МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ПРОГРАМУВАННЯ

Методичні рекомендації до виконання комплексного курсового проекту

для студентів спеціальності 122 "Комп'ютерні науки" першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2018

УДК 004.42(07.034)
П78

Укладачі: Ю. Е. Парфьонов
В. М. Федорченко
О. В. Щербаков

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 5 від 21.12.2017 р.

Самостійне електронне текстове мережеве видання

Програмування : методичні рекомендації до виконання комплексного курсового проекту для студентів спеціальності 122 "Комп'ютерні науки" першого (бакалаврського) рівня [Електронний ресурс] / уклад. Ю. Е. Парфьонов, В. М. Федорченко, О. В. Щербаков. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 48 с.

П78

Викладено питання організації курсового проектування, наведено вимоги до структури комплексного курсового проекту та оформлення пояснювальної записки, методичні рекомендації до розроблення його структурних елементів.

Рекомендовано для студентів спеціальності 122 "Комп'ютерні науки" першого (бакалаврського) рівня.

УДК 004.42(07.034)

Вступ

Сучасні умови господарювання вимагають від фахівців з економічного управління всебічного використання новітніх інформаційних технологій. Широкі можливості комп'ютеризованих засобів щодо збору, обробки та видачі необхідної інформації здатні значно підвищити якість економічних розрахунків, зробити більш ефективним процес обґрунтування економічних рішень.

Але використання потужних комп'ютеризованих засобів неможливо без програмного забезпечення. Важливість галузі розроблення програмного забезпечення збільшується, оскільки тенденції розвитку комп'ютерної техніки свідчать про те, що, з одного боку, її складність та функціональні можливості швидко зростають, а з другого – це потрібує більш досконалих програмних засобів для задоволення потреб користувачів.

Для створення таких програмних систем відповідні фахівці мають володіти як концепціями алгоритмізації та програмування узагалі, 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.

Необхідним елементом успішного засвоєння навчального матеріалу базових навчальних дисциплін "Програмування", "Алгоритми та структури даних", "Основи об'єктно-орієнтованого програмування" є самостійна робота студентів із технічною літературою, сучасними мовами, технологіями та середовищами програмування. Підвищенню її ефективності покликане сприяти курсове проектування.

Комплексний курсовий проект є важливим етапом підготовки студентів. Він спрямований на більш глибоке осмислення і закріплення теоретичних знань, отриманих під час вивчення зазначених дисциплін, та на удосконалення практичних навичок щодо розроблення програмного забезпечення та необхідної документації.

Ці методичні рекомендації встановлюють єдині правила та порядок виконання, оформлення і захисту комплексного курсового проекту. Вони призначені для допомоги студентам спеціальності 122 "Комп'ютерні науки" у виконанні комплексного курсового проекту.

1. Мета й завдання курсового проектування

Метою курсового проектування є закріплення та поглиблення знань з дисциплін "Алгоритми та структури даних", "Програмування", "Основи об'єктно-орієнтованого програмування".

Робота над комплексним курсовим проектом сприяє систематизації, поглибленню й закріпленню знань, отриманих студентами під час вивчення даних дисциплін. У процесі курсового проектування студенти розвивають навички практичного застосування отриманих знань під час створення комплексного застосунку з використанням сучасних інструментальних засобів розробки. Водночас студент повинен показати вміння користуватися спеціальною літературою, державними стандартами, довідниками та іншими матеріалами з інформаційних технологій.

Під час розроблення комплексного курсового проекту студент має показати знання:

предметної області відповідно до постановки завдання;

основних концепцій алгоритмізації та програмування;

об'єктно-орієнтованого підходу до програмування;

сучасного стану та перспектив розвитку технологій програмування;

сучасних інструментальних засобів, призначених для розроблення програмних систем.

Студент повинен уміти:

виконувати аналіз постановки завдання;

виконувати декомпозицію програмної системи на підсистеми;

розробляти загальну архітектуру програмної системи;

здійснювати деталізацію загальної архітектури програмної системи;

розробляти програмну реалізацію системи певною мовою програмування;

використовувати стандартні бібліотеки мови програмування або програмної платформи;

документувати вихідний код програми;

використовувати засоби розроблення програм та отримання довідкової інформації;

розробляти та оформлювати необхідну документацію.

Робота над комплексним курсовим проектом певною мірою визначає загальнотеоретичну та спеціальну підготовку студента і в остаточному

підсумку готує його до майбутнього виконання більш складного й завершального етапу навчального процесу – дипломного проектування. Студент має розглядати роботу над комплексним курсовим проектом як своєрідну репетицію дипломного проектування.

2. Організація курсового проектування

Якісне виконання курсового проекту вимагає чіткої організації роботи студента з моменту вибору теми проекту й до його захисту.

Керівництво курсовим проектуванням здійснюється викладачами кафедри інформаційних систем. Керівник курсового проекту рекомендує студенту основну літературу, орієнтує його на розроблення необхідних проектних рішень.

Студенту може бути призначена тема курсового проекту з переліку рекомендованих тем (додаток А). Також йому надається право самостійного вибору теми проекту. Якщо тема пропонується студентом, то вона має бути обговорена й погоджена з керівником курсового проекту.

Після затвердження вибраної теми студентові видається завдання на курсове проектування. У завданні міститься тема курсового проекту, вихідні дані до проекту, зміст пояснювальної записки, строки початку й закінчення роботи над курсовим проектом, обумовлені графіком навчального процесу, план-графік виконання етапів курсового проектування. Зразок завдання на курсовий проект наведено в додатку Б.

Після вивчення літературних джерел студент складає попередній план виконання курсового проекту, обговорює його з керівником. У процесі обговорення уточняються вихідні дані для проектування й строки, що регламентують роботу студента над проектом. Після цього студент складає уточнений план роботи над проектом, узгоджує його з керівником та приступає до проектування. У процесі виконання курсового проекту студент повинен регулярно відвідувати консультації керівника, подавати йому на перевірку робочі матеріали відповідно до плану-графіка виконання етапів курсового проектування.

Матеріали виконаного курсового проекту: пояснювальну записку в печатному та електронному вигляді, вихідний код програмного застосунку, програму-інсталятор, презентацію доповіді в електронному вигляді студент має здати на перевірку керівникові не пізніше визначеного строку.

Захист курсових проектів організовується кафедрою інформаційних систем за затвердженим графіком.

Під час захисту студент коротко доповідає про поставлене йому завдання, проектні рішення, що були прийняти, одержані результати, відповідає на питання. Доповідь має супроводжуватися демонстрацією електронної презентації, яка має містити як мінімум таки слайди:

1. Титульний слайд з вихідними даними щодо курсового проекту: тема проекту, П. І. Б. виконавця, П. І. Б. керівника тощо.
2. Зміст презентації.
3. Актуальність теми та мета курсового проекту.
4. Коротка постановка завдання.
5. Математичний опис задачі (декілька слайдів).
6. UML-діаграма класів, що реалізують основну бізнес-логіку програмної системи.
7. Використані інструментальні засоби та технології.
8. Висновки за результатами виконання курсового проекту.
9. Заключний слайд.

