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: 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

О-29 Об'єктно-орієнтоване програмування: [Електронне видання] методичні рекомендації до виконання курсового проєкту для здобувачів вищої освітиспеціальності 126 «Інформаційні системи та технології» освітньої програми «Інформаційні системи та технології» першого (бакалаврського) рівня / уклад. О. О. Тютюник, О. В. Гороховатський, Ю.Е. Парфьонов – Харків : ХНЕУ ім. С. Кузнеця, 2024. – 47 с.

Призначено для виконання курсового проєкту освітньо-кваліфікаційного рівня "бакалавр" здобувачами вищої освіти, що навчаються на другому курсі у четвертому семестрі. Подано вимоги до змістовного наповнення й структурного представлення курсового проєкту.

Рекомендовано для здобувачів вищої освіти освітньої програми"Інформаційні системи та технології" спеціальності 126 "Інформаційні системи та технології" першого (бакалаврського) рівня і викладачів, що є керівниками курсового проєкту за вказаною освітньою програмою.

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 покращення якості управління.

Використання сучасних ІТ-рішень допомагає підвищити рівень обґрунтованості та 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 обумовлюють такі фактори, як: зростання складності та функціональності комп'ютерної техніки, необхідність створенняпрограм, які можуть працювати на різних платформах та інтегрувати їхню функціональність, потреба у розробленні програм для обробки 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 використання 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 компонент вивчають «Програмування», «Основи алгоритмізації» та «Об'єктно-орієнтоване програмування». В процесі вивчення відповідних освітніх компонент 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 спрямований на більш глибоке осмислення і закріплення теоретичних знань, отриманих під час вивчення зазначених освітніх компонент, та на удосконалення практичних навичок щодо розроблення програмного забезпечення та необхідної документації.

У методичних рекомендаціях до виконання курсового проєкту з об’єктно-орієнтованого програмування наведено зміст та правила оформлення курсового проєкту, етапи виконання, орієнтована тематика проєктів (додаток А).

МЕТА І ЗАВДАННЯ ВИКОНАННЯ КУРСОВОГО ПРОЄКТУ З ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Метою написання курсового проєкту з об'єктно-орієнтованого програмування є закріплення теоретичних знань, отриманих на лекційних та лабораторних заняттях освітніх компонент «Програмування», «Основи алгоритмізації», «Об'єктно-орієнтоване програмування» 126 спеціальності «Інформаційні системи та технології» освітньо-професійної програми «Інформаційні системи та технології» першого (бакалаврського) рівня.

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

В результаті виконання курсового проєкту здобувач вищої освіти набуває наступні загальні і спеціальні компетентності та забезпечує програмні результати навчання (табл.1).

Таблиця 1

Компетентності та результати навчання

Компетентності	Результати навчання
КЗ 1, КЗ 2, КС 4.	ПР 2.
КЗ 3, КЗ 8, КС 4.	ПР 3.

де КЗ 1 – Здатність до абстрактного мислення, аналізу та синтезу. КЗ 2 – Здатністьзастосовуватизнання у практичнихситуаціях.

КЗ 3 – Здатність до розуміння предметної області та професійної діяльності.

КЗ 8 – Здатність оцінювати та забезпечувати якість виконуваних робіт.

КС 4 – Здатність проектувати, розробляти та використовувати засоби реалізації інформаційних систем, технологій та інфокомунікацій (методичні, інформаційні, алгоритмічні, технічні, програмні та інші).

ПР 2 – 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.

ПР 3 – 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 загально теоретичну та спеціальну підготовку здобувача вищої освіти і в остаточному підсумку готує його до майбутнього виконання більш складного й завершального етапу навчального процесу – дипломного проєктування.

1. **ЗМІСТ ТА ОФОРМЛЕННЯКУРСОВОГО ПРОЄКТУ**

Зміст курсового проєкту з об’єктно-орієнтованого програмування

Зміст повинен відповідати типовому змісту курсових проєктів з об’єктно-орієнтованого програмування, який запропоновано у методичних рекомендаціях за винятком розроблення індивідуальних тем курсових проєктів. Увесь матеріал курсового проєкту повинен бути цілеспрямованим, викладений у логічній послідовності і взаємопов’язаний з окремими його розділами.

Пояснювальна записка повинна містити всі розділи курсового проєкту:

Перший аркуш – титульний.

Другий аркуш – зміст пояснювальної записки, де зазначені сторінки розташування розділів проєкту.

Далі розділи проєкту відповідно до рекомендацій, наведених далі.

Список використаних джерел подається після висновку. Він повинен бути оформлений згідно зі встановленими правилами.

Додатки оформляють (якщо їх не багато) з використанням літер українського алфавіту (наприклад, Додаток А “Лістинги програм”).

ТИТУЛЬНИЙ АРКУШ

Першою сторінкою пояснювальної записки є **титульний аркуш**. Він містить такі дані:

відомості про виконавця роботи;

повна назва документа;

підписи відповідальних осіб, включаючи керівника роботи;

рік складання пояснювальної записки.

