальності 126 "Інформаційні системи та технології".

УДК 004.4(072.034)

Вступ

Сучасні умови господарювання вимагають від фахівців з управління та бізнесу всебічного використання новітніх інформаційних технологій (ІТ) для досягнення конкурентних переваг і поліпшення якості управління.

Застосування сучасних ІТ-рішень допомагає підвищити рівень обґрунтованості й точності управлінських рішень, знизити витрати та підвищити ефективність господарювання, що є критичним у конкурентному бізнес-середовищі.

Проте розроблення та впровадження сучасних ІТ-рішень неможливо уявити без відповідного програмного забезпечення. Важливість розроблення програмного забезпечення в сучасному світі також зумовлюють такі фактори: зростання складності та функціональності комп'ютерної техніки, необхідність створення програм, які можуть працювати на різних платформах та інтегрувати їхню функціональність, потреба в розробленні програм для оброблення та аналізу великих масивів даних, забезпечення безпеки інформації в автоматизованих інформаційних системах.

Для створення відповідних програмних систем фахівці мають володіти як концепціями алгоритмізації та програмування, так і різними парадигмами програмування, зокрема, структурною, функціональною та об'єктно-орієнтованою. Це дозволяє їм розробляти програмні системи з використанням широкого кола мов програмування, оскільки більшість сучасних мов програмування тією чи іншою мірою допускають затосування різних парадигм.

Здобувачі вищої освіти спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня у складі обов'язкових професійних освітніх компонентів вивчають "Програмування", "Основи алгоритмізації" та "Об'єктно-орієнтоване програмування". У процесі вивчення відповідних освітніх компонентів здобувачі вищої освіти набувають практичних навичок роботи з технічною літературою, сучасними мовами, технологіями та середовищами програмування. Виконання курсового проєкту з об'єктно-орієнтованого програмування закріплює набуті практичні навички.

Курсовий проєкт з об'єктно-орієнтованого програмування спрямовано на більш глибоке осмислення й закріплення теоретичних знань,

здобутих під час вивчення зазначених освітніх компонентів, і на вдосконалення практичних навичок щодо розроблення програмного забезпечення та необхідної документації.

У методичних рекомендаціях до виконання курсового проєкту з об'єктно-орієнтованого програмування наведено зміст і правила оформлення курсового проєкту, етапи виконання, орієнтовну тематику проєктів (додаток А).

1. Мета і завдання виконання курсового проєкту з об'єктно-орієнтованого програмування

Метою написання курсового проєкту з об'єктно-орієнтованого програмування є закріплення теоретичних знань, здобутих на лекційних і лабораторних заняттях освітніх компонентів "Програмування", "Основи алгоритмізації", "Об'єктно-орієнтоване програмування" спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня.

Завданнями виконання курсового проєкту з об'єктно-орієнтованого програмування є поглиблення набутих навичок щодо практичного застосування знань під час створення комплексного застосунку з використанням сучасних інструментальних засобів розроблення. Водночас здобувач вищої освіти має показати вміння користуватися спеціальною літературою, державними стандартами, довідниками та іншими матеріалами з інформаційних технологій.

У результаті виконання курсового проєкту здобувач вищої освіти набуває такі загальні та спеціальні компетентності й забезпечує програмні результати навчання (табл. 1).

Таблиця 1

Компетентності та результати навчання

Компетентності	Результати навчання
К31, К32, КС4	ПР2
К33, К38, КС4	ПР3

Примітка.

К31. Здатність до абстрактного мислення, аналізу та синтезу.

К32. Здатність застосовувати знання у практичних ситуаціях.

КЗ3. Здатність до розуміння предметної області та професійної діяльності.

КЗ8. Здатність оцінювати та забезпечувати якість виконуваних робіт.

КС4. Здатність проектувати, розробляти та використовувати засоби реалізації інформаційних систем, технологій та інфокомунікацій (методичні, інформаційні, алгоритмічні, технічні, програмні та інші).

ПР2. Застосовувати знання фундаментальних і природничих наук, системного аналізу та технологій моделювання, стандартних алгоритмів та дискретного аналізу при розв'язанні задач проектування і використання інформаційних систем та технологій.

ПР3. Використовувати базові знання інформатики й сучасних інформаційних систем та технологій, навички програмування, технології безпечної роботи в комп'ютерних мережах, методи створення баз даних та інтернет-ресурсів, технології розроблення алгоритмів і комп'ютерних програм мовами високого рівня із застосуванням об'єктно-орієнтованого програмування для розв'язання задач проектування і використання інформаційних систем та технологій.

Робота над курсовим проєктом певною мірою визначає загальнотеоретичну та спеціальну підготовку здобувача вищої освіти і в остаточному підсумку готує його до майбутнього виконання більш складного й завершального етапу навчального процесу – дипломного проєктування.

2. Зміст і оформлення курсового проєкту

Зміст курсового проєкту з об'єктно-орієнтованого програмування

Зміст має відповідати типовому змісту курсових проєктів з об'єктно-орієнтованого програмування, запропонованому в методичних рекомендаціях, за винятком розроблення індивідуальних тем курсових проєктів. Увесь матеріал курсового проєкту має бути цілеспрямованим, викладеним у логічній послідовності і взаємопов'язаним з окремими його розділами.

Курсовий проєкт містить такі складники:

титульний аркуш;

зміст, де зазначено сторінки розташування розділів проєкту;

розділи проєкту відповідно до рекомендацій, наведених далі;

список використаних джерел;

додатки.

Титульний аркуш

Першою сторінкою курсового проєкту є **титульний аркуш**. Він містить такі дані:

відомості про виконавця роботи;

повну назву документа;

підписи відповідальних осіб, зокрема керівника роботи;

назву міста і рік виконання.

Приклад титульного аркуша курсового проєкту з об'єктно-орієнтованого програмування наведено в додатку Б.

Зміст

У **змісті** зазначають: вступ; назви всіх розділів, підрозділів та пунктів основної частини курсового проєкту; висновки; список використаних джерел; додатки і номери сторінок, які містять початок матеріалу (додаток В).

Перелік умовних скорочень

Якщо в курсовому проєкті використовують маловідомі скорочення, нові символи, позначення тощо, то в ній має бути **перелік умовних скорочень**, який подають у вигляді окремого списку, який розміщують перед вступом. Незважаючи на це, за першої появи цих елементів у тексті документа надають їх розшифрування.

Якщо в роботі спеціальні терміни, скорочення, символи, позначення повторюють менш ніж три рази, то перелік не складають, а їх розшифрування наводять у тексті під час першого згадування.

Приклад переліку умовних скорочень подано в роботі [3].

Вступ

Вступ – це відображення роботи, тому слід ретельно його опрацювати. Краще формувати вступ після виконання основного тексту курсового проєкту.

У вступі потрібно зазначити важливість використання інформаційних систем у сучасних умовах, роль інформаційних систем у предметній галузі, що розглядають у курсовому проєкті, роль засобів збереження даних в інформаційних системах, мету та завдання курсового проєкту, відомості щодо розробленої програми (призначення, яке вона дозволяє

автоматизувати; технології та мова програмування, використані під час розроблення програми). Також слід навести інформацію щодо можливих галузей застосування результатів, отриманих у курсовому проєкті.

Розділ 1. Специфікація проєкту

Метою розділу 1 є розроблення постановки завдання, вимог до програмного забезпечення та математичного опису задачі.

У підрозділі 1.1 "Постановка завдання" наводять постановку завдання на розроблення програмного продукту відповідно до варіанта, виданого здобувачу вищої освіти. Також потрібно вказати, що розроблення виконано на підставі завдання на виконання курсового проєкту, затвердженого кафедрою інформатики та комп'ютерної техніки (додаток Г).

У підрозділі 1.2 "Вимоги до програмного забезпечення" містяться функціональні та нефункціональні вимоги до програмного продукту. Функціональні вимоги описують, що має робити програмний продукт і які перетворення вхідних даних виконувати.