3. Структура та обсяг курсового проекту

Курсовий проект складається з пояснювальної записки та інших матеріалів, зокрема програмного застосунку, що розробляється відповідно до завдання.

Обсяг пояснювальної записки – приблизно 29 – 42 друкованих сторінок формату А4 (без додатків). Матеріали пояснювальної записки мають бути зброшурованими.

У табл. 1 наведено структуру пояснювальної записки курсового проекту.

4. Методичні рекомендації до розроблення структурних елементів пояснювальної записки курсового проекту

Титульний аркуш є першою сторінкою пояснювальної записки. Він містить дані, які подають у такій послідовності:

відомості про виконавця роботи;

повна назва документа;

підписи відповідальних осіб, включаючи керівника роботи;

рік складення пояснювальної записки;

Приклад титульного аркуша наведено в додатку В.

Реферат – це короткий виклад змісту пояснювальної записки, що містить основні фактичні відомості і висновки, необхідні для початкового ознайомлення з нею.

Реферат має бути стислим, інформативним і містити відомості, які дозволяють прийняти рішення про доцільність читання пояснювальної записки.

Таблиця 1

Структура пояснювальної записки курсового проекту

Структурні елементи пояснювальної записки	Кількість сторінок
Титульний аркуш	1
Завдання на курсове проектування	2 (на одному аркуші)
Реферат	1
Зміст	1
Вступ	1
1 Специфікація проекту	3 – 7
1.1 Постановка завдання	1 – 2
1.2 Вимоги до програмного забезпечення	1 – 2
1.3 Математичний опис задачі	1 – 3
2 Програмна документація	18 – 27
2.1 Архітектура програмної системи	2 – 4
2.2 Тестування програмної системи	6 – 8
2.3 Розгортання програмного продукту	2 – 3
2.4 Керівництво користувача	8 – 12
2.4.1 Призначення програмного продукту	1
2.4.2 Використання програмного продукту	6 – 8
2.4.3 Повідомлення користувачеві	1 – 3
Висновки	1
Список використаних джерел	1
Додатки	

Реферат повинен містити:

текст реферату;

перелік ключових слів.

Текст реферату повинен відбивати подану в пояснювальній записці інформацію в такій послідовності:

об'єкт дослідження або розроблення;

мета роботи;

методи дослідження та апаратура;

результати та їх новизна;

основні технологічні й техніко-експлуатаційні характеристики та показники;

взаємозв'язок з іншими роботами;

рекомендації щодо використання результатів курсового проекту;

галузь застосування;

значущість роботи та висновки;

прогнозні припущення про розвиток об'єкта дослідження або розроблення.

Реферат належить виконувати обсягом не більш, як 500 слів.

Ключові слова призначені для розкриття сутності проекту та для розповсюдження інформації про розробку. Їх розміщують після тексту реферату. Перелік ключових слів містить від 5 до 15 слів (словосполучень), надрукованих великими літерами в називному відмінку в рядок через коми.

Зразок оформлення реферату наведено в додатку Г.

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

Зразок оформлення змісту пояснювальної записки наведено в додатку Д.

У вступі слід зазначити важливість використання інформаційних систем у сучасних умовах, роль інформаційних систем у предметній галузі, що розглядається в курсовому проекті, роль засобів збереження даних в інформаційних системах, мету та завдання курсового проекту, відомості щодо розробленої програми (призначення, яке вона дозволяє автоматизувати,

технології та мова програмування, які були використані під час розроблення програми), заключна частина.

Перший розділ "Специфікація проекту" містить постановку завдання, вимоги до програмного забезпечення та математичний опис задачі.

У підрозділі 1.1 "Постановка завдання" наводиться постановка завдання на розроблення програмного продукту відповідно до варіанта, який був виданий студенту. Також необхідно вказати, що розробка виконується на підставі завдання на виконання курсового проекту, затвердженого кафедрою інформаційних систем.

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

Функціональні вимоги явно описують, що повинна робити програмна система і які перетворення вхідних даних виконувати.

Нефункціональні вимоги визначають властивості програмної системи, прямо не пов'язані з її функціональністю. Прикладом таких властивостей може служити максимальний час відгуку системи на запит користувача, мінімальний час безперебійної роботи системи, наявність графічного інтерфейсу користувача тощо.

Функціональні та нефункціональні вимоги до програмної системи, що розробляється відповідно до будь-якої теми курсового проекту з переліку рекомендованих тем, наведено нижче. У обґрунтованих випадках деякі із цих вимог можуть бути змінені по узгодженню з керівником курсового проекту.

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 графічного інтерфейсу користувача:

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

Якщо студент обрав тему курсового проекту 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 можуть бути 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 цьому підрозділі необхідно навести математичні формули або логічні співвідношення, які виражають залежність показників, що розраховуються, від вхідних даних та необхідні пояснення.

Другий розділ "Програмна документація" складається із чотирьох підрозділів: 2.1 – 2.4. Цей розділ містить 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 розробляється. 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 рідка 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 про тестування.

Тест-вимоги розробляються на підставі системних і функціональних вимог 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 функціональності системи.

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 И.

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 Java SE наведено в додатку К.

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" 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.

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

Приклад керівництва користувача (у скороченні) наведено в додатку Л.

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. Джерела можна розміщувати в списку одним із таких способів: в порядку появи посилань у тексті, в алфавітному порядку прізвищ перших авторів або заголовків. Вимоги до оформлення списку використаних джерел наведено в роботі [1].

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.

5. Вимоги до оформлення пояснювальної записки курсового проекту

Важливе значення під час роботи над курсовим проектом має його оформлення, до якого висуваються певні вимоги.

Під час оформлення тексту пояснювальної записки слід керуватися державним стандартом України ДСТУ 3008-95 "Документація. Звіти у сфері науки і техніки. Структура і правила оформлення" [3]. Далі наведені загальні вимоги до оформлення пояснювальної записки курсового проекту.

Матеріали пояснювальної записки друкуються на аркушах формату А4. Їх текст повинен відповідати правилам граматики й стилістики.

Текст роботи необхідно форматувати, залишаючи на аркушах поля таких розмірів: ліве – 30 мм, праве – 10 мм, верхнє – 20 мм, нижнє – 20 мм.

Текст документа повинен бути виконаний із використанням шрифту Times New Roman (розмір 14), з міжрядковим інтервалом 1,2 (37 рядків на сторінці). Найменшим розміром шрифту може буту розмір 10 (його можна використовувати в разі подання вихідного тексту програм). Шрифт друку повинен бути чітким, текст – чорного кольору середньої жирності. Кольоровий друк дозволяється використовувати лише для рисунків (екранні форми, діаграми тощо). Щільність тексту повинна бути однаковою. Вирівнювання основного тексту проводиться "за шириною" сторінки.