Приклад титульного аркуша курсового проєкту з об’єктно-орієнтованого програмування наведено у додатку Б.

ЗМІСТ

До **змісту** додають: вступ; назви всіх розділів, підрозділів та пунктів основної частини пояснювальної записки; висновки; перелік посилань; назви додатків і номери сторінок, які містять початок матеріалу.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Якщо у пояснювальній записці використовують маловідомі скорочення, нові символи, позначення і таке інше, то в ній має бути **перелік умовних скорочень**, який подається у вигляді окремого списку, що 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 повторюються менше трьох разів, перелік не складають, а їх розшифрування наводять у тексті під час першого згадування.

Приклад переліку умовних скорочень наведений у роботі[3].

ВСТУП

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

У вступі необхідно зазначити важливість використання інформаційних систем у сучасних умовах, роль інформаційних систем у предметній галузі, щорозглядається в 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, які були використані під час розроблення програми).Також необхідно 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" наводиться постановка завдання на розроблення програмного продукту відповідно до варіанта,який був виданий здобувачу вищої освіти. Також необхідно вказати, що розробка виконується на підставі завдання на виконання курсового проєкту, затвердженого кафедрою інформатики та комп’ютерної техніки.

У підрозділі 1.2 "Вимоги до програмного забезпечення" містяться функціональні та нефункціональні вимоги до програмного продукту.

Функціональні вимоги явно описують, що повинен 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 служити максимальний час відгуку програми на запит користувача, мінімальний час безперебійної роботи системи, наявність зручного інтерфейсу користувача тощо.

Функціональні та нефункціональні вимоги до програмної системи, що розробляється відповідно до будь-якої теми курсового проєкту з переліку рекомендованих тем, які повинні бути реалізовані у курсовому проєкті.

Функціональні вимоги

1. Створення файлу бази даних та 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 вводяться користувачем.
8. Видача користувачу попереджуючих та інформаційних повідомлень.

Нефункціональні вимоги

Мінімальні вимоги до консольного інтерфейсу користувача:

1. Для забезпечення наявності діалогу користувача з програмним продуктом елементи інтерфейсу користувача повинні мати докладне головне меню, яке містить посилання на основні завдання, що виконуються в проекті.
2. Кожен пункт головного меню може мати меню другого рівня, яке докладно розкриває його зміст.
3. Відображення даних і результатів обчислень необхідно формувати в наочному вигляді з необхідними коментарями.

4. Дані, що зберігаються у файловій базі даних та результати запитів до неї, повинні відображатися втабличному вигляді без зміщень в рядках і стовпцях.

Мінімальні вимоги до графічного інтерфейсу користувача:

1. Елементи інтерфейсу користувача мають супроводжуватися допоміжними текстовими мітками або мати заголовки, які виконуються українською мовою.

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

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

4. Наявність діалогових вікон, за допомогою яких користувач має можливість 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 відображенняпопереджуючих та інформаційних повідомлень, що можуть з'являтися в ході виконання програми.

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

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

Вимоги до архітектури програми:

1. Використання не менше одної структури даних із стандартної бібліотеки колекцій.

2. Використання файлових потоків введення-виведення даних.

3. Використання механізму винятків для обробки 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 повинен 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

При узгодженні з керівником курсового проєкту, деякі із цих вимог можуть бути змінені.

У випадку, якщо здобувач вищої освіти обрав тему курсового проєкту самостійно, то вимоги до програми обговорюються з керівником та можуть відрізнятися від наведених вище.

У підрозділі 1.3 "Математичний опис задачі" має бути наведена математична формалізація задачі, тобто існуючі співвідношення між величинами. Водночас залежно від специфіки розв'язуваної задачі можутьбути використані різні розділи математики та інших дисциплін. Таким чином формується математична модель явища з певною точністю, припущеннями і обмеженнями.

Математична модель повинна задовольняти принаймні двом вимогам: реалістичності і реалізованості.

Під реалістичністю розуміється 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 з використанням обмежених ресурсів.

У підрозділі 1.3 необхідно навести математичні формули або логічні співвідношення, які виражають залежність показників, що розраховуютьсявід вхідних даних та необхідні пояснення.

Розділ 2.Програмна реалізація

Метою розділу 2 є описархітектури 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 розроблюється. Це може бути крупна колекція з безлічі компонентів програмного забезпечення, один застосунок або частина застосунку.

Для уявлення статичної структури моделі програмної системи призначена UML-діаграма 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 даному пункті пояснювальної записки необхідно навести:

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

2. Посилання на лістинг програми з вихідним кодом, що відповідає класам, які реалізують основну бізнес-логіку програмної системи. Лістинг програми повинен знаходитися в одному з додатків до пояснювальної записки.