Нефункціональні вимоги визначають властивості програмного продукту, прямо не пов'язані з його функціональністю. Прикладом таких властивостей може бути максимальний час відгуку програми на запит користувача, мінімальний час безперебійної роботи системи, наявність зручного інтерфейсу користувача тощо.

Функціональні та нефункціональні вимоги до програмної системи, розроблюваної відповідно до будь-якої теми курсового проєкту з переліку рекомендованих тем, які мають бути реалізованими у курсовому проєкті.

Функціональні вимоги

1. Створення файлу бази даних і запис до нього даних у певному форматі.
2. Читання всіх даних із файлу та їх відображення.
3. Додавання нового елемента даних до файлу бази.
4. Оновлення будь-якого елемента даних у файлі бази.
5. Видалення будь-якого елемента даних із файлу.
6. Отримання та відображення підсумкової інформації.
7. Перевірка допустимості основних даних, що вводить користувач.
8. Видача користувачу попереджувальних та інформаційних повідомлень.

Нефункціональні вимоги

Мінімальні вимоги до консольного інтерфейсу користувача:

1. Для забезпечення діалогу користувача з програмним продуктом елементи інтерфейсу користувача повинні мати докладне головне меню, яке містить посилання на основні завдання, виконувані в проєкті.

2. Кожен пункт головного меню може мати меню другого рівня, яке докладно розкриває його зміст.

3. Відображення даних і результатів обчислень слід формувати в наочному вигляді з необхідними коментарями.

4. Дані, що зберігаються у файловій базі даних, а також результати запитів до неї, потрібно відображати в табличному вигляді без зміщень у рядках і стовпцях.

Мінімальні вимоги до графічного інтерфейсу користувача:

1. Елементи інтерфейсу користувача повинні супроводжуватися допоміжними текстовими мітками або мати заголовки, виконані українською мовою.

2. Головне вікно застосунку – фрейм, який має панель меню з підтримкою "акселераторів", користувальницьку піктограму системного меню, панель інструментів із підтримкою спливаючих "підказок" для кнопок.

3. Дані, що зберігаються у файловій базі даних, повинні відображатися в табличному вигляді.

4. Наявність діалогових вікон, за допомогою яких користувач може додавати або редагувати дані.

5. Наявність стандартних діалогових вікон відкриття та збереження файлу, за допомогою яких користувач має можливість виконувати відповідні дії.

6. Наявність стандартних діалогових вікон для відображення попереджувальних та інформаційних повідомлень, що можуть з'являтися в ході виконання програми.

7. Наявність діалогового вікна "Про програму", за допомогою якого користувач може переглянути інформацію про розроблювача програми, зокрема його (її) фотографію.

8. Усі діалогові вікна повинні бути модальними та мати фіксований розмір. Під час розроблення інтерфейсу користувача рекомендовано керуватися методичними рекомендаціями, наведеними в додатку Д.

Вимоги до архітектури програми:

1. Використання не менш ніж одної структури даних із стандартної бібліотеки колекцій.

2. Використання файлових потоків уведення-виведення даних.

3. Використання механізму винятків для оброблення помилок.

Вимоги до вихідного коду застосунку:

1. Додержання принципу інкапсуляції щодо рівнів доступу до полів та методів класів.

2. Вихідний код кожного з класів програми має міститися в окремому файлі.

3. Наявність документаційних коментарів (для класів-призначення класу; для методів – призначення методу, опис параметрів та значення, що повертається) з обов'язковим використанням відповідних документаційних тегів.

4. Виконання угод щодо запису тексту програм певною мовою програмування.

За погодженням з керівником курсового проєкту деякі із цих вимог можна змінити.

Якщо здобувач вищої освіти вибрав тему курсового проєкту самостійно, то вимоги до програми обговорюють із керівником, і вони можуть відрізнятися від наведених вище.

У підрозділі 1.3 "Математичний опис задачі" наводять математичну формалізацію задачі, тобто наявні співвідношення між величинами. Водночас, залежно від специфіки розв'язуваної задачі, можна використати різні розділи математики та інших дисциплін. Таким чином формують математичну модель явища з певною точністю, припущеннями й обмеженнями.

Математична модель повинна відповідати принаймні двом вимогам: реалістичності та реалізованості.

Під реалістичністю розуміють правильне відображення моделлю найбільш істотних рис досліджуваного явища.

Реалізованість досягається розумною абстракцією, відволіканням від другорядних деталей, щоб звести задачу до задачі з відомим рішенням. Умовою реалізованості є можливість практичного виконання необхідних обчислень за відведений час із використанням обмежених ресурсів.

У цьому підрозділі потрібно навести математичні формули або логічні співвідношення, які виражають залежність показників, що розраховують, від вхідних даних, і необхідні пояснення.

Розділ 2. Програмна реалізація

Метою розділу 2 є опис архітектури розробленої програмної системи, відомості про тестування програмної системи та розгортання програмного продукту, а також керівництво користувача.

Підрозділ 2.1 "Архітектура програмної системи". Програмна система означає програмне забезпечення, що розроблюють. Це може бути велика колекція з безлічі компонентів програмного забезпечення, один застосунок або частина застосунку.

Для подання статичної структури моделі програмної системи призначено UML-діаграму класів. Вона може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин. У цьому підрозділі слід навести:

1) UML-діаграму класів, що реалізують основну бізнес-логіку програмної системи, та її опис. Приклад опису UML-діаграми класів наведено в додатку Е;

2) посилання на лістинг програми з вихідним кодом, що відповідає класам, які реалізують основну бізнес-логіку програмної системи. Лістинг програми має бути в одному з додатків до курсового проєкту.

Підрозділ 2.2 "Тестування програмної системи". Тестування програмної системи – процес виконання програмного коду, спрямований на виявлення наявних у ньому дефектів.

Під *дефектом* розуміють ділянку програмного коду, виконання якої за певних умов призводить до несподіваного поводження системи (тобто поводження, що не відповідає вимогам).

Завдання тестування – визначення умов, за яких виявляються дефекти системи, і протоколювання цих умов.

Мета застосування процедури тестування програмного коду – мінімізація кількості дефектів у кінцевому продукті.

Види тестування

Модульне тестування

У ході модульного тестування кожний модуль тестують як на відповідність вимогам, так і на відсутність проблемних ділянок програмного коду, які можуть викликати відмови та збої в роботі системи.

Інтеграційне тестування

Окремі модулі рідко функціонують самі по собі, тому наступне завдання після тестування окремих модулів – тестування коректності взаємодії декількох модулів, об'єднаних у єдине ціле. Таке тестування називають інтеграційним.

Системне тестування

Після завершення інтеграційного тестування всі модулі системи є погодженими за інтерфейсами і функціональністю. Починаючи із цього моменту, можна переходити до системного тестування, тобто тестування поводження системи загалом як єдиного об'єкта. Вхідною інформацією для проведення системного тестування є два класи вимог: функціональні й нефункціональні.

Системне тестування проводять у кілька етапів, на кожному з яких використовують один з видів системного тестування.

Важливим видом системного тестування є функціональне тестування, призначене для підтвердження того, що вся система загалом поводиться відповідно до очікувань користувача, формалізованих у вигляді системних вимог.

У ході функціонального тестування перевіряються усі функції системи з погляду її користувачів (як людей, так і інших програмних систем). Також потрібно перевірити функціональну повноту користувальницького інтерфейсу й коректність виведення інформації.

Документування процедури тестування. Основне призначення документації, створюваної під час тестування, – забезпечення гарантій того, що процес тестування виконано з необхідною якістю й усі аспекти поводження системи протестовано.

Перелік необхідної документації:

1. Тест-вимоги.
2. Тест-план.
3. Звіт про тестування.

Тест-вимоги розробляють на підставі системних і функціональних вимог до застосунку. У них докладно описують, які аспекти поводження системи слід протестувати, щоб упевнитися в її коректному функціонуванні, і на підставі якого зовнішнього ефекту можна переконатися, що функціональність, яка перевіряли, реалізовано правильно.