Весь текст пояснювальної записки, включаючи назви її структурних елементів, виконується шрифтом однакової жирності. Не дозволяється використання курсиву та підкреслення.

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

Друкарські помилки, описки і графічні неточності, які виявилися в процесі виконання документа, можна виправляти підчищенням або

зафарбовуванням білою фарбою і нанесенням на місці виправленого тексту. Допускається наявність не більше двох виправлень на одній сторінці.

Під час скорочення слів і словосполучень потрібно спочатку навести повну назву, а після цього в дужках – її скорочення.

Кожний структурний елемент пояснювальної записки (крім підрозділів, пунктів, підпунктів) повинен починатися з нової сторінки.

Назви елементів "РЕФЕРАТ", "ЗМІСТ", "ВСТУП", "ВИСНОВКИ", "СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ", "ДОДАТКИ" розміщують симетрично до тексту (від центру), без абзацного відступу, не нумерують (не можна друкувати "1. ВСТУП" або "РОЗДІЛ 6. ВИСНОВКИ"), виконують великими буквами без крапки наприкінці та не підкреслюють.

Заголовки підрозділів, пунктів і підпунктів слід починати з абзацного відступу і друкувати маленькими літерами, крім першої великої, не підкреслюючи, без крапки в кінці.

Переноси в середині слова в заголовках не допускаються.

Не дозволяється розміщати заголовки й підзаголовки в нижній частині сторінки, якщо на ній тільки один рядок наступного тексту.

Відстань між заголовком і подальшим чи попереднім текстом має бути два рядки. Відстань між основами рядків заголовку, а також між двома заголовками приймають такою, як у тексті.

Розділи, підрозділи, пункти, підпункти треба нумерувати арабськими цифрами. Розділи мають порядкову нумерацію, наприклад: 1, 2, 3. Підрозділи повинні мати порядкову нумерацію в межах розділу. Номер підрозділу містить номер розділу й порядковий номер підрозділу, які розділяються крапкою, наприклад: 1.1, 1.2, 1.3. Номер пункту містить номер розділу, підрозділу, порядковий номер параграфа, які розділяються крапкою, наприклад: 1.1.1, 1.1.2, 1.1.3.

Сторінки пояснювальної записки повинні бути пронумеровані арабськими цифрами в правому верхньому куті без крапки. Нумерація сторінок наскрізна від титульного аркуша до останнього аркуша тексту, враховуючи ілюстрації, таблиці, графіки. На титульному аркуші, завданні на курсовий проект нумерація сторінок не проставляється.

Викладені в тексті матеріали повинні наочно доповнювати й підтверджувати ілюстрації (схеми, рисунки, графіки, діаграми). Ілюстрації повинні відбивати тему курсового проекту. Студентові необхідно продумати, який матеріал проілюструвати. Це діаграми класів, схеми алгоритмів, схеми інформаційних зв'язків тощо.

Усі ілюстрації іменуються рисунками, їм привласнюється порядковий номер у межах номера розділу. Підпис ілюстрації складається із слова "Рисунок", номера ілюстрації та її назви (наприклад, "Рисунок 3.1 – Схема розміщення"). Рисунки потрібно виконувати на одній сторінці й розташовувати відразу після першого згадування в тексті, або на наступній сторінці. На всі ілюстрації мають бути посилання в тексті.

Посилання на ілюстрації роботи вказують порядковим номером ілюстрації, наприклад, "рис. 1.2".

У повторних посиланнях на ілюстрації треба вказувати скорочено слово "дивись", наприклад: "(див. рис. 1.2)".

У тому місті, де викладається тема, пов'язана з ілюстрацією, розміщують посилання у вигляді виразу в круглих дужках "(рис. 3.1)" або зворот типу: "... як показано на рис. 3.1".

Кожну формулу записують після першого згадування в тексті з нового рядка, симетрично до тексту. Між формулою і текстом пропускають один рядок. Формули нумеруються в межах розділу.

Пояснення до умовних літерних позначень в формулі наводять одразу під формулою. Для цього після формули ставлять кому і записують пояснення до кожного символу з нового рядка в тій послідовності, в якій вони наведені у формулі, розділяючи крапкою з комою. Перший рядок повинен починатися з абзацу із слова "де".

Наприклад,

$$I = \frac{U}{R}, \qquad (2.1)$$

де U – електрична напруга;

R – сила електричного струму.

Якщо формула займає декілька рядків, то вона може бути розірвана тільки на математичних знаках: додавання, віднімання, множення, ділення та інших, які повторюють на початку наступного рядка.

Таблицю необхідно розташовувати безпосередньо після тексту, в якому вона згадується вперше або на наступній сторінці. На всі таблиці повинні бути посилання. Таблиці послідовно нумеруються в межах розділу. Над лівим верхнім кутом таблиці міститься напис "Таблиця" із вказівкою її порядкового номера та назви (наприклад, "Таблиця 1.1 – Структурні елементи пояснювальної записки").

Переліки, за потреби, можуть бути наведені всередині пунктів або підпунктів. Перед переліком ставлять двокрапку.

Перед кожною позицією переліку слід ставити малу літеру української абетки з дужкою, або, не нумеруючи – дефіс (перший рівень деталізації).

Для подальшої деталізації переліку слід використовувати арабські цифри з дужкою (другий рівень деталізації).

Переліки першого рівня деталізації друкують малими літерами з абзацного відступу, другого рівня – відступом відносно місця розташування переліків першого рівня.

У тексті пояснювальної записки повинні бути посилання на літературу. Водночас наводиться її порядковий номер, записаний у квадратних дужках (наприклад, "...у роботах [1 – 7]").

Додатки потрібно оформляти як продовження пояснювальної записки на її наступних сторінках. Кожен додаток повинен починатися з нової сторінки. Додаток повинен мати заголовок, надрукований вгорі малими літерами з першої великої симетрично відносно тексту сторінки. Посередині рядка над заголовком малими літерами з першої великої повинно бути надруковано слово "Додаток" і велика літера, що позначає додаток (наприклад, "Додаток А").

Додатки позначаються послідовно великими літерами української абетки за винятком букв Ґ, Є, І, Ї, Й, О, Ч, Ь. Один додаток позначається як додаток А.

Рекомендована література

Основна

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання (ГОСТ 7.1-2003, IDT): ДСТУ ГОСТ 7.1:2006. – Київ : Держстандарт України, 2007. – 52 с.

2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. Максимчук, М. Энгл и др. ; пер. с англ. – Москва : ИД "Вильямс", 2008. – 720 с.

3. Документація. Звіти у сфері науки і техніки. Правила оформлення : ДСТУ 3008-95. – Київ : Держкомстат України, 1995. – 28 с.

4. 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, 2014. – 588 p.

Додаткова

5. Блинов И. Н. Java. Методы программирования : учеб.-метод. пособ. / И. Н. Блинов, В. С. Романчик. – Минск : Изд-во "Четыре четверти", 2013. – 768 с.

6. Леоненков А. Самоучитель UML 2 / А. Леоненков. – Санкт-Петербург : BHV, 2007. – 576 с.

