МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

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

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

Харків. Вид. ХНЕУ, 2010

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 1 від 26.08.2009 р.

М54 Методичні рекомендації до виконання курсового проекту з навчальної дисципліни "Об’єктно-орієнтоване програмування" для студентів напряму підготовки "Комп’ютерні науки" всіх форм навчання / укл. Ю. Е. Парфьонов, О. В. Щербаков, М. Ю. Лосєв, В. М. Федорченко. – Харків : Вид. ХНЕУ, 2010. – 52 с. (Укр. мов.)

Викладено питання організації курсового проектування з навчальної дисципліни, подано структуру курсового проекту, методичні рекомендації до розробки його структурних елементів. Наведено вимоги до оформлення пояснювальної записки курсового проекту.

Рекомендовано для студентів напряму підготовки "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 неможливе 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 потрібує 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 будь-якою мовою 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 з 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, мовою 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

Відповідно до навчального плану вивчення дисципліни "Об’єктно-орієнтоване програмування" включає лекційні та лабораторні заняття. Завершується вивчення дисципліни розробкою і захистом курсового проекту. Студент допускається до здачі екзамену з дисципліни "Об’єктно-орієнтоване програмування" тільки після успішного захисту курсового проекту.

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 беруть участь у викладанні цієї дисципліни. Керівник курсового проекту рекомендує студенту основну літературу, орієнтує його на розробку необхідних проектних рішень.

Тематика курсових проектів відповідає програмі навчальної дисципліни "Об’єктно-орієнтоване програмування". Студенту може бути призначена тема курсового проекту з переліку рекомендованих тем. Також йому надається право самостійного вибору теми проекту з урахуванням його схильностей і можливостей найбільш повно застосувати отримані знання. Якщо тема пропонується студентом, то вона повинна бути обговорена й погоджена з керівником курсового проекту.

Для затвердження обраної теми курсового проекту студент подає заяву на ім’я завідувача кафедри інформаційних систем. Після затвердження обраної теми студентові видається завдання на курсове проектування.

У завданні наводиться 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 наведений у додатку А.

Після вивчення літературних джерел студент складає попередній план виконання курсового проекту, обговорює його з керівником. У процесі обговорення уточнюються 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 додатка, програму-інсталятор, презентацію доповіді для захисту в електронному вигляді, студент здає на перевірку керівникові за тиждень до строку захисту.

Захист курсових проектів організовується кафедрою інформаційних систем у комісіях за графіком, затвердженим завідувачем кафедри.

Під час захисту студент коротко доповідає про поставлене йому завдання, проектні рішення, що були прийняті, одержані результати, відповідає на питання членів комісії. Демонстрація роботи розробленої студентом програми на контрольному прикладі та презентація розроблених проектних рішень є обов’язковими.

3. Структура та обсяг курсового проекту

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

Обсяг пояснювальної записки – приблизно 32 – 38 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	Кількість сторінок
Титульний аркуш	1
Завдання на курсове проектування	2
Реферат	1
Зміст	1
Вступ	1
1. Специфікація проекту	7-9
1.1. Призначення розробки та підстава для її виконання	0,5
1.2. Постановка завдання	1-2
1.3. Вимоги до програми	1-2
1.4. Вимоги до програмної документації	0,5
1.5. Структура програми	2
1.6. Етапи розробки	2
2. Програмна документація	17-21
2.1. Порадник системного програміста	7-9
2.1.1. Архітектура програми	5-6
2.1.2. Установка та перевірка програми	2-3
2.2. Порадник користувача	10-12
2.2.1. Призначення програми	1
2.2.2. Виконання програми	7-8
2.2.3. Повідомлення оператору	2-3
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 – юридичну особу (організацію) або фізичну особу;

повна назва документа;

підписи відповідальних осіб, включаючи керівника роботи;

рік складення пояснювальної записки.

Приклад титульного аркуша наведений у додатку Б.