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

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

Завдання тестування – визначення умов, за яких проявляються 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 тестується як на відповідність вимогам, так і на відсутність проблемних ділянок програмного коду, які можуть викликати відмови й 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 в цілому як єдиного об'єкта. Вхідною інформацією для проведення системного тестування є два класи вимог: функціональні й нефункціональні.

Системне тестування проводиться в кілька етапів, на кожному з яких використовується один з видів системного тестування.

Важливим видом системного тестування є функціональне тестування. Цей вид системного тестування призначений для підтвердження того, що вся система в цілому поводиться відповідно до очікувань користувача, формалізованих у вигляді системних вимог.

У ході функціонального тестування перевіряються всі функції системи з погляду її користувачів(як людей, так і інших програмних систем). Також необхідно 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 виконується з необхідною якістю й всі аспекти поводження системи протестовані.

Перелік необхідної документації:

1. Тест-вимоги.
2. Тест-план.
3. Звіт протестування.

Тест-вимоги розробляються на підставі системних і функціональнихвимог до застосунку. У них докладно описується, які аспекти поводження системи повинні бути протестовані, щоб упевнитися в її коректному функціонуванні, і на підставі якого зовнішнього ефекту можна переконатися, що функціональність, яка перевіряється, реалізована правильно.

Тест-вимоги повинні бути достатніми для побудови тест-плану перевірки програмної системи без ознайомлення з її програмним кодом. Структура тест-вимог повинна додержуватися структури функціональних вимог до системи. Як правило, одній системній або функціональній вимозі відповідає мінімум одна тест-вимога.

Для кожної тест-вимоги повинна існувати можливість перевірки – виконується ця вимога в реалізованій системі чи ні. На підставі тест-вимог створюються тест-план – документ, що містить докладний покроковий опис того, як повинні бути протестовані тест-вимоги.

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

.Для кожного тестового прикладу однозначно визначається критерій успішного проходження, за допомогою якого можна судити про відповідність поводження системи заданому.

Структура тест-плану повинна відповідати структурі тест-вимог.

Кожний пункт тест-плану повинен містити:

1. Посилання на вимогу(и), що перевіряється 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 створюється звіт про виконання тестування. Він є основним джерелом для висновку про ступінь відповідності протестованої системи вимогам. Такий звіт як мінімум повинен містити інформацію про кожний виконаний тестовий приклад і результат його виконання (успіх або невдача).

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

Наприкінці опису кожного тестового прикладу додається інформація про те, чи пройдений тестовий приклад у цілому. Наприкінці всього тест-плану, сполученого зі звітом,міститься графа "Тестових прикладів пройдене/усього", у яку заноситься число пройдених тестових прикладів і загальна їхня кількість.

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

У цьому пункті пояснювальної записки має знаходитися опис процедур функціонального тестування та їхніх результатів.

Для опису процедури тестування повинні бути складені наступні документи:

1. Тест-вимоги.

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

У звіті про проведення тестування вказуються як позитивні так і негативні результати виконання окремих тестів. Однак загальний результат тестування застосунку повинен бути позитивним, бо в протилежному випадку він не відповідає певним вимогам. Для цього в разі необхідності проводяться додаткові заходи щодо виправлення помилок та тестування. Вони також повинні бути описані у цьому пункті пояснювальної записки.

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

У **підрозділі 2.3 "Розгортання програмного продукту"** необхідно навести опис вимог до апаратних та 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 К.

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

У пункті 2.4.1 "Призначення програмного продукту" вказуються відомості про його призначення та інформація, достатня для розуміння функцій програмного продукту.

У пункті 2.4.2 "Використання програмного продукту" має бути вказана 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.

Упункті 2.4.3 "Повідомлення користувачеві" наводяться екранні форми повідомлень, що можуть з'являтися в ході виконання програми,та опис їх 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, якібуло цитовано, згадано або розглянуто в роботі. Список використаних джерел оформлюється згідно з ДСТУ 8302:2015 «Бібліографічне посилання. Загальні вимоги та правила складання» [1].

Список використаних джерел подається мовою оригіналу, розміщується в алфавітному порядку прізвищ перших авторів або назв, нумерується в порядку їх зростання. Нумерація безперервна.

Роботи одного автора розташовані за алфавітом назв або в хронології їх написання. Алфавітний список розташований за алфавітом у такій послідовності:

література зведеного кириличного алфавіту, для джерел і мовами з кириличною графікою (українська, болгарська та ін.);

література в латинському алфавіті;

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

Список літературних джерел обов’язково повинен містити прізвище та ініціали автора, повну назву джерела, місто видавництва, видавництво та рік видання, кількість сторінок чи посилання на сторінки тощо. Загальний обсяг книги в сторінках вказується, якщо посилання на неї проводиться повністю, сторінки (від... до) відмічаються, якщо посилання належать до окремої частини літературного джерела.

У **додатках** вміщують матеріал, який є необхідним для повноти пояснювальної записки, але не може бути послідовно розміщений в її 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 та розрахунки, текст допоміжного характеру та інші матеріали можуть бути оформлені у вигляді додатків.