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

Інформаційні ресурси

8. Уроки Java для начинающих [Электронный ресурс]. – Режим доступа : http://cybern.ru/category/java/begin-java.

9. Учебник по JavaFX 8 [Электронный ресурс]. – Режим доступа : http://code.makery.ch/library/javafx-8-tutorial/ru.

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

Додатки

Додаток А

Перелік рекомендованих тем курсових проектів

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 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 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 навчального закладу.

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

Зразок завдання на курсовий проект

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет економічної інформатики
Кафедра інформаційних систем

ЗАВДАННЯ
на комплексний курсовий проект: Програмування
студенту 2-го курсу групи 6.04.51.13.01
Іванову Івану Васильовичу

1. Тема проекту: "Розроблення програмної системи для автоматизації роботи з файловою базою даних <Назва предметної області>".
2. Термін здачі студентом закінченого проекту "__" _____ 201_ р.
3. Вхідні дані до проекту: літературні джерела, технічна документація щодо розроблення програм, ДСТУ з оформлення документації.
4. Зміст пояснювальної записки:
Вступ. Специфікація проекту. Програмна документація. Висновки. Додатки.
5. Перелік графічного матеріалу: UML-діаграма класів програмної системи.

Закінчення додатка Б

Таблиця Б.1

Календарний план
виконання курсового проекту

№ з/п	Назва етапу роботи	Строк виконання за планом	Відмітка про виконання
1.	З'ясування загальної постановки завдання. Розроблення чернетки першого розділу пояснювальної записки		
2.	Деталізація завдань курсового проектування. Уточнення архітектури програмної системи. Розроблення остаточного варіанту першого розділу пояснювальної записки		
3.	Програмна реалізація, налагодження та тестування застосунку		
4.	Остаточне налагодження та тестування застосунку. Розроблення чернетки другого розділу пояснювальної записки, висновків, списку використаних джерел, додатків, вступу, реферату)		
5.	Розроблення остаточного варіанта пояснювальної записки та підготовка електронної презентації		

Дата видачі завдання "__" _____ 201_ р.

Керівник: канд. техн. наук, доц. кафедри ІС_____ О. І. Петренко
(підпис)

Завдання прийняв до виконання _____ І. В. Іванов
(підпис)

Додаток В

Зразок оформлення титульного аркуша

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЕКТ: ПРОГРАМУВАННЯ

на тему: "Розроблення програмної системи для автоматизації роботи з файловою базою даних <Назва предметної області>"

Студента(ки) 2 курсу 6.04.51.16.01 групи
Спеціальності 122 "Комп'ютерні науки"
першого (бакалаврського) рівня
Іванова І. В.
Керівник: доцент кафедри ІС,
канд. техн. наук, доцент,
Петренко О. І.

Національна шкала _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Харків – 201_ рік

Додаток Г

Зразок оформлення реферату

РЕФЕРАТ

Пояснювальна записка до курсового проекту: 32 с., 20 рис., 3 табл., 6 джерел.

Об'єктом дослідження є методи моделювання інформаційних систем.

Метою роботи є дослідження способів представлення імітаційних моделей інформаційних систем.

Методами розробки обрано методи синтезу для поєднання переваг існуючих методів моделювання, метод аналізу для аналізу існуючих методів моделювання, табличний метод для відображення результатів оцінок ефективності розробленої моделі, метод імітаційного моделювання для розроблення моделі процесу обробки інформаційних повідомлень комп'ютером.

У результаті роботи було обґрунтовано вибір імітаційної моделі для представлення процесів функціонування інформаційних систем. Також було розроблено систему імітаційного моделювання, в якій реалізовано обраний алгоритм моделювання інформаційних систем.

Програмний продукт може бути використаний у науково-дослідницьких закладах під час побудови імітаційних моделей із використанням апарату Е-мереж.

ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЬ, Е-МЕРЕЖА, АДЕКВАТНІСТЬ МОДЕЛІ, СИСТЕМА МАСОВОГО ОБСЛУГОВУВАННЯ, ІМІТАЦІЙНА МОДЕЛЬ.

Додаток Д

Зразок оформлення змісту пояснювальної записки

ЗМІСТ

Додаток Ж

Методичні рекомендації щодо розроблення інтерфейсу користувача

Користувальницький інтерфейс є своєрідним комунікаційним каналом, за яким здійснюється взаємодія користувача й комп'ютера.

Щоб створити ефективний інтерфейс, що здійснював би роботу із програмою приємною, потрібно розуміти, які завдання будуть вирішувати користувачі за допомогою даної програми і які вимоги до інтерфейсу можуть виникнути в користувачів.

Загальні принципи проектування інтерфейсу користувача:

1. Програма повинна допомагати виконувати завдання.

Це означає, що інтерфейс повинен бути легким для освоєння й не створювати перед користувачем перешкоду, яку він повинен буде подолати, щоб розпочати роботу.

2. Під час роботи із програмою користувач не повинен відчувати дискомфорт.

Для реалізації цього принципу необхідно:

забезпечити перевірку результатів як можна більшого числа "некоректних" дій користувача, але не робити її повсюдно;

вказувати користувачеві, що саме йому робити, і виводити інформаційні повідомлення в ситуаціях, коли це дійсно необхідно;

надати досвідченим користувачам можливість відключення виведення інформаційних повідомлень;

добре продумувати зміст повідомлень, що виводяться користувачеві.

Широко відомі евристичні правила авторитетного американського фахівця в галузі проектування інтерфейсів Якоба Нільсена. Вони визначають мінімальні критерії, яким повинен відповідати інтерфейс будь-якої програми.

1. Наочність стану системи (правило зворотного зв'язку)

Система (у цьому випадку – комп'ютерна програма) повинна завжди інформувати користувача про стан своєї роботи за допомогою засобів зворотного зв'язку в прийнятний час.

Під час розгляду цього правила потрібно враховувати кілька аспектів:

користувач завжди повинен мати інформацію про поточний стан програми (наприклад, скільки часу пройшло від початку процесу копіювання файлів);

користувач обов'язково повинен бачити, до чого привела будь-яка його дія, наприклад, уведення даних, натискання кнопки;

вибір конкретного засобу зворотного зв'язку залежить від типу інформації, яку потрібно донести до користувача, а також типу дії, що викликає потребу у зворотному зв'язку.

Вважається, що якщо користувач виконав якусь дію й очікує результат його виконання, то цей результат (або повідомлення про помилку) повинен бути виведений в окремому діалоговому вікні. Якщо користувачеві необхідно надати поточну інформацію про процес, що не є прямим наслідком його дій, то можна обмежитися відображенням відповідного повідомлення в панелі стану.

Для організації зворотного зв'язку можуть бути використані й інші засоби. Найпопулярніші з них – звуки. Найчастіше звукове оповіщення допомагає тоді, коли поява на екрані діалогових вікон є небажаною, а повідомлення в панелі стану можуть бути не помічені користувачем.