Реферат – це короткий виклад змісту пояснювальної записки, що містіть 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 [1]:

відомості про обсяг звіту, кількість частин звіту, кількість ілюстрацій, таблиць, додатків, кількість джерел згідно з переліком посилань (усі відомості наводять, включаючи дані додатків);

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’єкта дослідження або розроблення.

Реферат належить виконувати обсягом не більше, 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, пунктів і підпунктів 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 предметній галузі, що розглядається в курсовому проекті, роль засобів збереження даних у інформаційних системах, мету та завдання 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 коротку характеристику призначення розробки, постановку завдання, вимоги до програми та програмної документації, структуру додатку та етапи його розробки.

У підрозділі 1.1 "Призначення розробки та підстава для її виконання" вказується:

функціональне та експлуатаційне призначення програми;

документ, на підставі якого виконується розробка, та організація, що затвердила цей документ.

У підрозділі 1.2 "Постановка завдання" наводиться текстовий опис постановки завдання на розробку програми, що було видано студенту.

У підрозділі 1.3 "Вимоги до програми" містяться вимоги до графічного інтерфейсу користувача, архітектури, функціональності та вихідного коду програми.

Вимоги до програми, що розробляється відповідно до будь-якої теми курсового проекту з переліку рекомендованих тем, наведені нижче.

Програма виконується мовою C# з використанням бібліотеки класів програмної платформи Microsoft .NET. Відомості щодо основних елементів цієї бібліотеки, які використовуються в курсовому проекті, наведені в додатку Е.

Вимоги до графічного інтерфейсу користувача:

1. Назви елементів інтерфейсу повинні виконуватися українською або російською мовою.

2. Головне вікно додатка – фрейм з такими елементами:

- панель меню з підтримкою "акселераторів";
- користувальницька піктограма системного меню;
- панель інструментів з підтримкою спливаючих "підказок" для кнопок;
- рядок стану, в якому має відображатися інформація про основні режими роботи додатка.

3. Дані бази повинні відображатися у табличному вигляді (компонент DataGridView).

4. Наявність модального діалогового вікна "Про програму" з інформацією про розроблювача програми, зокрема з його (її) фотографією.

Вимоги до архітектури програми:

1. Використання не менше одного стандартного контейнерного класу.

2. Використання файлового введення-виведення даних.

3. Використання механізму виключень для обробки помилок введення-виведення даних.

Вимоги до функціональності додатка:

1. Створення файла бази даних (ім’я файла бази та каталог файлової системи для його збереження вибираються користувачем з використанням відповідного діалогового вікна).

2. Читання всіх даних з файла бази (каталог файлової системи та ім’я файла бази вибираються користувачем з використанням відповідного діалогового вікна) та їх відображення.

3. Додавання елементу даних до файла бази.

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

5. Видалення будь-якого елемента даних з файла.

6. Сортування інформації, що відображується в графічному інтерфейсі користувача, за різними реквізитами.

7. Фільтрація інформації, що відображується в графічному інтерфейсі користувача, за різними критеріями.

8. Отримання та відображення підсумкової інформації.

9. Забезпечення перевірки допустимості даних, що вводяться користувачем, з використанням компоненту ErrorProvider.

10. Видача користувачу попереджуючих та інформаційних повідомлень.

Вимоги до вихідного коду додатка

1. Вихідний код кожного з класів програми повинен міститися в окремому файлі.

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

3. Виконання угод щодо запису тексту програм мовою програмування C# (див. додаток Ж).

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

У підрозділі 1.4 "Вимоги до програмної документації" описується склад програмної документації відповідно до структури пояснювальної записки курсового проекту та, за необхідності, спеціальні вимоги до неї.

У підрозділі 1.5 "Структура програми" мають бути наведені відомості про структуру програми, її складові частини та зв’язки між ними.