Тест-вимоги мають бути достатніми для побудови тест-плану перевірки програмної системи без ознайомлення з її програмним кодом.

Структура тест-вимог має додержуватися структури функціональних вимог до системи. Як правило, одній системній або функціональній вимозі відповідає мінімум одна тест-вимога.

Для кожної тест-вимоги має бути можливість перевірки – виконується ця вимога в реалізованій системі чи ні. На підставі тест-вимог створюють тест-план – документ, що містить докладний покроковий опис того, як потрібно протестувати тест-вимоги.

На відміну від тест-вимог, у тест-плані описуються конкретні способи перевірки функціональності системи. Зазвичай, тест-план складається з окремих тестових прикладів, кожний із яких перевіряє певну функцію або набір функцій системи

Для кожного тестового прикладу однозначно визначають критерій успішного проходження, за допомогою якого можна робити висновок про відповідність поводження системи заданим вимогам.

Структура тест-плану має відповідати структурі тест-вимог.

Кожний пункт тест-плану повинен містити:

1. Посилання на вимогу(и), яку перевіряють цим пунктом.

2. Конкретне значення вхідних даних.

3. Очікувану реакцію програми (тексти повідомлень, значення результатів).

4. Опис послідовності дій, необхідних для виконання пунктів тест-плану.

За результатами виконання тестів створюють звіт про виконання тестування. Він є основним джерелом для висновку про ступінь відповідності протестованої системи вимогам. Такий звіт як мінімум має містити інформацію про кожний виконаний тестовий приклад і результат його виконання (успіх або невдачу).

Іноді тест-план сполучають зі звітом про проведення тестування, додаючи до нього інформацію про отриману реакцію системи та збіг (розбіжності) отриманих результатів з очікуваними.

Наприкінці опису кожного тестового прикладу додають інформацію про те, чи пройдено тестовий приклад у цілому. Наприкінці всього тест-плану, сполученого зі звітом, міститься графа "Тестових прикладів пройдено/усього", у яку заносять число пройдених тестових прикладів і їхню загальну кількість.

Приклад тест-плану, сполученого зі звітом про проведення тестування, наведено в додатку Ж.

У цьому пункті подають опис процедур функціонального тестування та їхніх результатів.

Для опису процедури тестування складають такі документи:

1. Тест-вимоги.

2. Тест-плани, сполучені зі звітами про проведення тестування.

У звіті про проведення тестування вказують як позитивні, так і негативні результати виконання окремих тестів. Однак загальний результат тестування застосунку має бути позитивним, адже інакше він не відповідає певним вимогам. Для цього в разі потреби проводять додаткові заходи щодо виправлення помилок і тестування. Їх також слід описати в цьому пункті.

Розгортання – це процес поширення готового застосунку або компонента для установки на інші комп'ютери.

У підрозділі 2.3 "Розгортання програмного продукту" потрібно навести опис вимог до апаратних і програмних засобів, необхідних для функціонування розробленого програмного продукту, та дій щодо його інсталяції на комп'ютері користувача. Приклад опису процедури розгортання програмного продукту, створеного на платформі JavaSE наведено в додатку И.

Підрозділ 2.4 "Керівництво користувача" містить призначення програми, інструкцію щодо її використання за призначенням, опис повідомлень користувачу, що можуть з'явитися в процесі роботи програми.

У пункті 2.4.1 "Призначення програмного продукту" вказують відомості про його призначення та інформацію, достатню для розуміння функцій програмного продукту.

У пункті 2.4.2 "Використання програмного продукту" має бути вказано послідовність дій користувача, яка забезпечує запуск, виконання та завершення програми, наведено опис функцій, формату та можливих варіантів дій, за допомогою яких користувач керує виконанням програми, а також реакцію програми на ці дії.

У пункті 2.4.3 "Повідомлення користувачеві" наводять екранні форми повідомлень, що можуть з'являтися в ході виконання програми, та опис їхнього змісту.

У **висновках** наводять оцінку одержаних результатів роботи (негативних також); можливі галузі використання результатів роботи; економічну, наукову, соціальну значущість роботи.

Список використаних джерел – це перелік джерел інформації, які було цитовано, згадано або розглянуто в роботі. Список використаних джерел оформлюють згідно з ДСТУ 8302:2015 "Бібліографічне посилання. Загальні вимоги та правила складання" [1].

Список використаних джерел подають мовою оригіналу, розміщують в алфавітному порядку прізвищ перших авторів або назв, нумерують у порядку їх зростання. Нумерація безперервна. Роботи одного автора розташовують за алфавітом назв або в хронології їх написання.

Алфавітний список складають у такій послідовності:

література українською мовою і мовами з кириличною графікою (болгарська та ін.);

література мовами з латинською графікою;

електронні ресурси в тій самій послідовності, що й друковані видання (спочатку кирилицею, а потім латиницею).

Список літературних джерел обов'язково має містити прізвище та ініціали автора, повну назву джерела, назву міста видання, назву видавництва та рік видання, кількість сторінок чи посилання на сторінки тощо. Загальний обсяг книги в сторінках указують, якщо посилання на неї дають повністю; сторінки (від... до) зазначають, якщо посилання належать до окремої частини літературного джерела.

Вимоги до оформлення списку використаних джерел наведено в роботі [1].

У **додатках** вміщують матеріал, який є необхідним для повноти курсового проєкту, але не може бути послідовно розміщений у його основній частині через великий обсяг або способи відтворення та з інших причин. Ілюстрації (діаграми бізнес-процесів, схеми алгоритмів, технологічних процесів, сценарії діалогів та ін.), таблиці, проміжні математичні докази, формули й розрахунки, текст допоміжного характеру та інші матеріали можна оформити у вигляді додатків.

Кожний додаток починається з нової сторінки і має заголовок (назву), надрукований малими літерами з першої великої. Додатки позначають великими літерами українського алфавіту, починаючи з літери А, за винятком літер Ґ, Є, З, І, Ї, Й, О, Ч, Ь. Якщо додатків багато, їх оформлюють у вигляді окремої частини, на титульному аркуші якої великими літерами друкують слово "ДОДАТКИ".

Оформлення текстової частини курсового проєкту з об'єктно-орієнтованого програмування

Курсовий проєкт виконують із додаванням обов'язкових матеріалів (схеми, діаграми, графіки залежностей, таблиці, рисунки, лістинги програм тощо), що розробляють відповідно до завдання.

Обсяг курсового проєкту – 20 – 40 друкованих сторінок формату А4 (без додатків).

Курсовий проєкт друкують на аркушах формату А4. Текст виконують з використанням шрифту Times New Roman (кегль 14), з міжрядковим інтервалом 1,5.

Поля сторінок такі: – 25 мм від лівого краю аркуша, 10 мм – від правого краю аркуша, по 20 мм – від верхнього та нижнього країв аркуша.

Абзацний відступ має бути однаковим по всій роботі та дорівнювати 1,25 см.

Вирівнювання основного тексту проводять по ширині.

Загальними вимогами до тексту курсового проєкту є логічна послідовність викладення матеріалу, чіткість і конкретність викладення теоретичних і практичних результатів роботи, сутності постановки завдання та мети роботи, методів дослідження, прийнятих рішень, доведеність висновків і обґрунтованість рекомендацій. У тексті курсового проєкту потрібно дотримуватися єдиної термінології. Робота не має бути перевантаженою малоінформативним матеріалом, описом загальновідомих даних, виведенням формул тощо. Слід посилатися на джерела інформації.

Текст викладають у формі першої множини (наприклад: "вивчаємо", "розглядаємо", "з'ясуємо", "доведемо"), третьої особи множини (наприклад: "визначають", "обчислюють", "уважають"), у безособовій формі (наприклад: "подано", "наведено", "вивчено").