Важливо пам'ятати, що звукове оповіщення не повинне бути основним засобом організації зворотного зв'язку. Звук повинен лише доповнювати текстові повідомлення.

Проміжок часу, протягом якого користувач одержує інформацію про реакцію на його дію або про подію, повинен бути мінімальним. Це особливо важливо, тому що наявність або відсутність у користувача інформації про поточний стан системи визначає його подальші дії.

2. Відповідність між системою й реальним світом

Система повинна взаємодіяти з користувачем на його мові. У цьому випадку мається на увазі використання понять, образів і цілих концепцій, які знайомі користувачеві з реальної предметної області. У жодному разі не можна використовувати спеціалізовані терміни, які придатні тільки для професійних довідників із програмування.

Найпоширеніший приклад реалізації цього принципу – побудова користувальницького інтерфейсу, що імітує об'єкти реального світу. Наприклад,

більшість із програм, що реалізують функції годинників, калькуляторів, програвачів компакт-дисків, записних книжок виглядають майже точно так, як їхні матеріальні аналоги. Знаменитий "кошик сміття" на робочому столі операційної системи Windows, у який можна "кинути" непотрібний файл або папку, – класичний приклад побудови інтерфейсу на основі об'єктів реального світу.

3. Управління користувачами та свобода їхніх дій

Користувачі повинні мати можливість управління системою й зміни її поточного стану. Однак, вони часто дають різні команди помилково (наприклад, випадково натиснувши кнопку або обравши не той пункт меню). Отже, у користувача повинен бути "аварійний вихід" із цієї ситуації, чітко позначений у програмі. Найчастіше такий "вихід" реалізується у вигляді кнопки "Відміна", розташованої в діалоговому вікні, що дозволяє припинити виконання поточної операції або закрити це діалогове вікно. Крім цього, традиційним і тому звичним для більшості користувачів засобом "аварійного виходу" є натискання на клавіатурі клавіші <Escape>.

Хорошим тоном вважається сполучення цих способів "аварійного виходу".

Ще один засіб виходу з помилкової ситуації – команди "Undo" ("Відмінити") і "Redo" ("Повторити"). Вони є настільки зручними й підтримуються такою великою кількістю програм, що користувачі підсвідомо очікують, що будь-яку їхню дію можливо відмінити, повернувшись до попереднього стану.

Усе це зобов'язує розроблювача дружнього користувальницького інтерфейсу комп'ютерної програми підтримувати дані команди. Якщо через якісь причини дію, на виконання якої дав команду користувач, не можна відмінити, то на екрані повинно з'явитися відповідне попередження, а також прохання підтвердити виконання команди.

4. Несуперечність і стандарти

Цей принцип означає використання тих самих засобів для вираження схожих образів і виконання дій, що мають однакову природу.

По-перше, це означає несуперечність під час вибору засобів оповіщення про події та дії. Наприклад, інформація про поточний стан програми, звичайно виводиться в панелі стану, а повідомлення з результатами запитів користувача – в окремих діалогових вікнах.

Повідомлення про критичні помилки водночас повинні сильно відрізнятися від звичайних інформаційних повідомлень: наприклад, вони можуть супроводжуватися різким звуком.

По-друге, це означає несуперечність під час оформлення елементів користувальницького інтерфейсу. Наприклад, якщо він ґрунтується на класичному інтерфейсі Windows-застосунків, що характеризується певною колірною гамою, прямими лініями й кутами, то дуже дивним виглядало б рішення додати одному з вікон програми овальну форму й розфарбувати його яскравими кольорами.

По-третє, це означає несуперечність під час вибору термінів. Користувачів не повинне спантеличувати те, що кілька різних понять, які використовуються в програмі, насправді означають те саме.

Головне – вирішити, що для позначення якоїсь конкретної дії або події буде застосовуватися один конкретний термін, що буде використовуватися певним способом (наприклад, слово "Інтернет" буде починатися із великої букви й не відмінюватися).

Принцип несуперечності – одне з найважливіших правил проектування користувальницьких інтерфейсів. Несуперечливий інтерфейс інтуїтивно зрозумілий і дуже легкий для освоєння, тому що під час його вивчення користувач не зіштовхується із сюрпризами, і навіть ті частини інтерфейсу, які він бачить уперше, здаються йому давно знайомими.

5. Запобігання помилок

Стосовно теми проектування користувальницького інтерфейсу комп'ютерних програм, цей принцип означає таке: "Дизайн, що запобігає виникнення помилок, краще, ніж найкраще повідомлення про помилку".

6. Розуміння краще, ніж запам'ятовування

Під час розроблення інтерфейсу потрібно робити всі об'єкти, функції, дії легкодоступними користувачеві. Користувач не повинен постійно намагатися утримати в пам'яті інформацію з однієї частини програми, щоб застосувати її в іншій. У будь-який момент часу користувачеві повинно бути зрозуміло, що йому зараз потрібно робити. У хорошому інтерфейсі інструкції з використання системи доступні для виклику в будь-який час, коли це потрібно. Це може бути реалізоване як у вигляді продуманої організації елементів інтерфейсу, так і у вигляді підказок користувачеві.

Це правило відбиває принцип "прозорого" інтерфейсу – інтерфейсу, що зрозумілий і не змушує користувача згадувати, яку кнопку потрібно натиснути або який пункт меню вибрати в даний момент.

7. Гнучкість і ефективність використання

Під час проектування інтерфейсу користувача перед розроблювачем часто постає така проблема: потрібно, щоб інтерфейс був однаково зручний і для новачків, і для досвідчених користувачів.

Для вирішення цієї проблеми використовують простий прийом: функції, які прискорюють роботу, оформлюють так, щоб їх не бачив початківець, але щоб вони були доступні досвідченим користувачам. Найпростіший приклад – це "гарячі клавіші", за допомогою яких можна швидко викликати функції програми, що часто виконуються. Позначення "гарячих клавіш" пишуться поруч із відповідними пунктами меню, тому вони, з одного боку, не заважають новачкам, а, з іншого – доступні досвідченим користувачам.

Інший приклад реалізації універсального користувальницького інтерфейсу – можливість виконати складні функції програми як за допомогою "майстра", що, немов за руку, "проведе" починаючого користувача по всіх етапах процесу, так і вручну, за допомогою настроювання опцій у відповідному діалоговому вікні.

8. Естетичний і мінімалістський дизайн

Це правило означає: "Нічого зайвого". Не потрібно захаращувати користувальницький інтерфейс програми елементами, які є недоречними й малокорисними. Справа в тому, що кожний елемент, наприклад, кнопка або текстовий підпис, обов'язково відволікає частину уваги користувача. Це може призвести до того, що видимість і, відповідно, легкість сприйняття користувачем дійсно потрібних і корисних частин інтерфейсу буде набагато менше за рахунок елементів, без яких цілком можна було б обійтися.

9. Розпізнавання й виправлення помилок

"Допомагайте користувачеві розпізнавати й виправляти помилки" – стверджує це правило.

