МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Методичні рекомендації до виконання курсового проекту з навчальної дисципліни "ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

для студентів напряму підготовки 6.050101 "Комп'ютерні науки" всіх форм навчання

Харків. Вид. ХНЕУ ім. С. Кузнеця, 2015

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 2 від 03.10.2014 р.

Укладач Парфьонов Ю. Е.

М54 Методичні рекомендації до виконання курсового проекту з навчальної дисципліни "Об'єктно-орієнтоване програмування" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" всіх форм навчання / укл. Ю. Е. Парфьонов. – Х. : Вид. ХНЕУ ім. С. Кузнеця, 2015. – 44 с. (Укр. мов.)

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

Рекомендовано для студентів напряму підготовки 6.050101 "Комп'ютерні науки" всіх форм навчання.

Вступ

Сьогоднішні умови господарювання вимагають від фахівців з економічного управління всебічного використання новітніх інформаційних технологій. Широкі можливості комп'ютеризованих засобів у питаннях 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 розробки програмного забезпечення збільшується, оскільки тенденції розвитку комп'ютерної техніки свідчать про те, що з одного боку складність та функціональні можливості комп'ютерної техніки постійно і швидко зростають, а з другого боку, це потребує більш досконалих програмних засобів для задоволення потреб користувачів.

Істотною рисою таких програмних систем є рівень складності: для одного розробника практично неможливо охопити усі її аспекти. Причому ця складність є неминучою: з нею можливо справитися, але позбавитися від неї неможливо.

У теперішній час найбільш розповсюдженим методом боротьби зі складністю є об'єктно-орієнтований підхід до розробки програмного забезпечення. З використанням цього підходу розробляється більша частина програм у всьому світі. Це потрібує від відповідних фахівців чіткого уявлення концепцій об'єктно-орієнтованого програмування, що дає можливість їх практичного використання при розробці додатків на будь-якій мові програмування.

Метою навчальної дисципліни "Об'єктно-орієнтоване програмування" є засвоєння необхідних знань з основ об'єктно-орієнтованого програмування, а також формування твердих практичних навичок щодо розробки додатків із використанням об'єктно-орієнтованого підходу.

Завершальним етапом вивчення даної дисципліни є курсове проектування.

1. Мета й завдання курсового проектування

Метою курсового проектування є закріплення та поглиблення знань з дисципліни "Об'єктно-орієнтоване програмування".

Робота над курсовим проектом сприяє систематизації, поглибленню й закріпленню знань, отриманих студентами при вивченні даної дисципліни. У процесі курсового проектування студенти розвивають навички практичного застосування отриманих знань при створенні комплексного додатку з використанням сучасних інструментальних засобів розробки. При цьому студент повинен показати вміння користуватися спеціальною літературою, державними стандартами, довідниками та іншими матеріалами з інформаційних технологій.

У розробленому курсовому проекті студент повинен показати **знання:**

предметної області відповідно до постановки завдання;

основних концепцій об'єктно-орієнтованого програмування;

сучасного стану та перспектив розвитку технологій програмування;

сучасних інструментальних засобів, призначених для розробки об'єктно-орієнтованих додатків.

Студент повинен показати **вміння** з:

аналізу постановки завдання;

декомпозиції програмної системи на підсистеми;

розробки загальної архітектури програмної системи;

деталізації загальної архітектури програмної системи у термінах об'єктно-орієнтованого програмування;

програмної реалізації системи, що розробляється, на мові програмування;

застосування стандартних бібліотек мови програмування;

документування вихідного коду програми;

використання засобів розробки програм та отримання довідкової інформації;

розробки та оформлення програмної документації.

Робота над 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 повинен розглядати роботу над курсовим проектом як своєрідну "репетицію" дипломного проектування.

2. Організація курсового проектування

Курсовий проект студент виконує самостійно. 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, які приймають участь у викладанні цієї дисципліни. Керівник курсового проекту рекомендує студенту основну літературу, орієнтує його на розробку необхідних проектних рішень.

Студенту може бути призначена тема курсового проекту з переліку рекомендованих тем. Також йому надається право самостійного вибору теми проекту. Якщо тема пропонується студентом, то вона повинна 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 у додатку А.