У підрозділі 1.6 "Етапи розробки" вказуються необхідні етапи розробки, перелік та зміст робіт за кожним з етапів, а також строки виконання етапів та робіт.

Другий розділ "Програмна документація" складається з двох підрозділів: 2.1 "Порадник системного програміста" та 2.2 "Порадник користувача".

Підрозділ 2.1 "Порадник системного програміста" містить опис розроблених студентом проектних рішень, архітектури розробленого додатку, відомості про призначення, настройку та перевірку програми.

У пункті 2.1.1 "Архітектура програми" має знаходитися UML-діаграма класів програми, докладний опис їх призначення, ієрархії, елементів та взаємодії. Даний пункт має завершуватися описом схеми алгоритму реалізації одного з методів програми, що був розроблений студентом. Вибір цього методу узгоджується з керівником курсового проекту. Приклад опису архітектури програми наведений у додатку И.

У пункті 2.1.2 "Установка та перевірка програми" наводиться опис апаратних та програмних засобів, необхідних для функціонування розробленої програми, дій щодо інсталяції програми на комп'ютері користувача, способів перевірки, які дозволяють зробити загальний висновок про працездатність програми (контрольні приклади, методи прогону, результати тощо).

Підрозділ 2.2 "Порадник користувача" містить призначення програми, порадник щодо її використання за призначенням, опис повідомлень користувачу, що можуть з'явитися у процесі роботи програми.

У пункті 2.2.1 "Призначення програми" вказуються відомості про призначення програми та інформація, достатня для розуміння функцій програми.

У пункті 2.2.2 "Виконання програми" має бути вказана послідовність дій оператора, яка забезпечує запуск, виконання та завершення програми, наведено опис функцій, формату та можливих варіантів команд, за допомогою яких оператор керує виконанням програми, а також реакцію програми на ці команди.

У пункті 2.2.3 "Повідомлення оператору" наводяться екранні форми повідомлень, що можуть з'являтися в ході виконання програми, опис їх змісту та дії оператора при появі таких повідомлень.

Приклад порадника користувача наведений у додатку К.

У висновках наводять оцінку одержаних результатів роботи (негативних також); можливі галузі використання результатів роботи; народногосподарську, 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: у порядку появи посилань у тексті, в алфавітному порядку прізвищ перших авторів або заголовків.

Приклад списку використаних джерел наведений у додатку Л.

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.

У одному з додатків повинен знаходитися вихідний код програми, яка розроблена студентом при виконанні курсового проекту. Вихідний код, який згенерований візуальним дизайнером системи програмування, наводити не треба.

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".

Далі наведені загальні вимоги до оформлення пояснювальної записки курсового проекту.

Матеріали пояснювальної записки друкуються за допомогою комп’ютерної техніки на аркушах формату А4 (210х297 мм). 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 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 документа, 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.

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 або 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 місці, 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".

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 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 Г, Є, З, І, Ї, Й, О, Ч, Ь. Один додаток позначається як додаток А.

Рекомендована література

Основна

1. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К. : Держстандарт України, 1995. – 38 с.

2. C# 2005 для профессионалов / Нейгел К., Ивьен Б., Глинн Дж. и др. ; пер. с англ. – М. : Изд. дом "Вильямс", 2006. – 1376 с.

3. Троелсен Э. C# и платформа.Net / Э. Троелсен ; пер. с англ. – СПб. : Питер, 2007 – 796 с.

4. Троелсен Э. Язык программирования C# 2005 и платформа .Net 2.0 / Э. Троелсен ; пер. с англ. – М. : Издательский дом "Вильямс", 2007. – 1168 с.

5. Шилдт Г. Полный справочник по C# / Г. Шилдт ; пер. с англ. – М. : Изд. дом "Вильямс", 2004. – 752 с.

6. Шилдт Г. C# : учебный курс / Г. Шилдт. – СПб. : Питер, 2003 – 512 с.