Воно визначає проектування повідомлень про помилки. "Гарні" повідомлення про помилки – це повідомлення, які пояснюють, у чому полягає проблема й, найголовніше, як її виправити. Таким чином, "гарне" повідомлення

про помилку повинне складатися із двох частин: опису помилки й опису вирішення проблеми.

Опис помилки повинен бути чітким, ясним і зрозумілим, давати користувачеві всю необхідну інформацію про причини й місце виникнення помилки. Найпростіше рішення – створити в довідковій системі програми відповідний розділ, що роз'ясняє зміст проблеми й причини її виникнення. У самому ж діалоговому вікні з повідомленням про помилку може бути присутнім кнопка "Довідка" для виклику цього розділу.

Ще одним прикладом рішення даної проблеми є кнопка "Докладніше", під час натискання на яку діалогове вікно з повідомленням про помилку розгортається, відображаючи більш докладну інформацію про причину виникнення збою.

Дуже важливо пам'ятати те, що повідомлення про помилку повинне містити її опис, а не числовий код помилки.

Існує багато програм, у яких повідомлення про помилки містять недостовірну інформацію, повідомляючи користувача зовсім не про ті проблеми, які виникли насправді. Тому під час складання описів помилок потрібно не забувати перевіряти коректність повідомлень, що генеруються програмою.

Відомо, що інформація про те, як виправити помилку або вирішити проблему має навіть більше значення, ніж опис помилки або проблеми. У ході опису шляху вирішення проблеми потрібно уникати складання занадто об'ємних текстів. В іншому разі користувачі будуть просто пробігати їх очами, не доходячи до змісту написаного. Найкраще скласти покрокову інструкцію, кожний крок якої складається з 1 – 2 речень.

10. Довідка й документація

Це правило полягає в необхідності надання користувачеві довідкової системи й документації до програми.

Проектування форм

Форми – це "будівельні блоки" інтерфейсу користувача.

Щоб створити добре спроектовану форму, необхідно усвідомити її призначення, спосіб і час використання, а також її зв'язки з іншими елементами програми.

Особливий вид форм – форми, призначені для введення даних.

Під час їхньої розробки основну увагу варто приділити швидкості їх використання. Основне правило – якщо користувач збирається ввести в базу даних велику кількість записів, то він не повинен підтверджувати введення кожної з них.

Щоб прискорити процес уведення даних, необхідно:

1. По можливості використовувати для додавання й редагування даних одну й ту саму форму.

2. Призначати клавіатурні сполучення для команд.

3. Не змушувати користувача "перестрибувати" з однієї частини форми в іншу.

4. Не ставити процес уведення даних у залежність від вмісту окремих елементів управління форми.

5. Використовувати засоби зворотного зв'язку з користувачем.

Ефективні меню

Ще одна важлива частина розроблення форм – створення змістовних і ефективних меню. От деякі важливі рекомендації:

1. Додержуйтеся стандартних угод про розташування пунктів меню, прийнятим в операційній системі.

2. Групуйте пункти меню в логічному порядку й за змістом.

3. Для угруповання пунктів у меню, що розкриваються, використовуйте розділові лінії.

4. Уникайте надлишкових меню.

5. Уникайте пунктів меню верхнього рівня, які не мають меню, що розкриваються.

6. Не забувайте використовувати символ <…> для позначення пунктів меню, що активізують діалогові вікна.

7. Обов'язково використовуйте клавіатурні еквіваленти команд і "гарячі" клавіші.

8. Поміщайте на панель інструментів часто використовувані команди меню.

Приклад опису UML-діаграми класів

Діаграму класів програми наведено на рис. 2.1.

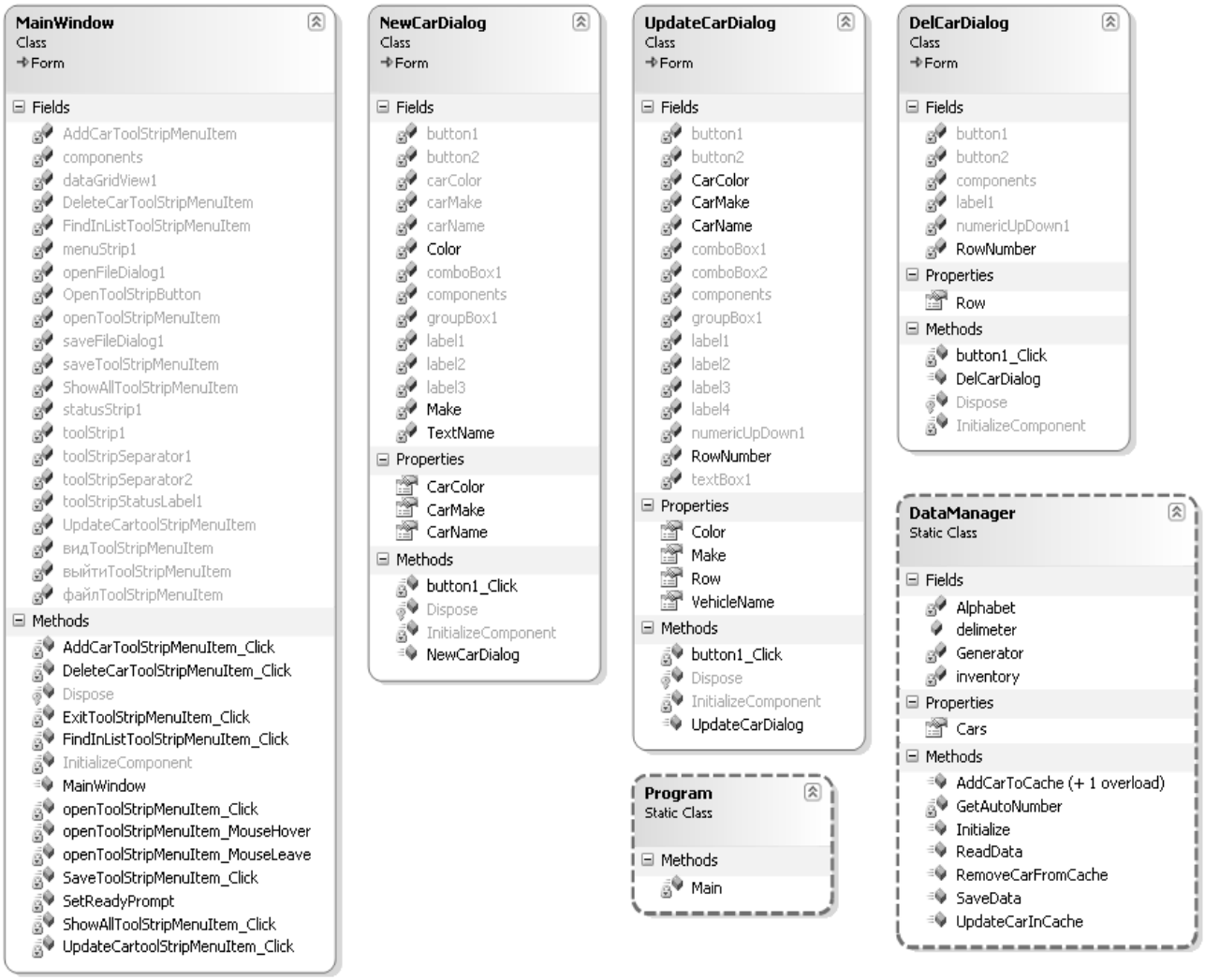

Рис. 2.1 – UML-діаграма класів

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 про новий 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 інтерфейсу програми.