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 застосування, 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 повинна супроводжуватися демонстрацією електронної презентації.

3. Структура та обсяг курсового проекту

Курсовий проект складається з пояснювальної записки та інших матеріалів, зокрема програмного застосування, 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 структура пояснювальної записки курсового проекту.

Таблиця 1

Структура пояснювальної записки курсового проекту

Структурні елементи пояснювальної записки	Приблизна 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	

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 аркушу наведений у додатку Б.

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.

Реферат повинен містити:

текст реферату;

перелік ключових слів.

Текст реферату повинен відбивати подану в пояснювальній записці інформацію у 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'єкту дослідження або розроблення.

Реферат належить виконувати обсягом не більш, як 500 слів, і, бажано, щоб він уміщувався на одній сторінці формату А4.

Ключові слова призначені для розкриття суті 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.

Приклад реферату наведений у додатку В.

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.

Приклад змісту пояснювальної записки наведений у додатку Д.

У вступі слід зазначити важливість використання інформаційних систем у сучасних умовах, роль інформаційних систем у предметної галузі, що розглядається в курсовому проекті, роль засобів збереження даних у інформаційних системах, мету та задачі курсового проекту, відомості щодо розробленої програми (призначення, що вона дозволяє автоматизувати, технології та мова програмування, що були використані, при розробленні 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 варіанту, 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 та 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 й таке інше.

Функціональні та нефункціональні вимоги до програмної системи, що розробляється відповідно до будь-якої теми курсового проекту з переліку рекомендованих тем, наведено нижче. В обґрунтованих випадках деякі з цих вимог можуть бути змінені за узгодженням керівника курсового проекту.

Функціональні вимоги

1. Створення файлу бази даних та запис до нього даних у певному форматі.

2. Читання всіх даних з файлу та їх відображення.

3. Додавання нового елементу даних до файлу бази.

4. Оновлення будь-якого елементу даних у файлі бази.

5. Видалення будь-якого елементу даних з 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 у 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 інформацію про розроблювача програми, зокрема його (її) фотографію.

8. Усі діалогові вікна повинні бути модальними та мати фіксований розмір.

При розробленні 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 виключень 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 тегів JavaDoc.

4. Виконання угод щодо запису тексту програм мовою програмування Java (додаток З).

Якщо студент вибрав тему курсового проекту самостійно, то вимоги до програми обговорюються з керівником та можуть відрізнятися.

У підрозділі 1.3 "Математичний опис задачі" має бути наведена математична формалізація задачі, тобто існуючі співвідношення між величинами. При цьому 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 та 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 реалізованим є можливість практичного виконання необхідних обчислень за відведений час при доступних витратах необхідних 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 з 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 розроблюється. Це може бути крупна колекція з безлічі компонентів програмного забезпечення, один додаток або частина додатку.

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 наведений у 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 приводить до несподіваного поводження системи (тобто поводження, що не відповідає вимогам).

Завдання тестування – визначення умов, при 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

По завершенню інтеграційного тестування всі модулі системи є узгодженими за інтерфейсами і функціональностями. 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 не функціональні.

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

Важливим видом системного тестування є функціональне тестування. Даний 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 при тестуванні, – 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 знайомства з її програмним кодом.

Структура тест-вимог повинна додержуватися структури функціональних вимог на систему. 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 деяку 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-вимогам.

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 наведений у додатку К.

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

Для опису процедури тестування повинні бути складені такі 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 додатку 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 в даному пункті пояснювальної записки.

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.

Підрозділ 2.4 "Керівництво користувача" містить призначення програми, інструкцію щодо її використання за призначенням, опис повідомлень користувачу, що можуть з'явитися у 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 наведений у 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 у 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 у [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 при роботі над курсовим проектом має його оформлення, до якого пред'являються певні вимоги.

При оформленні 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 документу повинен бути виконаний з 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 при поданні 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 документу, можна виправляти підчищенням або зафарбовуванням білою фарбою і нанесенням на тому ж місці виправленого тексту. Допускається наявність не більше двох виправлень на одній сторінці.