Додаткова

7. Andrew Troelsen. Pro C# 2008 and the .NET 3.5 Platform.:apress, 2007. – 1370 p.

Ресурси мережі Internet

8. http://msdn.microsoft.com/ru-ru/default.aspx – база знань Microsoft Developer Network російською мовою

9. www.functionx.com – материалы по C# (English)

10. www.intuit.ru – Internet- інститут інформаційних технологій

11. www.rsdn.ru – материалы по C# (Русский)

ДОДАТКИ

Додаток А

Зразок завдання на курсовий проект

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра Інформаційних систем

Дисципліна: "Об’єктно-орієнтоване програмування"

ЗАВДАННЯ

На курсовий проект
студенту 2-го курсу 5-ої групи

_____
(прізвище, ім’я, по батькові)

1. Тема проекту: _____

2. Дата здачі студентом закінченого проекту ____

3. Вхідні дані до проекту: літературні джерела, технічна документація щодо розробки програм, ДСТУ з оформлення документації.

4. Зміст пояснювальної записки:

Вступ. Специфікація проекту. Програмна документація. Висновки. Додатки.

5. Перелік графічного матеріалу: діаграма класів додатку; схема алгоритму реалізації одного з розроблених методів (за узгодженням з керівником курсового проекту).

Календарний план
виконання курсового проекту

№ з/п	Назва етапу роботи	Строк виконання	Відмітка про виконання
1	2	3	4
1	З'ясування загальної постановки завдання. Розробка чернетки першого розділу пояснювальної записки		

Закінчення додатка А

1	2	3	4
2	Деталізація завдань курсового проектування. Уточнення архітектури додатка. Розробка остаточного варіанта першого розділу пояснювальної записки		
3	Програмна реалізація, налагодження та тестування додатка. Розробка чернетки пояснювальної записки (другий розділ, висновки, списку використаних джерел, додатків)		
4	Остаточне налагодження та тестування додатка. Розробка чернетки пояснювальної записки (вступ, другий розділ, висновки, списку використаних джерел, додатків)		
5	Розробка остаточного варіанта пояснювальної записки вступ, другий розділ, висновки, списку використаних джерел, додатків)		

Дата видачі завдання ____

Керівник проекту _____ _____
(підпис) (посада, П. І. Б.)

Студент _____ _____
(підпис) (П. І. Б.)

Додаток Б

Зразок оформлення титульного аркуша

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Факультет економічної інформатики
Кафедра інформаційних систем

КУРСОВИЙ ПРОЕКТ
за темою:

"<НАЗВА ТЕМИ КУРСОВОГО ПРОЕКТУ>"

з навчальної дисципліни
"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

Виконав:
студент факультету ЕІ
5 курсу, 2 групи
Іванов В. М.

Перевірив:
к.т.н., доцент кафедри ІС
Петров І. І.

Харків, 2009

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 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 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.

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

Додаток Д

Зразок оформлення змісту пояснювальної записки

ЗМІСТ

Додаток Е

Відомості щодо використання деяких елементів бібліотеки класів програмної платформи Microsoft .NET

Обробка виключень

Проблеми "традиційного підходу" до обробки помилок під час виконання програми:

1. "Помилки можуть траплятися з іншими, але не в моєму коді".
2. "Перевірочний" код повинен перебувати всередині "корисного" коду.
3. Код із численними перевірками помилок легко стає нечитабельним.

Переваги обробки виключень порівняно з "традиційним підходом" до обробки помилок:

1. Відділення "корисного" коду від "перевірочного".
2. Групування типів помилок.
3. Можливість передачі помилки в інший контекст.