Додаток И

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 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 відкриття файлу	Діалогове вікно відкриття файлу з'являється	Так
2.	У діалоговому вікні відкриття файлу вибрати ім'я файлу з 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, коректна 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 форматі та натиснути кнопку "ОК"	Повинне 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): 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 Java SE

Вимоги до апаратних засобів:

1. Процесор – не нижче Pentium 2 266 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
• Mac OS X 10.7.3 (Lion) і старше
• Solaris 10 і старше
2. Java Runtime Environment 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 Java Runtime Environment 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 у start.bat наступні команди (операційна система Windows):

```
@echo OFF
set PATH =.\appjre\bin
java -jar dist\install.jar
pause> NUL
```

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

Додаток Л

Приклад керівництва користувача (у скороченні)

2.4 Керівництво користувача

2.4.1 Призначення програмного продукту

Програмний продукт призначено для проведення тестування у закладах вищої освіти. Універсальність програми дозволяє працювати з нею як викладачам, так і студентам. Викладач може працювати із такими інструментами: створення, видалення, редагування тестів. Студент може проходити тестування за дисципліною з автоматичним виставлянням оцінки або складанням висновків, якщо тест демонстраційний.

2.4.2 Використання програмного продукту

Запуск програми

Запуск програми в операційній системі сімейства Windows здійснюється одним з стандартних способів:

а) подвійним клацанням лівою кнопкою миші на ярлику програми;

б) викликом контекстного меню з вибором його пункту "Відкрити";

в) натисканням кнопки "Пуск" панелі завдань із подальшим вибором пункту "Усі програми" та подвійним клацанням лівою кнопкою миші на ярлику програми.

Вхід користувача в систему

Після запуску програми на екрані монітора з'являється вікно "Вхід" для введення імені й пароля користувача (рис. 2.1).

Для входу в програму в цьому вікні необхідно вибрати ім'я користувача з відповідного списку та ввести пароль користувача в поле "Пароль", після чого натиснути кнопку "Так". Під час натискання кнопки "Вихід" робота програми завершується.

Основні елементи графічного інтерфейсу програми

Графічний інтерфейс програми складається з головного вікна (рис. 2.2) та додаткових діалогових вікон.

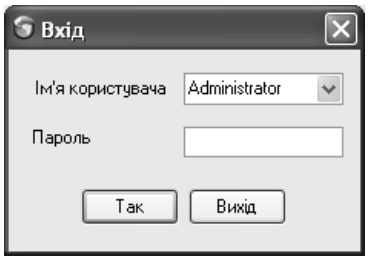

Рис. 2.1 – Вікно входу в програму

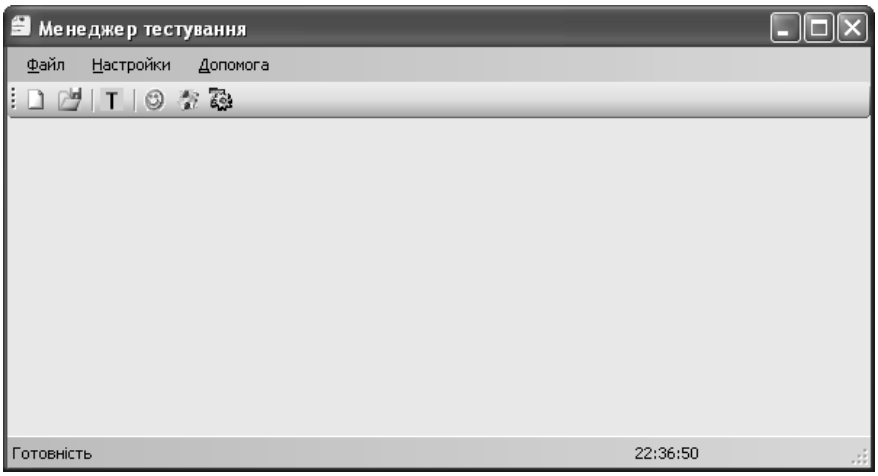

Рис. 2.2 – Головне вікно програми

Головне вікно програми має панель меню, панель інструментів та панель стану.

Меню програми містить усі команди для керування її виконанням. Воно має наступну структуру:

а) "Файл"

1. "Створити тест";
2. "Змінити тест";
3. "Почати тест";
4. "Вихід".

б) "Настройки"

1. "Панель інструментів";
2. "Панель стану";
3. "Годинник";

в) "Менеджер облікових записів";

г) "Допомога";

д) "Про програму".

На панелі інструментів знаходяться кнопки для виклику команд управління виконанням програми, які використовуються найбільш часто. Результат натискання будь-якої кнопки панелі інструментів є аналогічним вибору відповідного пункту меню програми.

На панель стану виводиться додаткова інформація щодо функціонування програми, зокрема назва поточного режиму її роботи.

Робота з програмою

Створення нового тесту

Для початку роботи необхідно вибрати пункт меню **"**Створити тест". Після цього на екран виводиться вікно "Параметри тесту", представлене на рис. 2.3, де користувачеві необхідно вибрати тип тесту, ввести його назву, вибрати шкалу оцінювання та ввести назву дисципліни, до якої належить тест.

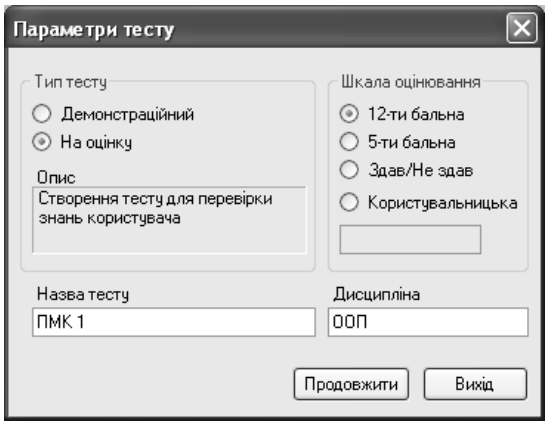

Рис. 2.3 – Вікно параметрів нового тесту

Демонстраційний тип тесту призначений для "психологічного" тестування, коли у вікні результатів тестування можна одержати інформацію, щодо загального рівня знань студента з дисципліни та рекомендації щодо його удосконалення. У даному випадку група елементів управління "Шкала оцінювання" буде недоступною. Для типу тесту "На оцінку" використовується одна з трьох стандартних шкал оцінювання, або користувальницька шкала.