Під час викладення матеріалу не слід використовувати:

розмовні звороти;

жаргонні слова та звороти;

різні терміни для позначення одного поняття;

іншомовні слова та терміни за наявності в українській мові рівнозначних слів і термінів;

скорочення слів і словосполучень, крім установлених правилами орфографії та нормативними документами.

Потрібно уникати повторень, надмірного цитування. Цитати мають бути короткими й органічно пов'язаними з основним текстом. На використані в роботі цифрові дані, цитати, таблиці мають бути посилання із зазначенням порядкового номера відповідного джерела в списку літератури. У тексті цей порядковий номер беруть у квадратні дужки наприклад: "*…подані в роботі [8] результати використано в курсовому проєкті для…*".

Оформлення ілюстрацій, таблиць, формул

Ілюстрації слід розміщувати безпосередньо після тексту, де їх згадують уперше, або на наступній сторінці. На всі ілюстрації дають посилання. Ілюстрації можуть мати назву, яку розміщують під ілюстрацією. Ілюстрацію позначають словом "Рисунок", яке разом із номером і назвою ілюстрації розміщують після пояснювальних даних.

Ілюстрації слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу.

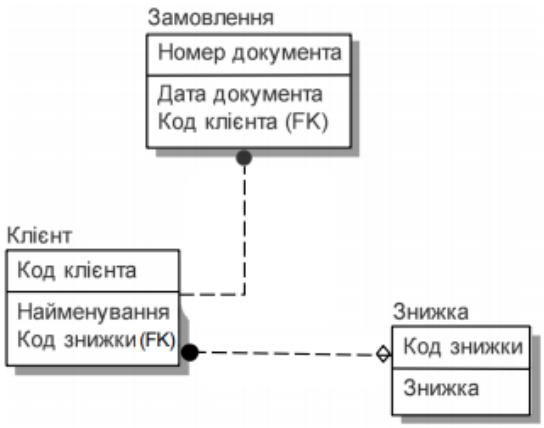

Рисунок 1 – Приклад оформлення рисунка

Таблицю розташовують безпосередньо після тексту, у якому її згадують вперше, або на наступній сторінці. Таблиця повинна мати назву, яку друкують малими літерами (окрім першої великої) і вміщують над таблицею. Назва має бути стислою і відображати зміст таблиці.

Слово "Таблиця", її номер і назву вказують один раз зліва над першою частиною таблиці. Над іншими частинами пишуть: "Продовження таблиці номер" із зазначенням номера таблиці.

Таблиця 1 – Приклад таблиці

№ з/п	Назва стовпця 1	Назва стовпця 2	Назва стовпця 3

Формули розташовують безпосередньо після тексту, у якому їх згадують, посередині сторінки. Вище і нижче від кожної формули залишають не менше одного вільного рядка.

Формули слід нумерувати порядковою нумерацією в межах розділу. Номер формули складається з номера розділу і порядкового номера формули, між якими ставлять крапку. Номер формули друкують на рівні формули в дужках у крайньому правому положенні на рядку.

Пояснення значень символів, що входять до формули, наводять безпосередньо під формулою в тій послідовності, у якій їх подано у формулі. Пояснення значення слід давати з нового рядка. Перший рядок пояснення починають з абзацу словом "де" без двокрапки.

Наприклад:

$$f(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} X_{ij} \to \min, \qquad (1.1)$$

де $f(x)$ – це цільова функція; тощо.

3. Порядок організації захисту та критерії оцінювання курсових проєктів

Завершений роздрукований курсовий проєкт з об'єктно-орієнтованого програмування здобувачі вищої освіти підписують і здають керівнику для перевірки, перевірки на антиплагіат і рецензування. Перевірку курсових проєктів здійснюють протягом 10 днів з моменту їх надання.

Для перевірки курсового проєкту на антиплагіат здобувач вищої освіти має здати електронну версію курсового проєкту у форматі *.rtf, *.doc, *.docx, *.pdf.

Здати проєкт здобувач вищої освіти має не пізніше ніж за два тижні до захисту, терміни якого встановлюють відповідно до навчальних планів і графіків. Керівник курсового проєкту після перевірки робить висновок про актуальність і якість виконання роботи та про допуск (недопуск) здобувача вищої освіти до захисту роботи. Позначку про допуск (недопуск) до захисту роблять на титульній сторінці курсового проєкту за підписом керівника.

Якщо в результаті перевірки виявлено істотні помилки, неповний обсяг, низьку якість роботи або проєкт не пройшов перевірку на антиплагіат, його повертають здобувачу вищої освіти для доопрацювання із зауваженнями керівника в письмовій формі. На титульному аркуші керівник ставить напис "Доробити" чи "До захисту" і дата перевірки.

У разі відповідності курсового проєкту вимогам цих методичних рекомендацій керівник робить напис на титульному аркуші "До захисту". Допущений до захисту курсовий проєкт здобувач вищої освіти захищає у присутності здобувачів вищої освіти академічної групи. Не допущену до захисту роботу передають здобувачу вищої освіти на доопрацювання, і процедура подання на перевірку та рецензування повторюється.

Захист курсових проєктів може проходити в такій формі: автору проєкту дають до 10 хв для доповіді основних положень, після чого йому ставлять питання щодо змісту роботи.

Оцінювання якості виконання і захисту здобувачами вищої освіти курсового проєкту здійснюють за 100-бальною шкалою. Підсумкову оцінку визначають викладачі, які приймають захист курсових проєктів. Об'єктами оцінювання є три складові: зміст, оформлення та захист курсового проєкту (табл. 2).

Таблиця 2

Розподіл бальної оцінки за виконання курсового проєкту

Об'єкт оцінювання	Максимальна кількість балів, яку може одержати здобувач вищої освіти
Розкриття змісту курсового проєкту	50
Оформлення курсового проєкту	10
Захист курсового проєкту	40

Критерії оцінювання змісту в курсовому проєкті (0 – 50 балів):

- ступінь розкриття теоретичних аспектів проблеми, вибраної для дослідження;
- наявність практичного висвітлення проблематики курсового проєкту;
- наочність і якість ілюстративного матеріалу;
- дослідження вітчизняних і зарубіжних джерел літератури;
- рівень обґрунтування запропонованих рішень;
- відповідність отриманих результатів поставленим цілям із завданням.

Критерії оцінювання оформлення курсового проєкту (0 – 10 балів):

- відповідність обсягу та оформлення курсового проєкту встановленим вимогам;
- посилання на використану літературу і нормативні документи.

Критерії оцінювання захисту курсового проєкту (0 – 40 балів):

- уміння чітко, зрозуміло та стисло викладати основні завдання проведеного дослідження;
- повнота, глибина, обґрунтованість відповідей на запитання членів комісії за змістом;
- ґрунтовність висновків і рекомендацій щодо практичного використання результатів дослідження.

4. Етапи виконання курсового проєкту

Курсовий проєкт з об'єктно-орієнтованого програмування здобувачі вищої освіти виконують у IV семестрі. Завдання видають на початку семестру.

Вибір теми та її затвердження

Здобувач вищої освіти може вибрати будь-яку запропоновану в цих методичних рекомендаціях тему або погодити з керівником власну тему. Вибрані здобувачами вищої освіти теми затверджують на засіданні кафедри інформатики та комп'ютерної техніки, після чого тему змінити не можна.

Складання плану і його реалізація

Після отримання теми здобувач вищої освіти має протягом трьох тижнів подати керівникові перший розділ, у якому потрібно навести

специфікацію проєкту. Після уточнення з викладачем плану теми затверджують розклад консультацій з розрахунку 2 год на курсовий проєкт. Здобувач вищої освіти має право на консультацію за курсовим проєктом у зазначений час, в інший час – тільки за погодженням з викладачем.

Рекомендована література

Основна

1. ДСТУ 8302:2015 Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. – Київ : Держстандарт України, 20015. – 52 с.

2. Карпенко М. Ю. Технології створення програмних продуктів та інформаційних систем : навч. посіб. / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 93 с.