Послідовність обробки виключення подана на рис. 1:

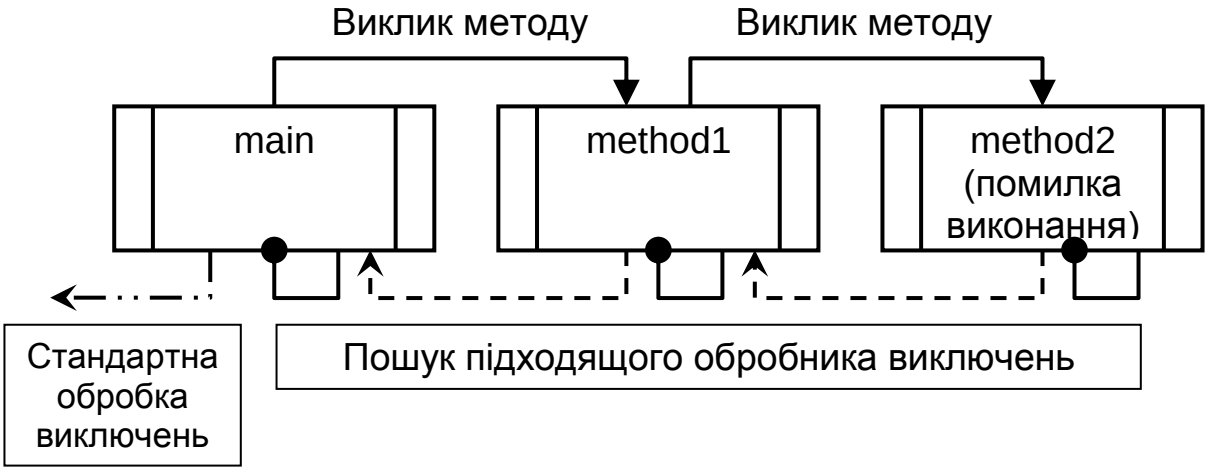

Рис. 1. **Механізм обробки виключень**

Синтаксис обробки виключень:

```
try {
    // блок, у якому може виникнути помилка
}
catch(ExceptionType1 ex) {
    //блок обробки виключення типу ExceptionType1
}
……..
```

```
catch(ExceptionTypeN ex) {
    // блок обробки виключення типу ExceptionTypeN
}
// необов'язковий блок
finally {
// оператори, які виконуються в будь-якому разі
}
```

Ієрархія класів системи обробки виключень бібліотеки класів платформи Microsoft .Net наведена на рис. 2.