Якщо вибрана користувальницька шкала, то необхідно ввести максимальну оцінку за цією шкалою.

Для виходу з режиму створення тесту необхідно натиснути кнопку "Вихід".

Для продовження роботи зі створення тесту необхідно натиснути кнопку "Продовжити". Після цього керування передається вікну редагування нового тесту (рис. 2.4).

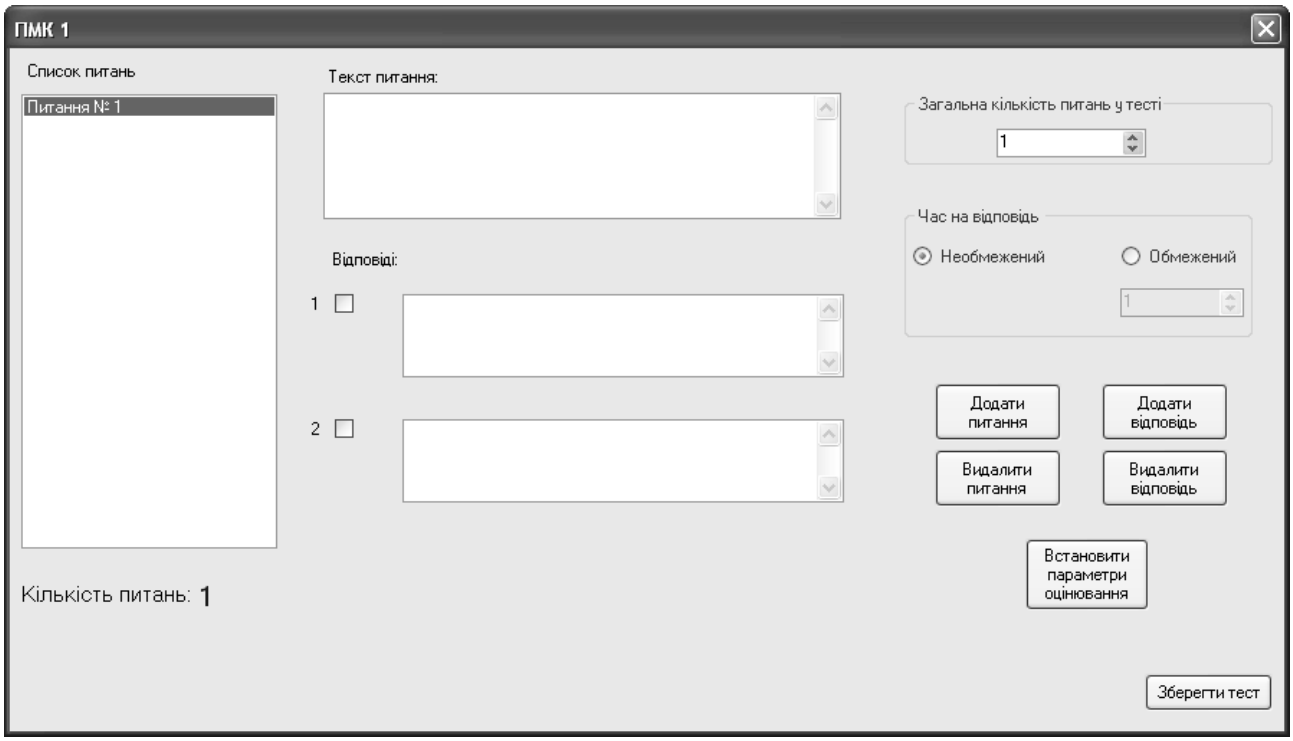

Рис. 2.4 – Вікно редагування нового тесту

У цьому вікні містяться елементи управління для редагування тексту питань та відповідей, вибору правильних відповідей, відображення списку питань та їх кількості, визначення загальних параметрів тесту та необхідні командні кнопки.

Якщо час на відповідь обмежується, то необхідно задати його значення за допомогою списку в групі елементів управління "Час на відповідь".

Кнопка "Додати відповідь" призначена для додавання до даного вікна поля для введення нового варіанта відповіді на поточне питання.

Кнопка "Видалити відповідь" використовується для виконання протилежної операції.

Кнопка "Додати питання" призначена для додавання до тесту нового питання, а кнопка "Видалити відповідь" – для видалення питання з тесту.

Кнопка "Встановити параметри оцінювання" призначена для виклику діалогового вікна, у якому необхідно вибрати параметри шкали оцінювання для тесту.

Під час натискання кнопки "Зберегти тест" робота з даним тестом припиняється, він зберігається на жорсткому диску комп'ютера та керування передається головному вікну програми.

Увага! Далі необхідно навести інформацію про порядок використання інших можливостей програмного продукту, що доступні користувачу, та відповідні екранні форми.

2.4.3 Повідомлення користувачеві

Під час входу в програму виводиться інформація, що повідомляє користувача про його роль під час роботи в програмі (рис. 2.5).

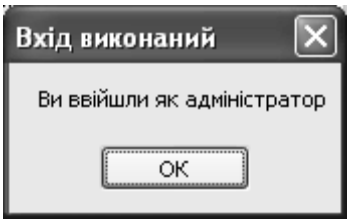

Рис. 2.5 – Повідомлення при вході в програму

Під час створення нового тесту необхідно вказувати всі його атрибути у відповідних полях уведення, інакше подальша робота з тестом буде неможлива (рис. 2.6).

Під час створення тесту, якщо не обрано жодної правильної відповіді, виводиться відповідне діалогове вікно (рис. 2.7).

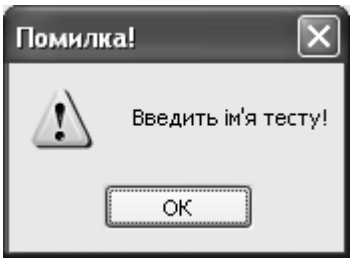

Рис. 2.6 – Повідомлення про неповну вказівку атрибутів нового тесту

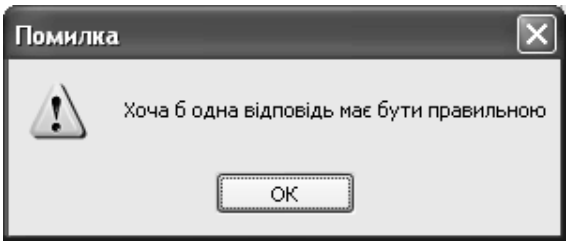

Рис. 2.7 – Повідомлення про відсутність правильної відповіді

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ПРОГРАМУВАННЯ

Методичні рекомендації
до виконання комплексного курсового проекту
для студентів спеціальності
122 "Комп'ютерні науки"
першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладачі: **Парфьонов** Юрій Едуардович
Федорченко Володимир Миколайович
Щербаков Олександр Всеволодович

Відповідальний за видання *О. Г. Руденко*

Редактор *О. В. Анацька*

Коректор *О. В. Анацька*

План 2018 р. Поз. № 139 ЕВ. Обсяг 48 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 20.02.2015 р.