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

Кожній 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 включає номер розділу й порядковий номер підрозділу, які розділяються крапкою, наприклад: 1.1, 1.2, 1.3. Номер пункту включає 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, включаючи ілюстрації, таблиці, графіки. На титульному аркуші, завданні на курсовий проект нумерація сторінок не проставляється.

Викладені у 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 зі 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 у 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 у 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 у формулі наводять зразу ж 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 математичні знаки: додавання, віднімання, множення, ділення та інших, які повторюють на початку наступного рядка.

Таблицю необхідно розташовувати безпосередньо після тексту, у 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. При цьому 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 угорі 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 Г, 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. – К. : Держстандарт України, 2007. – 52 с.

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

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

4. Шилдт Г. Java. Руководство для начинающих / Г. Шилдт ; пер. с англ. – М. : ООО "И.Д. Вильямс", 2012 – 624 с.

5. Шилдт Г. Swing: руководство для начинающих / Г. Шилдт ; пер. с англ. – М. : ООО "И.Д. Вильямс", 2007 – 704 с.

Додаткова

6. Сьерра К. Изучаем Java / К. Сьерра, Б. Бейтс ; пер. с англ. – М. : ЭКСМО, 2012. – 718 с.

Ресурси мережі Інтернет

7. Веб-сайт проекту Eclipse. – Режим доступу : www.eclipse.org.

8. Електронний посібник "The Java Tutorials" [Електронний ресурс]. – Режим доступу : http://download.oracle.com/javase/tutorial/.

9. Онлайн-документація щодо бібліотеки класів Java SE 8 [Електронний ресурс]. – Режим доступу : http://download.oracle.com/javase/8/docs/api/index.html.

10. Программирование на Java для детей, родителей, дедушек и бабушек [Електронний ресурс]. – Режим доступу : http://myflex.org/books/java4kids/JavaKid8x11_ru.pdf.

11. Java for Beginners [Електронний ресурс]. – Режим доступу : http://www.homeandlearn.co.uk/java/java.html.

12. Java Swing и другая Джава [Електронний ресурс]. – Режим доступу : http://javaswing.wordpress.com/.

13. Java Swing [Електронний ресурс]. – Режим доступу : http://www.zentut.com/java-swing/.

14. Java Tutorial [Електронний ресурс]. – Режим доступу : http://www.tutorialspoint.com/java/index.htm.

Додатки

Додаток А

Зразок завдання на курсовий проект

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет економічної інформатики
Кафедра інформаційних систем
Дисципліна: "Об'єктно-орієнтоване програмування"

ЗАВДАННЯ
на курсовий проект
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.

Календарний план
виконання курсового проекту

№ п/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 додатку		
4.	Остаточне налагодження та тестування додатку. Розроблення чернетки другого розділу пояснювальної записки, висновків, списку використаних джерел, додатків, вступу, реферату)		
5.	Розроблення остаточного варіанту пояснювальної записки та підготовка електронної презентації		

Дата видачі завдання "__" _____ 201_ р.

Керівник к.т.н., доц. кафедри ІС_____ О. І. Петренко
(підпис)

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

Додаток Б

Зразок оформлення титульного аркуша

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

КУРСОВИЙ ПРОЕКТ

з навчальної дисципліни "Об'єктно-орієнтоване програмування"
на тему: "Розроблення програмної системи для автоматизації роботи з файловою базою даних <Назва предметної області>"

Студента(ки) 2 курсу 6.04.51.13.01 групи
напряму підготовки 6.050101
Іванова І. В.
Керівник: доцент кафедри ІС, к.т.н., 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 В

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 подання 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 розробки 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 при побудові імітаційних моделей з 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 значить, що інтерфейс повинен бути легким для освоєння й не створювати перед користувачем перешкоду, яку він повинен буде подолати, щоб приступити до роботи.

2. При роботі 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.

При розгляді 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 привело будь-яку його дію, 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 в зворотному зв'язку.

Уважається, 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 ж 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. Найбільш часто 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 з 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 "сміттєвий кошик" на "робочому столі" операційної системи Windows, у яку 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 вибравши 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.