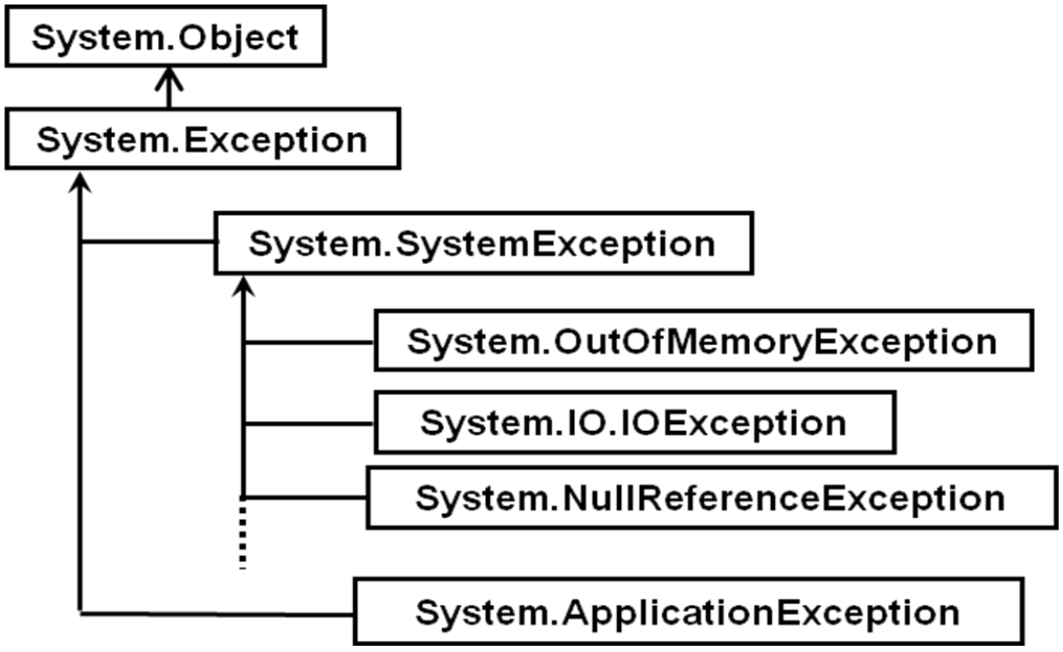

Рис. 2. **Класи системи обробки виключень Microsoft .Net**

Введення-виведення даних

Класи системи введення-виведення Microsoft .Net (простір імен System.IO) наведені на рис. 3.

Введення даних з файла за допомогою методу ReadAllLines() класу File:

```
try
{
  String[ ] Lines = File.ReadAllLines("d:\myfile.txt");
}
catch(FileNotFoundException e) { }
```

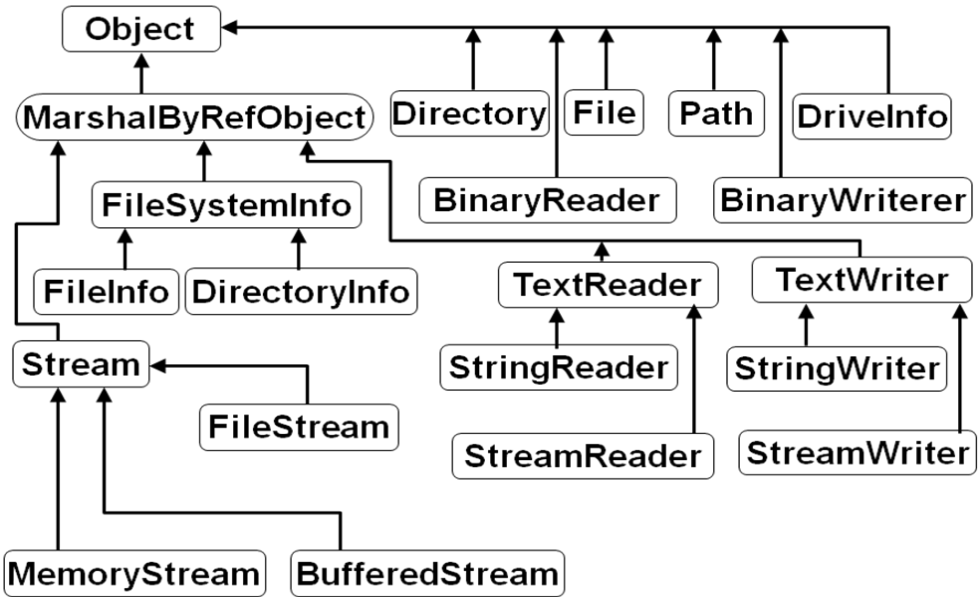

Рис. 3. **Основні класи системи введення-виведення Microsoft .Net**

Виведення даних до файла за допомогою методу WriteAllLines() класу File:

```
try
{
  File.WriteAllLines("d:\myfile.txt");
}
catch(FileNotFoundException e) { }
```

Введення даних з файла за допомогою символьного потоку введення (класу StreamReader):

```
StreamReader sw = new StreamReader(ChosenFile);
string line = null;
while ((line = sw.ReadLine()) != null)
{
   string[ ] tokens = line.Split(delimeter);
   CarList.Add(new Car(tokens[0], tokens[1], tokens[2]));
}
sw.Close();
```

Виведення даних до файла за допомогою символьного потоку виведення (класу StreamWriter):

```
StreamWriter sw = new StreamWriter(chosenFile, false);
foreach (string line in textBoxContents.Lines)
        sw.WriteLine(line);
sw.Close();
```

Короткі відомості про деякі візуальні та невізуальні компоненти Windows Forms

Середовище MS Visual Studio надає велику кількість компонентів, серед яких можна виділити елементи управління і невізуальні компоненти. Елементи управління – це візуальні компоненти, що забезпечують взаємодію між користувачем і програмою. Невізуальні компоненти не мають графічного інтерфейсу і є класами, що забезпечують додаткові функціональні можливості для додатка.