Кожний додаток повинен починатись з нової сторінки і мати заголовок (назву) надрукований малими літерами з першої великої та з указівкою посередині сторінки слова “Додаток” та його позначення. Якщо додатки позначаються прописними (великими) літерами українського алфавіту, починаючи з літери А за винятком літер Ґ, Є, З, І, Й, О, Ч, Ь.

Якщо додатків багато, їх оформлюють у вигляді окремої частини, на титульному аркуші якої повинно бути написано великими літерами «ДОДАТКИ» або «ДОДАТОК» і його назву (за наявності).

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

Курсовий проєкт складається з пояснювальної записки та інших обов’язкових матеріалів (схеми, діаграми, графіки залежностей, таблиці, малюнки, лістинги програм тощо), що розробляються відповідно до завдання.

Обсяг пояснювальної записки – 20 – 40 друкованих сторінок формату А4 (без додатків).

Пояснювальна записка виконується у друкований спосіб на аркушах формату А4. Її текст має бути виконаний з використанням шрифту TimesNewRoman (кегль 14), з міжрядковимінтервалом 1,5.

Поля сторінок пояснювальної записки – 25 мм від лівого краю аркушу, 10 мм – від правого краю аркушу, по 20 мм від верхнього та нижнього країв аркушу.

Абзацний відступ повинен бути однаковим впродовж усього тексту та дорівнювати 1,25 см.

Вирівнювання основного тексту проводиться «за шириною».

Загальними вимогами до тексту пояснювальної записки 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 пояснювальної записки необхідно дотримуватись єдиної термінології. Вона не має бути перевантажена малоінформативним матеріалом, описом загальновідомих даних, виведенням формул тощо. Необхідно посилатися на джерела інформації.

Текст пояснювальної записки не слід викладати від першої особи, переважніше безособова форма (наприклад, «обчислюється», «знаходимо») за всім текстом у визначеному відмінку й часі.

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 встановлених правилами орфографії та нормативними документами.

У роботі не може бути повторень, надмірного цитування. Цитати мають бути короткими й органічно пов'язуватись з основним текстом. На використані у роботі цифрові дані, цитати, таблиці мають бути посилання із зазначенням порядкового номера відповідного джерела у списку літератури. По тексту цей порядковий номер полягає у квадратні дужки (наприклад: *…подані у роботі [8] результати використані у курсовому проєкті для…*).

Оформлення ілюстрацій, таблиць, формул

Ілюстрації слід розміщувати у світі безпосередньо після тексту, де вони згадуються вперше, або на наступній сторінці. На всі ілюстрації мають бути посилання. Ілюстрації можуть мати назву, яку розміщують під ілюстрацією. Ілюстрація позначається словом "Рисунок номер – назва", яке разом з назвою ілюстрації розміщують після пояснювальних даних.

Ілюстрації слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу.

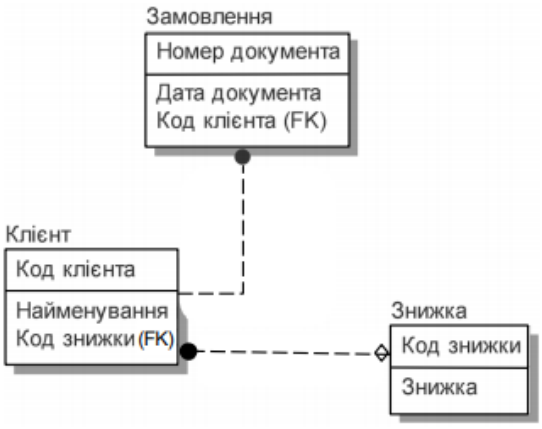

Рисунок 1– Приклад оформлення рисунку

Таблиця розташовується безпосередньо після тексту, у якому вона згадується 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 номер - найменування” вказують один раз зліва над першою частиною таблиці, над іншими частинами пишуть: “Продовження таблиці номер” з зазначенням номера таблиці.

Таблиця 1 – Приклад таблиці

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

Формули розташовують безпосередньо після тексту, в якому вони згадуються, посередині сторінки. Вище і нижче кожної формули повинно бути залишено не менше одного вільного рядка.

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

Пояснення значень символів, що входять до формули слід наводити безпосередньо під формулою у тій послідовності, в якій вони наведені. Пояснення значення слід давати з нового рядка. Перший рядок пояснення починають з абзацу словом “де” без двокрапки.

Наприклад,

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

де $f(x)$ – це цільова функція; і т.п.

2. 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 антіплагіат і рецензування. Перевірка курсових проєктів здійснюється протягом 10 днів з моменту їх надання.

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

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

Якщо в результаті перевірки виявлені істотні помилки, неповний обсяг, низька якість роботи чи проєкт не пройшов перевірку на антіплагіат, він повертається здобувачу вищої освіти для доопрацювання. Зауваження керівника в письмовому виглядіподають студенту. На титульному листі керівник ставить підпис «Доробити» чи «До захисту» і дата перевірки.