Все це зобов'язує розроблювача дружнього користувальницького інтерфейсу комп'ютерної програми підтримувати дані команди. Якщо ж за якимись причинами дію, на виконання якої дав команду користувач, не можна відмінити, то на екран повинне буде виведене 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 при виборі 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 при цьому повинні сильно відрізнятися від звичайних інформаційних повідомлень: наприклад, вони можуть супроводжуватися різким звуком.

По-друге, це означає несуперечність при оформленні елементів користувальницького інтерфейсу. Наприклад, якщо він ґрунтується на класичному інтерфейсі Windows-додатків, що характеризується строгою 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 при виборі 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 при його вивченні 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

При розробленні 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

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

Для рішення 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 боку, 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, будь то кнопка або текстовий підпис, обов'язково відволікає частину уваги користувача. Це може привести до того, що видимість і, відповідно, легкість сприйняття користувачем дійсно потрібних і корисних частин інтерфейсу буде сильно зменшена за рахунок елементів, без яких цілком можна було б обійтися.

9. Розпізнавання й виправлення помилок

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

Воно визначає проектування повідомлень про помилки. "Гарні" повідомлення про помилки – це повідомлення, які пояснюють, у чому заключається 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", при натисканні на яку діалогове вікно з повідомленням про помилку "розорюється", 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 при складанні 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. При описі 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.

При їхній розробці основну увагу варто приділити швидкості їхнього 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 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 розробки форм – створення змістових 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.

Додаток 3

Угоди щодо запису тексту програм мовою програмування Java

1. Функціональність імен:

result
firstNumber
secondNumber

2. Перед та після знаків операцій (+ – * …) робиться пробіл:

result = firstNumber + secondNumber;

3. Після ключових слів робиться пробіл:

if (…)

4. Імена класів, інтерфейсів починаються з великої літери:

MyClass

5. Імена полів, методів, посилань на об'єкти починаються з малої літери:

int firstNumber, secondNumber;

int getResult();

6. Імена пакетів записуються малими літерами:

package **my.model**

7. Великі літери в іменах констант, між словами – знаки підкреслення:

final int STRING_LENGTH = 100;

Додаток И

Приклад опису UML-діаграми класів

Діаграма класів програми наведено на рис. 2.1.