3. Стандарт вищої освіти України галузі знань 12 Інформаційні технології спеціальності 126 Інформаційні системи та технології першого (бакалаврського) рівня вищої освіти від 12.12.2018 р. № 1380 // ВВФ. – 2018. – № 10.

4. Щербаков О. В. Основи об'єктно-орієнтованого програмування [Електронний ресурс] : навч. посіб. / О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 237 с. – Режим доступу : http://www.repository.hneu.edu.ua/handle/123456789/23847.

5. Pro JavaFX 8 : A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients / J. Vos, W. Gao, S. Chin et al. – New York : Apress, 2018. – 588 p.

Додаткова

6. Michaelis M. Essential C# 6.0 / M. Michaelis, E. Lippert. – Boston : Addison-Wesly, 2016. – 1004 p.

7. Sharan K. Learn JavaFX 8 / K. Sharan. – New York : Apress, 2015. – 1200 p.

Інформаційні ресурси

8. Парфьонов Ю. Е. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Основи об'єктно-орієнтованого програмування" [Електронний ресурс] / Ю. Е. Парфьонов. – Режим доступу : http://www.ikt.hneu.edu.ua/course/view.php?id=879.

9. Programming Tutorials and Source Code Examples [Electronic resource]. – Access mode : http://www.java2s.com.

10. Design Patterns [Electronic resource]. – Access mode : http://sourcemaking.com/design_patterns.

Додатки

Додаток А

Тематика курсових проєктів з об'єктно-орієнтованого програмування

Дозволено самостійний вибір здобувачами вищої освіти теми курсового проєкту за погодженням з викладачем.

1. Розроблення програмного продукту для роботи з файловою базою даних про прихід товарів на склад підприємства.
2. Розроблення програмного продукту для роботи з файловою базою даних про фізичних осіб.
3. Розроблення програмного продукту для роботи з файловою базою даних про рух пасажирських поїздів на станції Харків.
4. Розроблення програмного продукту для роботи з файловою базою даних про тривалість розмов абонентів АТС.
5. Розроблення програмного продукту для роботи з файловою базою даних про товари магазину побутової техніки.
6. Розроблення програмного продукту для роботи з файловою базою даних про банківські операції.
7. Розроблення програмного продукту для роботи з файловою базою даних результатів моніторингу якості палива на автозаправній станції.
8. Розроблення програмного продукту для роботи з файловою базою даних про поштові операції.
9. Розроблення програмного продукту для роботи з файловою базою даних заявок на ремонт мобільних телефонів у сервісному центрі.
10. Розроблення програмного продукту для роботи з файловою базою даних контактів.
11. Розроблення програмного продукту для роботи з файловою базою даних про поштову індексацію м. Харкова та населених пунктів районів Харківської області.
12. Розроблення програмного продукту – довідника куратора студентської групи.
13. Розроблення програмного продукту для роботи з файловою базою даних про продаж палива на автозаправній станції.

14. Розроблення програмного продукту для роботи з файловою базою даних про час використання комп'ютерів підприємства.

15. Розроблення програмного продукту для роботи з файловою базою даних про надання послуг абонентам інтернет-провайдера.

16. Розроблення програмного продукту для роботи з файловою базою даних про хід виконання робіт на задану дату.

17. Розроблення програмного продукту для роботи з файловою базою даних про рух транспортних засобів.

18. Розроблення програмного продукту для роботи з файловою базою даних штатного розпису підприємства.

19. Розроблення програмного продукту для роботи з файловою базою даних про залізничні пасажирські перевезення.

20. Розроблення програмного продукту для роботи з файловою базою даних про продаж книг у книгарні.

21. Розроблення програмного продукту для роботи з файловою базою даних про нарахування зарплати співробітникам підприємства.

22. Розроблення програмного продукту для роботи з файловою базою даних про витрати палива на автобазах міста.

23. Розроблення програмного продукту для роботи з файловою базою даних про використання машинного часу в обчислювальному центрі.

24. Розроблення програмного продукту для роботи з файловою базою даних про споживання електроенергії на заводах міста.

25. Розроблення програмного продукту для роботи з файловою базою даних про рух матеріалів на складі підприємства.

26. Розроблення програмного продукту для роботи з файловою базою даних про прибуток підприємства за звітний період.

27. Розроблення програмного продукту для роботи з файловою базою даних про відвідування занять студентами.

28. Розроблення програмного продукту для роботи з файловою базою даних про поставки продукції.

29. Розроблення програмного продукту для роботи з файловою базою даних про перевезення авіапасажирів.

30. Розроблення програмного продукту для роботи з файловою базою даних про час роботи верстатів підприємства.

31. Розроблення програмного продукту для роботи з файловою базою даних про випуск деталей робітниками цеху.

32. Розроблення програмного продукту для роботи з файловою базою даних про рух основних фондів підприємства.

33. Розроблення програмного продукту для роботи з файловою базою даних про облік запчастин на складі підприємства.

34. Розроблення програмного продукту для роботи з файловою базою даних про успішність студентів.

35. Розроблення програмного продукту для роботи з файловою базою даних про облік оплати за телефонні розмови.

36. Розроблення програмного продукту для роботи з файловою базою даних про облік автомобілів в автосалоні.

37. Розроблення програмного продукту для роботи з файловою базою даних про відвідування інтернет-ресурсів користувачами.

38. Розроблення програмного продукту для роботи з файловою базою даних про автомобільні запчастини на складі автомагазину.

39. Розроблення програмного продукту для роботи з файловою базою даних про отримання студентами літератури в бібліотеці факультету.

40. Розроблення програмного продукту для роботи з файловою базою даних про рух матеріалів на складі.

41. Розроблення програмного продукту для роботи з файловою базою даних про абітурієнтів, які подали документи для вступу до закладу вищої освіти.

42. Розроблення програмного продукту для роботи з файловою базою даних про ціни на основні продукти харчування в місті.

43. Розроблення програмного продукту для роботи з файловою базою даних про нарахування зарплати співробітникам.

44. Розроблення програмного продукту для роботи з файловою базою даних про випуск продукції підприємством.

45. Розроблення програмного продукту для роботи з файловою базою даних про співробітників підприємства.

46. Розроблення програмного продукту для роботи з файловою базою даних про замовлення клієнтів.

Додаток Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Кафедра інформатики
та комп'ютерної техніки

КУРСОВИЙ ПРОЄКТ З ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ
на тему "_____"

Здобувач (-ка) вищої освіти
2-го курсу_____групи
спеціальності 126 "Інформаційні
системи та технології"
_____
(ПІБ)

Керівник: учений ступінь, учене
звання, посада
_____
(ПІБ)

Національна шкала:_____
Кількість балів:___ Оцінка ECTS:__

Члени комісії: _____ _____
(підпис) (ПІБ)
_____ _____
(підпис) (ПІБ)
_____ _____
(підпис) (ПІБ)

Харків – 2024

Додаток В

Зразок оформлення змісту курсового проєкту

ЗМІСТ

Додаток Г

Зразок завдання на курсовий проєкт

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет інформаційних технологій
Кафедра інформатики та комп'ютерної техніки

ЗАВДАННЯ
на курсовий проєкт із об'єктно-орієнтованого програмування
здобувачу вищої освіти 2-го курсу групи 6.04.126.010.21.01
Панченку Івану Васильовичу

1. Тема проєкту: "Розроблення програмної системи для автоматизації роботи з файловою базою даних".

2. Дата видачі завдання: "__" _____20_ р.

3. Термін здачі здобувачем вищої освіти закінченого проєкту " " 20_ р.

4. Вхідні дані до проєкту: літературні джерела, технічна документація щодо розроблення програм, ДСТУ з оформлення документації.

5. Зміст курсового проєкту: Вступ. Специфікація проєкту. Програмна документація. Висновки. Додатки.

6. Перелік графічного матеріалу: UML-діаграма класів програмної системи.