У разі відповідності курсового проєкту вимогам цих методичних рекомендацій керівник ставить на титульному листі «До захисту». Допущений до захисту курсовий проєкт захищається здобувачем вищої освіти у присутності здобувачів вищої освіти академічної групи. Не допущена до захисту робота передається здобувачу вищої освіти на доопрацювання, і процедура подання на перевірку та рецензування повторюється.

Захист курсових проєктів може проходити в такій формі: автору проєкту дається до 10 хвилин для доповіді основних положень, після чого йому задаються питання стосовно змісту роботи. Оцінювання якості виконання і захисту здобувачами вищої освіти курсово гопроєкту здійснюється за 100-бальною шкалою. Підсумкову оцінку визначають

викладачі, які приймають захист курсових проєктів. Об'єктами оцінювання є три складові: зміст, оформлення та захист курсовогопроєкту (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 у курсовому проєкті (0 – 50 балів):

- ступінь розкриття теоретичних аспектів проблеми, обраної для досліджень;
- наявність практичного висвітлення проблематики курсового проєкту;
- наочність та якість ілюстративного матеріалу;
- рівень обґрунтування запропонованих рішень;
- відповідність отриманих результатів поставленим цілям і 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):

- вміння чітко, зрозуміти та стисло викладати основні задачі проведеного дослідження;
- повнота, глибина, обґрунтованість відповідей на запитання членів комісії за змістом;
- ґрунтовність висновків і рекомендацій щодо практичного використання розробленого програмного засобу.

Етапи виконання курсового проєкту

Курсовий проєкт з об’єктно-орієнтованого програмування виконується у IV семестрі, завдання видається на початку семестру.

Вибір теми і її затвердження

Здобувач вищої освіти може обрати будь-яку запропоновану в цих методичних рекомендаціях тему або узгодити з керівником власну тему. Обрані здобувачами вищої освіти теми затверджуються на засіданні кафедри інформатики та комп’ютерної техніки, після чого здобувач вищої освіти не може змінити тему.

Складання плану і його реалізація

Після отримання теми здобувач вищої освіти повинен протягом трьох тижнів надати керівникові перший розділ, у якому потрібно навести специфікацію проєкту. Після уточнення з викладачем плану теми, затверджується розклад консультацій з розрахунку 2 години на курсовийпроєкт. Здобувач вищої освітимає право на консультацію за курсовим проєктом у зазначений час, в інший час тільки за узгодженням з викладачем.

Рекомендована література

1. ДСТУ 8302:2015 "Бібліографічне посилання. Загальні положення та правила складання".– Київ:Держстандарт України, 2007. – 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.

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

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

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.

Інформаційніресурси

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.

11. NET DesignPatterns [Electronicresource]. – Access mode : https://www.dofactory.com/net/design-patterns.

Додатки

Додаток А

Тематика курсових проєктів з об’єктно-орієнтованого програмування

Дозволяється самостійний вибір здобувачами вищої освіти теми курсового проєкту за узгодженістю 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 по 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 магазина 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 базоюданих 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 даату.

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 розкладу 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 продажі книг в 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 витрат 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 вищого закладу 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 нарахуванні 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 закази клієнтів.

47. Розроблення програмного продукту для роботи з файловою базою даних про публікації студентів.

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 курсу,_____групи, спеціальності126 "Інформаційні системи та технології"

_____
(П.І.Б.)

Керівник: вчений ступінь, вчене звання, посада

_____
(П.І.Б.)

Національна шкала_____
Кількість балів:___ Оцінка:ECTS__

Члени комісії _____ _____
(підпис) (П.І.Б.)

_____ _____
(підпис) (П.І.Б.)

_____ _____
(підпис) (П.І.Б.)

Харків – 2024

Додаток В

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: Програмування
студенту 2-го курсу групи 6.04.126.010.22.01
ІвановуІвану 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 пояснювальної записки: Вступ. Специфікація проєкту. Програмна документація. Висновки. Додатки.

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

Закінчення додатка В

Таблиця В.1

Календарний план виконання курсового проєкту

Здобувач вищої освіти _____ Іван ІВАНОВ

Керівник _____Олексій ГОРОХОВАТСЬКИЙ

Додаток Г

Зразок оформлення реферату

РЕФЕРАТ

Пояснювальна записка до курсового проєкту: 32 с., 20 рис., 3 табл.,6 джерел. Об’єктом проєктування є програмний модуль «Розроблення програмної системи для автоматизації роботи з файловою базою даних»

Метою роботи є автоматизація процесу обробки і подання інформації.

Метод проєктування – використання програмних систем ARIS Toolset, IBM RationalRoseEnterprise, Ramus, NetBeans IDE, WampServer, Firebase.