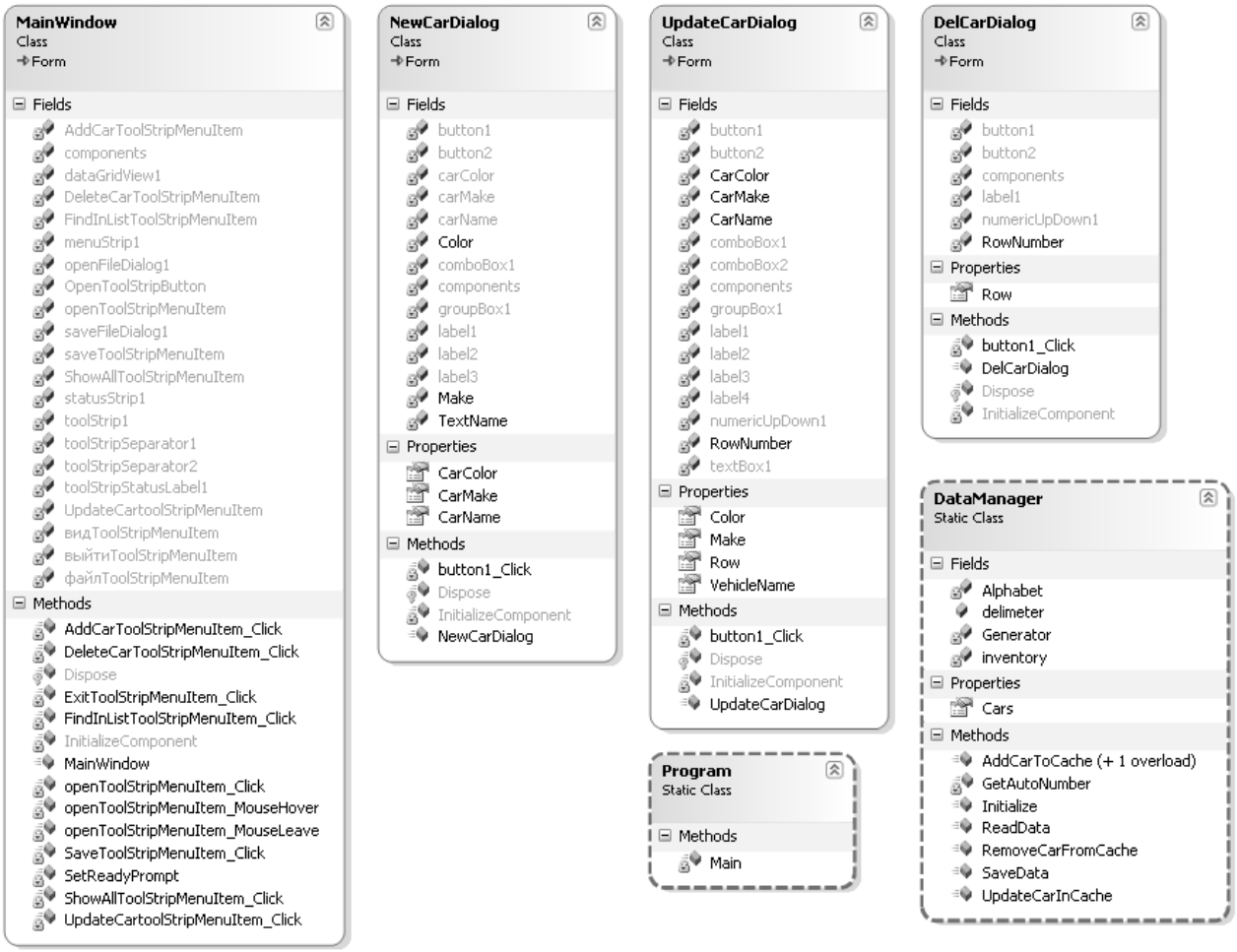

Рис. 2.1 **UML-діаграма класів**

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" при 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 К

Приклад тест-плану, сполученого зі звітом про проведення тестування

Тестовий приклад: № 1

Призначення: перевірка того, що програмна система дозволяє виконувати читання даних з файлу та коректна відображати їх у графічному інтерфейсі користувача.

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

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

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

№ п/п	Крок сценарію	Очікуваний результат	Отриманий результат	Відмітка про проходження кроку сценарію (Так/Ні)
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 коректно відобразитися дані, що були завантажені з файлу	Дані, що були завантажені з файлу, коректно відображуються у головному вікні програми	Так
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 (пройдений/не пройдений): пройдений

Тестовий приклад: № 2

……

Тестовий приклад: № 3

……

Тестових прикладів виконано: 3

Тестових прикладів пройдено: 1

Додаток Л

Приклад керівництва користувача (у скороченні)

2.4. Керівництво користувача

2.4.1. Призначення програмного продукту

Програмний продукт призначено для проведення тестування у вищих навчальних закладах. 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 по заданій дисципліні 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 з 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).

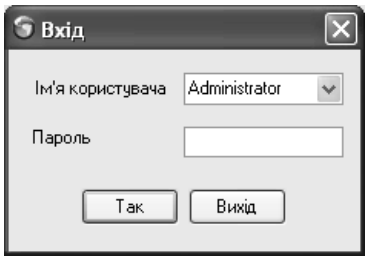

Рис. 2.1 **Вікно входу в програму**

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". При натисканні кнопки "Вихід" робота програми завершується.