Закінчення додатка Г

Таблиця Г.1

Календарний план виконання курсового проєкту

Здобувач вищої освіти _____ Іван ПАНЧЕНКО

Керівник _____Олексій ГОРОХОВАТСЬКИЙ

Додаток Д

Методичні рекомендації щодо розроблення інтерфейсу користувача

Розроблення інтерфейсу користувача (UI) – це процес вдосконалення презентації та інтерактивності додатків, який фокусують на зовнішньому вигляді програми та її взаємодії з користувачами.

Користувацький інтерфейс містить в себе текстові блоки, що читають користувачі, кнопки, текстові поля, форми, зображення та інші візуальні елементів, які бачить і з якими взаємодіє користувач під час використання програми. Також користувацький інтерфейс складається з макета екранів, переходів між сторінками, анімації та кожної мікровзаємодії з користувачем.

Розроблення взаємодії з користувачем (UX) є процедурою поліпшення загального досвіду користувачів під час взаємодії з додатком або вебсайтом для досягнення основної мети – забезпечити максимальне задоволення споживачів, щоб користувачі знаходили в продукті додаткову цінність.

UX – підхід включає деякий повторювальний цикл під час розроблення додатка, який складається з:

дослідження (початкове дослідження користувачів для розуміння цільової аудиторії та її можливостей, обмежень, цілей та очікування);

дизайну (у ході створення якого використовують статистику досліджень користувачів, щоб допомогти генерувати ідеї та початковий прототип для реалізації концепцій);

оцінювання (фіксування відгуків користувачів протягом розроблення проєкту).

Основні властивості, яким мають відповідати інтерфейси:

Адаптованість, яка означає, що інтерфейс має:

- бути сумісним із потребами та можливостями користувача;
- забезпечувати простоту переходу від виконання однієї функції до іншої;
- на високому рівні забезпечувати користувача вказівками стосовно його можливих дій, а також генерувати належний зворотний зв'язок на його запити;

• надавати користувачу можливість відчувати себе повноправним керівником ситуації під час розв'язання всіх типів задач, тобто, забезпечувати його всією необхідною інформацією; користувач має бути впевненим, що він сам розв'язує поставлену задачу;

• забезпечувати користувача різними, взаємодоповнюючими формами подання результатів залежно від типу запиту або від характеру отриманого рішення;

• ураховувати особливості користувачів різних рівнів.

Достатність інтерфейса означає:

• допустимі запити користувача мають бути чіткими і однозначними для користувачів усіх рівнів, а також для прикладних задач усіх типів;

• реакція системи на всі типи запитів також має бути однозначною та зрозумілою і, по можливості, простою.

Дружність інтерфейсу. Це максимальна простота його використання і готовність повною мірою задовольнити запити користувача під час розв'язання визначеного класу задач.

Гнучкість інтерфейсу. Гнучкість інтерфейсу – це можливість його адаптування до розв'язання конкретної задачі. Якщо розв'язувана задача дуже складна, то інтерфейс має полегшувати формулювання запитів і видавати результати у формі, яка легко і швидко сприймається користувачем.

Тобто інтерфейс має бути максимально простим навіть у випадку, коли розв'язують дуже складну задачу.

Основні правила створення інтерфейсу користувача

Правило доступності. Система має бути настільки зрозумілою, щоб користувач, який ніколи раніше її не бачив, але добре розбирається в предметній області, міг без жодного навчання почати її використовувати. Це правило є деяким ідеалом, якого потрібно прагнути, оскільки на практиці досягти такої міри доступності майже ніколи не вдається.

Правило ефективності. Система не має перешкоджати ефективній роботі досвідчених користувачів, які працюють із нею протягом тривалого часу. Очевидним прикладом порушення цього правила є націленість системи тільки на новачків, використання засобів, які добре підходять для недосвідченого користувача, обмежуючи його в можливості зробити щось не так, але неефективні для експерта, який і так знає, що і де йому треба зробити.

Правило безперервного розвитку. Система має сприяти безперервному вдосконаленню знань, умінь і навичок користувача і пристосовуватися до його досвіду, що змінюються. Погані результати приносить надання тільки базових можливостей або залишення початкуючого користувача-початківця наодинці зі складним інтерфейсом, яким упевнено користуються експерти. Порушення безперервності під час переходу від одного набору можливостей до іншого також є незручним, оскільки користувач вимушений розбиратися з доданими можливостями в новому контексті.

Правило дотримання контексту. Система має бути погодженою з контекстом, у якому їй належить працювати. Це правило вимагає від системи бути працездатною не "взагалі", а саме в тому середовищі, в якому нею користуватимуться. У контекст можуть входити специфіка і об'єми вхідних і вихідних даних, тип і цілі організацій, в яких система має працювати, рівень користувачів, зашумленість приміщень тощо.

Принципи розроблення інтерфейсу користувача

Принцип структуризації. Призначений для користувача інтерфейс має бути доцільно структурований. Близькі за змістом, споріднені його частини мають бути пов'язані видимим чином, а незалежні – розділені; схожі елементи мають виглядати схоже, а несхожі – розрізнятися.

Принцип простоти. Найпоширеніші операції мають виконуватися максимально просто. При цьому повинні бути видимі посилання на складніші процедури.

Принцип видимості. Усі функції і дані, необхідні для розв'язання певної задачі, мають бути видимі, коли користувач намагається її розв'язати.

Принцип зворотного зв'язку. Користувач має отримувати повідомлення про дії системи і про важливі події всередині неї. Повідомлення повинні бути інформативними, короткими, однозначними і написаними мовою, зрозумілою користувачеві.

Принцип толерантності. Інтерфейс має бути гнучким і терпимим до помилок користувача. Збиток від помилок повинен знижуватися завдяки можливості відміни і повтору дій і завдяки розумній інтерпретації будь-яких розумних дій користувача і введених їм даних. По можливості слід уникати взаємодії (модальних діалогів), основаної на обмеженні свободи користувача.

Принцип повторного використання. Слід намагатися багаторазово використовувати внутрішні та зовнішні компоненти, забезпечуючи уніфікованість інтерфейсу і схожість між його схожими елементами.

Взаємодія між користувачем і комп'ютером

Людино-машинний інтерфейс забезпечує зв'язок між користувачем і комп'ютером. Він дозволяє досягати поставлених цілей, успішно знаходити розв'язання поставленої задачі.

Взаємодія – обмін діями і реакціями на ці дії між комп'ютером і користувачем. Є різні стилі взаємодії, які підрозділяють на два види.

1. Використання інтерфейсу мови команд – уведення команд текстовими засобами.

2. Безпосереднє маніпулювання.

Отже, є кілька способів, якими користувач міг би зв'язуватися з комп'ютером:

мова команд – користувач управляє системою, уводячи відповідні команди в текстовому режимі;

питання і відповідь – діалог, коли комп'ютер ставить питання, а користувач відповідає йому (чи навпаки);

форми – користувач заповнює форми або поля діалогу, уводячи дані у відповідні поля;

меню – користувач забезпечений рядом опцій і управляє системою,вибираючи необхідні пункти;

пряме маніпулювання – користувач управляє об'єктами на екрані за допомогою пристрою маніпулювання типу миші.

Мета створення ергономічного інтерфейсу полягає в тому, щоб відобразити інформацію настільки ефективно, наскільки це можливо для людського сприйняття, і структурувати відображення на дисплеї так, щоб привернути увагу до найбільш важливих одиниць інформації. Основна мета – мінімізувати загальну інформацію на екрані й подати тільки те, що є необхідним для користувача.

Основні принципи створення меню

Під час проєктування меню застосунку потрібно вибрати найкращий спосіб відображення меню, щоб воно було зрозумілим і легким у використанні. Зазвичай команди меню впорядковано деяким ієрархічним способом. Основна проблема полягає в тому, щоб правильно розподілити різні пункти меню за різними рівнями і правильно їх згрупувати.