В результаті виконання курсового проєкту було обґрунтовано вибір програмного забезпечення для моделювання інформаційного процесу та створення модуля автоматизації. На основі алгоритму моделювання інформаційної системи реалізовано програмний продукт автоматизації процесу обробки і подання інформації. Програмний продукт може бути використаний будь-якими закладами.

ІНФОРМАЦІЙНА СИСТЕМА, ДОДАТОК, МОКАП, МАКЕТ, ГЕНЕРАЦІЯ КОДУ, БІЗНЕС-ПРОЦЕС, АВТОМАТИЗАЦІЯ.

Додаток Д

Зразок оформлення змісту пояснювальної записки

ЗМІСТ

Додаток Ж

Методичні рекомендації щодо розроблення інтерфейсу користувача

Розробка інтерфейсу користувача (UI) - це процес вдосконалення презентації та інтерактивності додатків, який фокусується на зовнішньому вигляді програми та її взаємодії з користувачами.

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

Розробка взаємодії з користувачем (UX) - це процедура поліпшення загального досвіду користувачів при взаємодії з додатком або веб-сайтом для досягнення своєї основної мети - забезпечити максимальне задоволення споживачів, щоб користувачі знаходили в продукті додаткову цінність.

UX підхід включає в себе деякий повторювальний цикл під час розробки додатку, який складається з:

дослідження (початкове дослідження користувачів, для розуміння цільової аудиторії та їх можливості, обмеження, цілі та очікування);

дизайну (при створенні якого використовується статистика досліджень користувачів, щоб допомогти генерувати ідеї та початковий прототип для реалізації концепцій);

оцінки (фіксування відгуків користувачів протягом розробкипроєкту).

Основними властивостями, яким повинні задовольняти інтерфейси, є такі:

Адаптованість означає, що інтерфейс повинен бути:

– сумісним з потребами та можливостями користувача;

– забезпечувати простоту переходу від виконання однієї функції до іншої;

– забезпечувати користувача на високому рівні вказівками стосовно його можливих дій, а також генерувати належний зворотний зв’язок на його запити;

– надавати користувачу можливість відчувати себе повноправним керівником ситуації при розв’язанні всіх типів задач, тобто, забезпечувати його всією необхідною інформацією; користувач повинен бути впевненим, що він сам розв’язує поставлену задачу;

– забезпечувати користувача різними, взаємно доповнюючими формами представлення результатів в залежності від типу запиту або від характеру отриманого рішення;

– враховувати особливості користувачів різних рівнів;

Достатність інтерфейса означає:

– допустимі запити користувача повинні бути чіткими і однозначними для користувачів всіх рівнів, а також для прикладних задач всіх типів;

– реакція системи на всі типи запитів також повинна бути однозначною і зрозумілою і, по можливості, простою.

Дружність інтерфейсу. Це максимальна простота його використання і готовність в повній мірі задовольнити запити користувача при розв’язанні визначеного класу задач.

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

Тобто інтерфейс повинен буди максимально простим навіть у випадку, коли розв’язується дуже складна задача.

Основні правила створення інтерфейсу користувача

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

Правило ефективності. Система не повинна перешкоджати ефективній роботі досвідчених користувачів, що працюють з нею довгий час. 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 повинна сприяти безперервному росту знань, умінь і навичок користувача і пристосовуватися до його досвіду, що міняється. Погані результати приносить надання тільки базових можливостей або залишення початкуючого користувача наодинці із складним інтерфейсом, яким упевнено користуються експерти. Порушення безперервності при переході від одного набору можливостей до іншого також приносить незручності, оскільки користувач вимушений розбиратися з доданими можливостями в новому контексті.

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

Принципи розробки 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 повинні виглядати схоже, а несхожі – розрізнятися.

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

Принцип видимості. Усі функції і дані, що необхідні для розв'язання певного завдання, мають бути видні, коли користувач намагається її вирішити.

Принцип зворотного зв'язку. Користувач повинен отримувати повідомлення про дії системи і про важливі події всередині неї. Повідомлення мають 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 за рахунок можливості відміни і повтору дій і за рахунок розумної інтерпретації будь-яких розумних дій користувача і введених їм даних. По можливості слід уникати взаємодії (модальних діалогів) заснованої на обмеженні свободи користувача.

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

Взаємодія між користувачем і комп'ютером

Людиномашинний 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. Існує ряд стилів взаємодій, які підрозділяються на два види.

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

2. Безпосереднє маніпулювання.

Таким чином, є ряд способів, якими користувач міг би зв'язуватися з комп'ютером:

мови команд – користувач управляє системою, вводячи відповідні команди в текстовому режимі;

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

форми – користувач заповнює форми або поля діалогу, вводячи 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 притягнути увагу до найбільш важливих одиниць інформації. Основна ж мета в тому, щоб мінімізувати загальну інформацію на екрані і представити тільки те, що є необхідним для користувача.