Основні елементи графічного інтерфейсу програми

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

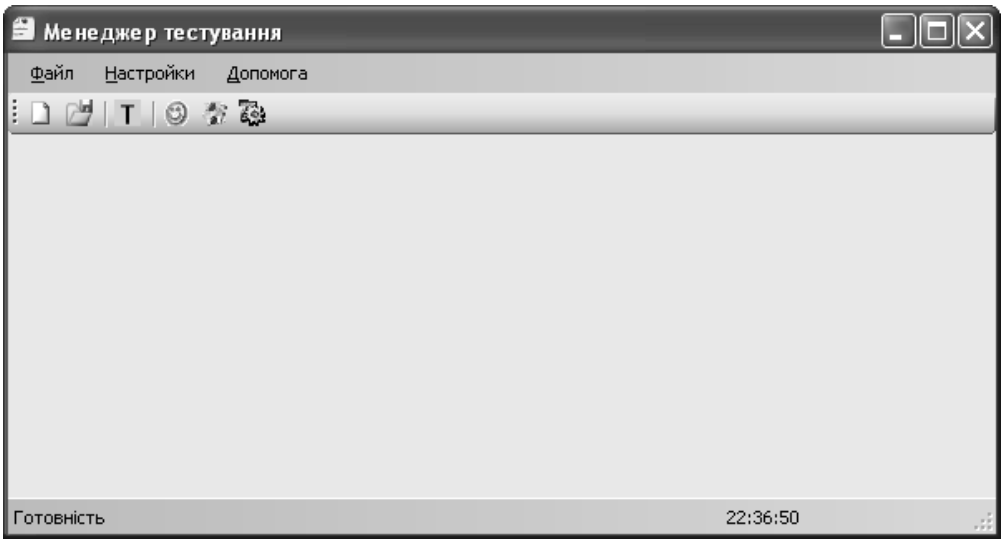

Рис. 2.2 **Головне вікно програми**

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

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

а) "Файл":

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

б) "Настройки":

1. "Панель інструментів".
2. "Панель стану".
3. "Годинник".
4. "Менеджер облікових записів".

в) "Допомога".

1. "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 відноситься тест.

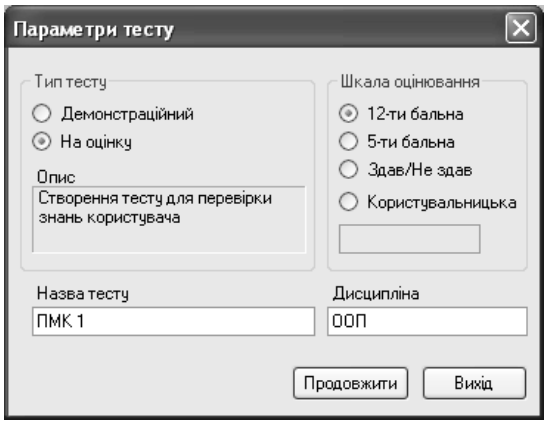

Рис. 2.3 **Вікно параметрів нового тесту**

Демонстраційний тип тесту призначений для "психологічного" тестування, коли в 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 по створенню тесту необхідно натиснути кнопку "Продовжити". Після цього керування передається вікну редагування нового тесту (рис. 2.4).