Принципи проєктування меню:

структура меню має відповідати структурі завдання, розв'язуваного системою. Організація меню має відобразити найефективнішу послідовність кроків, що ведуть до розв'язання поставленої задачі;

пункти меню мають бути короткими, граматично правильними і відповідати своєму заголовку. Порядок пунктів меню вибирають відповідно до угоди, частоти і порядку використання, а також залежно від потреб завдання або користувача;

вибір пунктів меню слід забезпечити декількома способами – за допомогою клавіатури, за допомогою миші та через інші об'єкти призначеного для користувача інтерфейсу.

Важливо зафіксувати поєднання клавіш, що легко запам'ятовуються, для швидшого доступу до пунктів меню, оскільки це дуже економить час.

Основні принципи проєктування форм

Форми – основний елемент інтерфейсу. Призначення форм – зручне введення і перегляд даних, стану, повідомлень розробленого застосунку.

При розробленні форм застосунку потрібно дотримуватись таких принципів проєктування:

форму проєктують для зручного, зрозумілого і швидкого розв'язання поставленої задачі. Якщо форму переносять з паперового варіанта, то пересування по суміжних полях не має викликати утруднень у користувача;

розміщення інформаційних одиниць на просторі форми має відповідати логіці її майбутнього використання: це залежить від необхідної послідовності доступу до інформаційних одиниць, частоти їх використання, а також від відносної важливості елементів;

логічні групи елементів потрібно відділяти пропусками, рядками, колірними або іншими візуальними засобами;

взаємозалежні або пов'язані елементи слід відображатися в одній формі.

Під час розроблення форм потрібно продумати і вказати, які кнопки в смузі системного меню мають бути доступні в тому або іншому вікні, чи повинне вікно допускати зміни користувачем його розміру, яким має бути заголовок вікна.

Під час проєктування форм слід прагнути до використання обмеженого набору кольорів і приділяти увагу їх правильному поєднанню. Для фону форми вибирають нейтральні кольори (світло-сірі). Колір не використовують як основний засіб передачі інформації, потрібно вибирати системні кольори, які користувач може перебудовувати на свій розсуд.

Елементи, що управляють, і функціонально пов'язані з ними компоненти екрана слід зорово об'єднувати в групи, заголовки яких коротко і чітко пояснюють їхнє призначення. Кожне вікно повинне мати деяку центральну тему, підпорядковану його композиції.

Користувач має розуміти, для чого призначено це вікно і що в ньому найважливіше. Неприпустимо перевантажувати вікно великим числом елементів управління введення і відображення інформації.

Формати введення. Слід забезпечити введення значень за замовчуванням в усі поля, які це допускають і де така функція не дратуватиме користувача. Можна призначити клавіші або коди для введення значень, що часто повторюються.

Вхідні дані мають бути значущими і загальноприйнятими. Не треба об'єднувати поля введення чисел і символів, оскільки числові й алфавітні клавіші розміщено незручно одна відносно одної на клавіатурі.

Слід виключити часте перемикання між верхнім і нижнім регістрами для прискорення введення даних.

Бажано використати значення за замовчуванням, щоб мінімізувати процес уведення інформації.

Організація системи навігації та системи відображення станів.

Навігація забезпечує користувачеві можливість переміщення між різними екранами, інформаційними одиницями і підпрограмами в автоматизованій системі. У повноцінній системі користувач завжди може отримати інформацію про стан системи, про процес виконання або активну підпрограму.

Загальні принципи проєктування

Є певні навігаційні засоби і прийоми, які допомагають користувачеві орієнтуватися в системі. Вони включають використання заголовків сторінок для кожного екрана, номерів сторінок, рядків і стовпців, відображення поточного імені файлу вгорі екрана. Тип системи навігації залежить від прийнятого стилю інтерфейсу. Для інтерфейсів мови команд існує дуже мало способів забезпечення повноцінної навігації.

В інтерфейсах з меню можна використати ієрархічно структуроване меню. Діалогові інтерфейси самі по собі захищають користувача від помилкових дій. Інформація стану зазвичай відображається внизу екрана і містить дані про кількість записів, число оброблених одиниць, процес друку, черги друку тощо.

Проєктування повідомлень. Повідомлення потрібні для спрямування дій користувача, підказок і попереджень під час виконання необхідних дій на шляху розв'язання задачі. Вони також включають підтвердження дій з боку користувача і підтвердження з боку системи, що завдання виконано успішно або з якихось причин не виконані.

Повідомлення можна вивести у формі діалогу, екранних заставок та ін. Повідомлення можуть запропонувати користувачеві:

вибрати із запропонованих альтернатив опцію або набір опцій;

увести інформацію;

вибрати опцію з набору опцій, які можуть змінюватися залежно від поточного контексту;

підтвердити фрагмент уведеної інформації перед продовженням уведення.

Повідомлення можна помістити в модальні діалогові вікна, які змушують користувача відповісти на питання, перш ніж почне виконуватися будь-яка інша дія. Це може бути корисно, коли система має змусити користувача обдумати рішення перед продовженням роботи. Немодальні діалогові вікна дозволяють працювати з іншими елементами інтерфейсу, тоді як саме вікно можна ігнорувати.

Запобігання, виявлення і виправлення помилок

Помилки користувача можуть ґрунтуватися на неправильному розумінні дії чи порядку дій або бути випадковими, неумисними, наприклад, друкарська помилка під час уведення тексту.

Помилки другого виду можна розподілити на три підвиди:

неточність у виборі опції (наприклад, користувач випадково натиснув кнопку "Вихід", і програма закрилася);

втрата активності, коли користувач забуває необхідну послідовність дій для продовження роботи;

помилка режиму або стану, коли користувач думає, що він перебуває в одному стані, а фактично – в іншому.

Користувач завжди робитиме помилки навіть у відмінній програмній системі, тому в системі, що розроблюють, завжди має бути передбачено захист від помилок. Техніка цього захисту включає такі аспекти:

примусові дії в системі, які запобігають або утрудняють появу помилок;

забезпечення хороших та інформативних повідомлень про помилки;

забезпечення нормальної діагностики системи, у процесі якої користувачеві пояснюють, у чому полягає сутність помилки, і вказують шляхи її виправлення.

Основні принципи оброблення помилок у формах введення

забезпечення можливості посимвольного редагування введених записів для виправлення помилок уведення (друкарських помилок);

якщо помилку виявлено системою, бажано повернути курсор у поле з помилковими даними і яким-небудь чином виділити це поле;

виводити значущі повідомлення про помилки, що використовують стиль мови користувача і відповідну термінологію;

виводити повідомлення про помилки, які пояснюють і пропонують шляхи їх усунення.

Ефективність запобігання і подолання помилок користувачів тим вища, чим рідше користувачі помиляються під час роботі із цим інтерфейсом і чим менше часу та зусиль потрібно для подолання наслідків уже зроблених помилок.