Основні принципи створення меню

При проєктуванніменю додатка необхідно прийняти найкращий спосіб відображення меню, щоб воно було зрозумілим і легким у використанні. Зазвичай команди меню впорядковані деяким ієрархічним способом. Основна проблема полягає в тому, щоб правильно розподілити різні пункти меню по різних рівнях і правильно їх згрупувати. Принципи проектування меню:

структура меню повинна відповідати структурі завдання, що розв'язується системою; організація меню повинна 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 вибирається відповідно до угоди, частоти і порядку використання, а також залежно від потреб завдання або користувача;

вибір пунктів меню має бути забезпечений декількома способами – за допомогою клавіатури, за допомогою миші і 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 додатку.

При розробленні форм додатку необхідно дотримуватись наступних принципів проєктування:

форма проєктується для зручнішого, зрозумілого і швидкого рішення поставленої задачі. Якщо форма переноситься з паперової

форми, то пересування по суміжних полях не повинне викликати утруднень у користувача;

розміщення інформаційних одиниць на просторі форми повинне 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 необхідно відділяти пропусками, рядками, колірними або іншими візуальними засобами;

взаємозалежні або пов’язані елементи повинні відображатися в одній формі.

При розробці форм необхідно 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.

При проєктуванні форм необхідно прагнути до використання обмеженого набору кольорів і приділяти увагу їх правильному поєднанню. Для фону форми обираються нейтральні кольори (світло-сірі). Колір не повинен використовуватися 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 екрану слід зорово об’єднувати в групи, заголовки яких коротко і чітко пояснюють їх призначення. Кожне вікно повинне мати деяку центральну тему, яка підпорядковується його композиції.

Користувач повинен розуміти, для чого призначено це вікно і що в нім 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 значимими і загальноприйнятими. Не слід об’єднувати поля введення чисел і символів, оскільки числові і алфавітні клавіші знаходяться незручно один відносно одного на клавіатурі.

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

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

Організація системи навігації і 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

Існує ряд навігаційних засобів і прийомів, які допомагають користувачеві орієнтуватися в системі. Вони включають використання заголовків сторінок для кожного екрану, номерів сторінок, рядків і стовпців, відображення поточного імені файлу вверху екрану. Тип системи навігації залежить від прийнятого стилю інтерфейсу. Для інтерфейсів мови команд існує дуже мало способів забезпечення повноцінної навігації.

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

Проєктування повідомлень. Повідомлення потрібні для спрямування дій користувача в потрібну сторону, підказок і попереджень при виконанні необхідних дій на шляху розв’язання задачі. Вони також включають підтвердження дій з боку користувача і підтвердження з боку системи, що завдання виконані успішно або з якихось причин не виконані.

Повідомлення можуть бути виведені у формі діалогу, екранних заставок і тому подібне. Повідомлення можуть запропонувати користувачеві:

вибрати із запропонованих альтернатив опцію або набір опцій;

ввести інформацію;

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

підтвердити фрагмент введеної інформації перед продовженням введення.

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

Запобігання, виявлення і виправлення помилок

Помилки користувача можуть бути засновані на неправильному розумінні дії або порядку дій або бути випадковими, неумисними, наприклад, друкарська помилка при введенні тексту.

Помилки другого виду можуть бути розділені ще на шість підвидів:

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 знаходиться в одному стані, а фактично – в іншому;

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

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

забезпечення хороших і інформативних повідомлень про помилки;

забезпечення нормальної діагностики системи, в процесі якої користувачеві пояснюється, в чому суть помилки, і вказуються шляхи її виправлення.

Основні принципи обробки помилок у формах введення

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

якщо помилка виявлена системою, бажано повернути курсор в поле з помилковими даними і яким-небудь чином виділити це поле;

виводити значимі 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 вище, чим рідше користувачі помиляються при роботі з цим інтерфейсом і чим менше часу і зусиль вимагається для подолання наслідків вже зроблених помилок.