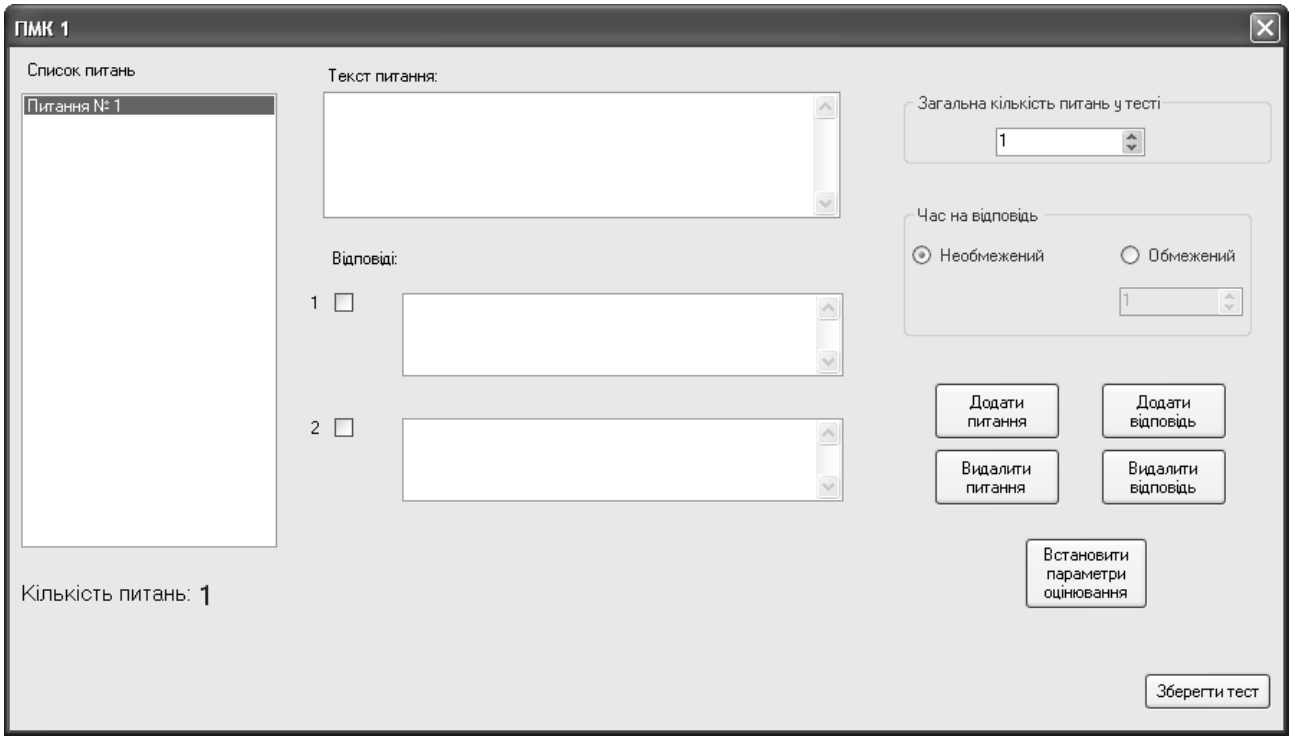

Рис. 2.4 **Вікно редагування нового тесту**

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.

При натисканні 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.

При вході в програму виводиться інформація, що повідомляє користувача про його роль при роботи в програмі (рис. 2.5).

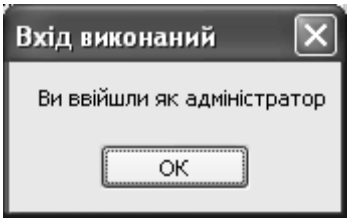

Рис. 2.5 **Повідомлення при вході в програму**

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

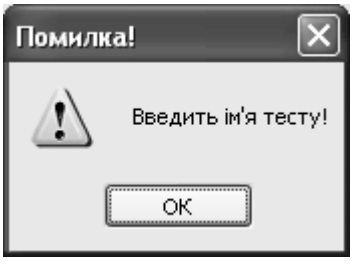

Рис. 2.6 **Повідомлення про неповну вказівку атрибутів нового тесту**

При створенні тесту, якщо не обрано жодної правильної відповіді, виводиться відповідне діалогове вікно (рис. 2.7).

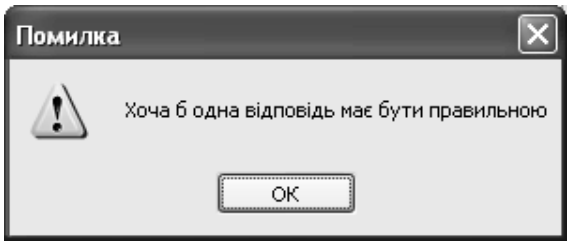

Рис. 2.7 **Повідомлення про відсутність правильної відповіді**

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації до виконання курсового проекту з навчальної дисципліни "ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

для студентів напряму підготовки 6.050101 "Комп'ютерні науки" всіх форм навчання

Укладач **Парфьонов** Юрій Едуардович

Відповідальний за випуск **Пономаренко В. С.**

Редактор **Промський М. Н.**

Коректор **Маркова Т. А.**

План 2015 р. Поз. № 61.
Підп. до друку 29.12.2014 р. Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 2,75. Обл.-вид. арк. 3,44. Тираж 50 прим. Зам. № 340.

Видавець і виготівник – видавництво ХНЕУ ім. С. Кузнеця, 61166, м. Харків, пр. Леніна, 9-А

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***