Приклад UML-діаграми класів

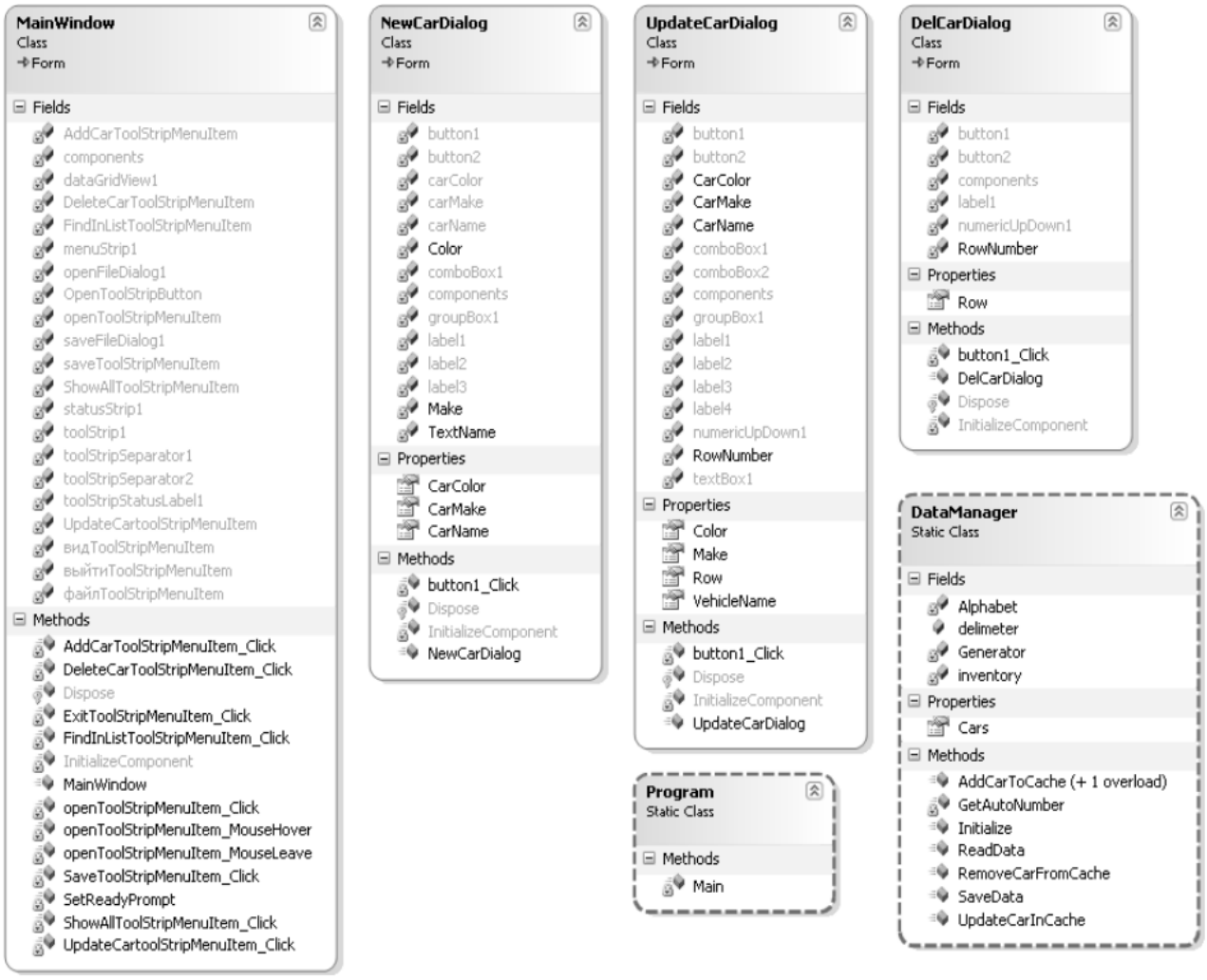

Рисунок 2.1 – UML-діаграма класів

Програма складається з шести класів. Чотири з них – MainWindow, NewCarDialog, UpdateCarDialog, DelCarDialog – є похідними від класу javax.swing.JFrame, тобто мають графічний інтерфейс.

Клас MainWindow становить головне вікно програми. Найважливішим елементом управління в головному вікні є таблиця, яка відображує дані про автомобілі.

Клас NewCarDialog є діалоговим вікном для введення даних про новий автомобіль. Майже всі поля цього класу відповідають елементам управління діалогового вікна.

Клас UpdateCarDialog є діалоговим вікном для оновлення даних про автомобіль. Призначення його полів, методів та властивостей аналогічно відповідним елементам класу NewCarDialog.

Клас DelCarDialog є діалоговим вікном, призначеним для видалення даних про автомобіль.

Клас DataManager призначено для управління даними програми, що зберігаються в оперативній пам'яті та на жорсткому диску комп'ютера. Він містить статичні поля, властивості та методи.

Клас Program – головний клас програми. Містить метод main, який є "точкою входу" під час запуску програми на виконання.

Взаємодія об'єктів класів програми відбувається таким чином.

Після запуску програми на виконання створюється об'єкт класу MainWindow і в його конструкторі ініціалізуються елементи графічного інтерфейсу шляхом створення об'єктів відповідних класів. Також у ньому ініціалізується клас DataManager.

Створення об'єктів інших класів та виконання їхніх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейс у програми.

Додаток Ж

Приклад тест-плану, сполученого зі звітом про проведення тестування

Тестовий приклад: № 1. Призначення: перевірка того, що програмна система дозволяє виконувати читання даних із файлу та коректно відображати їх у графічному інтерфейсі користувача.

Тест-вимоги, що перевіряють: функціональна вимога № 2.

Передумови для тесту: програмну систему потрібно запустити, а на диску комп'ютера має бути файл із даними у визначеному форматі.

Критерій проходження тесту: реальна поведінка програмної системи збігається з очікуваною.

Таблиця Ж.1

Тест-план

№ з/п	Крок сценарію	Очікуваний результат	Отриманий результат	Відмітка про проходження кроку сценарію (Так/Ні)
1	У меню "Файл" вибрати пункт "Відкрити"	Має з'явитися діалогове вікно відкриття файла	Діалогове вікно відкриття файла з'являється	Так
2	У діалоговому вікні відкриття файла вибрати ім'я файла з даними визначеного формату та натиснути кнопку "ОК"	У головному вікні програми мають коректно відобразитися дані, завантажені з файла	Дані, що були завантажені з файла, коректно відображуються в головному вікні програми	Так
3	У діалоговому вікні відкриття файла вибрати ім'я файла з даними в недопустимому форматі та натиснути кнопку "ОК	Має з'явитися діалогове вікно з повідомленням про те, що дані не можуть бути завантажені	З'являється діалогове вікно з повідомленням	Так

Відмітка про проходження тесту (пройдено / не пройдено): пройдено.
Тестовий приклад: № 2.

Тестовий приклад: № 3.

Тестових прикладів виконано: 3.
Тестових прикладів пройдено: 1.

Додаток И

Опис процедури розгортання програмного продукту, створеного на платформі JavaSE

Вимоги до апаратних засобів:

1. Процесор – не нижче Pentium 2266 МГц.
2. Вільний дисковий простір – не менше 124 Мб.
3. Доступний простір ОЗУ – не менше 128 Мб.

Вимоги до програмних засобів:

1. Операційна система:

• Windows XP – Windows 10;
• Linux • MacOSX 10.7.3(Lion) і старше;
• Solaris10 і старше.

2. JavaRuntimeEnvironment 8 і старше.

Розгортання програмного продукту на комп'ютері користувача у вигляді автономного застосунку:

1. Створіть на цільовому диску каталог для застосунку, наприклад, MyApp.

2. У каталозі MyApp створіть новий каталог, наприклад, dist.

3. Скопіюйте виконуваний jar-файл застосунку (наприклад, install.jar) у каталог dist.

4. Завантажте з вебсайту компанії Oracle програмну платформу JavaRuntimeEnvironment потрібної версії та розпакуйте відповідний архів у деякий каталог, наприклад, у папку appjre.

5. Перемістить каталог appjre у каталог MyApp.

6. У каталозі MyApp створіть командний файл цільової операційної системи, наприклад, start.bat (операційна система Windows).

7. Додайте в start.bat такі команди (операційна система Windows): @echoOFFset PATH =.\appjre\binjava -jar dist\install.jarpause> NUL.

8. Для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою мишки на файлі start.bat (операційна система Windows).

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні рекомендації до виконання курсового проєкту

для здобувачів вищої освіти спеціальності 126 "Інформаційні системи та технології" освітньої програми "Інформаційні системи та технології" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладачі: **Удовенко** Сергій Григорович
Тютюник Ольга Олександрівна
Гороховатський Олексій Володимирович
Парфьонов Юрій Едуардович

Відповідальний за видання *С. Г. Удовенко*

Редактор *Н. Г. Войчук*

Коректор *В. О. Дмитрієва*

План 2024 р. Поз. № 97 ЕВ. Обсяг 43 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру
ДК № 4853 від 20.02.15 р.