Додаток 3

Приклад UML-діаграми класів

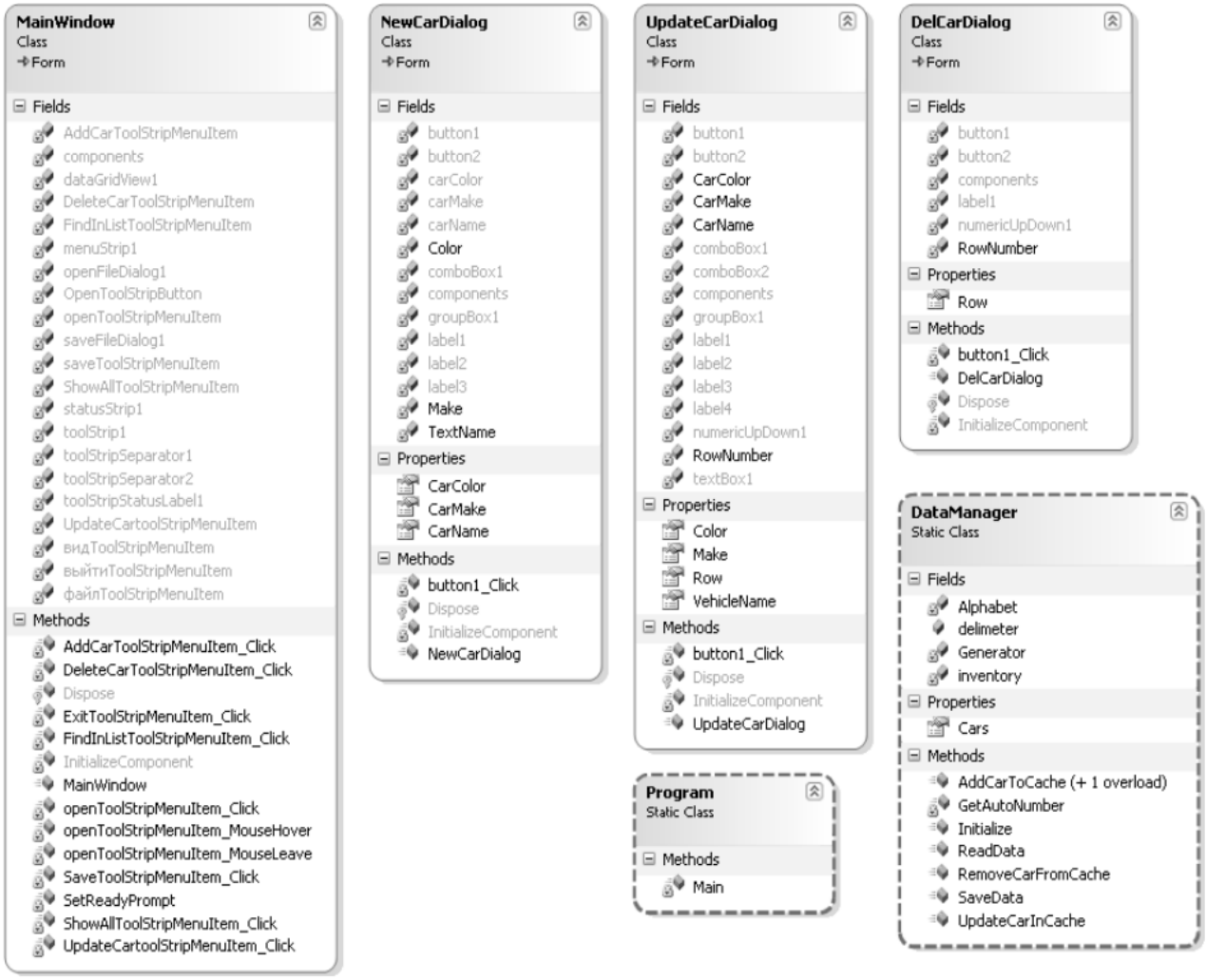

Рис.2.1 – UML-діаграма класів

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 проновий автомобіль. Майже всі поля даного 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, що призначено для видалення даних про автомобіль.

Клас DataManager призначений 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 наступним чином.

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

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

Додаток І

Приклад тест-плану, сполученого зі звітом пропроведення 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 перевіряються: функціональна вимога №2.

Передумови для тесту: програмна система повинна бути запущена, а на диску комп'ютера має знаходитися файл із даними у визначеному форматі.

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

№ з/п	Крок сценарію	Очікуваний результат	Отриманий результат	Відмітка про проходження кроку сценарію (Так/Ні)
1	2	3	4	5
1	У меню "Файл" вибрати пункт "Відкрити"	Повинне з'явитися діалогове вікно від криття файлу	Діалогове вікно відкриття файлу з'являється	Так
2	У діалоговому вікні відкриття файлу вибрати ім'я файлуз даними визначеного формату та натиснути кнопку "ОК"	У головному вікні програми мають коректна відобразитися дані, що були завантажені з файлу	Дані,що були завантажені з файлу, коректно відображуються у головному вікні програми	Так

1	2	3	4	5
3	У діалоговому вікні відкриття файлу вибрати ім'я файлу з даними, у недопустимому форматіта натиснути кнопку "ОК	Повинне з'явитися діалогове вікно з повідомленням проте,що дані не можуть бути завантажені	З'являється діалогове вікно з повідомленням	Так

Відмітка про проходження тесту (пройдено/непройдено): пройдений.

Тестовий приклад:№ 2

Тестовий приклад:№ 3.

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

Додаток К

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. В каталозі MyApp створіть новий каталог, наприклад, dist.

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

4. Завантажте з веб-сайту 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(операційнасистемаWindows).

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

8. Для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою мишінафайлістart.bat (операційна система Windows).

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

методичні рекомендації до виконаннякурсовогопроєкту
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: **Тютюник**Ольга Олександрівна
ГороховатськийОлексій Володимирович
Парфьонов Юрій Едуардович

Відповідальний за видання С.Г. Удовенко

Редактор В.О. Дмитрієва

Коректор

План 2023 р. Поз. 103 ЕВ. Обсяг 50стор.

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