У вікні Toolbox всі компоненти розбиті на декілька груп:

елементи управління загального призначення (Common Controls) – часто використовувані елементи управління;

контейнери (Containers) – елементи управління, які можуть містити інші елементи управління;

меню і панелі інструментів (Menus & Toolbars);

дані (Data) – елементи управління для роботи з даними;

компоненти (Components) – компоненти, що не мають графічного інтерфейсу;

друк (Printing) – стандартні діалогові вікна для управління процесом друку документів;

діалоги (Dialogs) – стандартні діалогові вікна загального призначення.

Для додавання компонентів на форму можна використовувати спосіб перетягання їх мишею.

Діалогові вікна для роботи з файлами (група Dialogs)

Компоненти OpenFileDialog і SaveFileDialog є невізуальними і дозволяють використовувати в програмі стандартні діалогові вікна Windows. Ці компоненти застосовуються для отримання від користувача бажаних значень настройок, наприклад, для введення повного імені файла разом з шляхом пошуку.

Компонент "Відкрити файл" (OpenFileDialog) призначений для вибору файла з метою подальшого відкриття. Компонент SaveFileDialog виконує діалог "Зберегти файл як…". Усі властивості цих компонентів однакові, тільки їх сенс декілька різний для відкриття і закриття файлів.

Основна властивість, в якій повертається у вигляді рядка вибраний користувачем файл, – FileName.

Приклад: обробник пункту меню "Зберегти".

```
private void saveMenuItem_Click(object sender, EventArgs e)
{
   string FName;
   saveFileDialog1.Filter = "Текстові файли |*.txt|Всі файли|*.*";
   if ( saveFileDialog1.ShowDialog() == DialogResult.OK)
   {
     FName = saveFileDialog1.FileName;
     StreamWriter sw = new StreamWriter(FName);
     sw.Write(textBox1.Text);
     sw.Close();
    }
}
```

Приклад: обробник пункту меню "Відкрити".

```
private void OpenMenuItem_Click(object sender, EventArgs e)
{
   string FName="";
   openFileDialog1.Filter = "Текстові файли |*.txt|Всі файли|*.*";
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
     FName = openFileDialog1.FileName;
     StreamReader sr = new StreamReader(FName);
     textBox1.Text = sr.ReadToEnd();
     sr.Close();
   }
}
```

Елемент управління "Таблиця" (група Data)

Елемент управління "Таблиця" (клас DataGridView) призначений для виведення на форму даних у табличному вигляді. Він володіє величезною функціональністю – реалізує 118 власних подій на додаток до 69 подій, успадкованих від класу Control.

Додавання колонок

Додавання колонок – крок, абсолютно необхідний перед тим, як таблиця буде пред’явлена кінцевому користувачеві. Якщо не додати

жодної колонки, DataGridView буде тьмяно-сірим прямокутником на формі.

Продовження додатка Е

абсолютно позбавлений якої-небудь функціональності. Існує ряд способів додавання колонок. У найзагальнішому випадку є чотири можливих сценарії:

1. Є джерело даних, воно доступне під час розробки, і ми готові додавати колонки в цей час.

Після завдання значень властивостей *DataSource* і *DataMember* таблиця автоматично вивчає схему джерела даних і генерує по колонці для кожної колонки таблиці або властивості об’єкта, колекція яких використовується як джерело даних. Причому робить це "розумно", підбираючи не тільки відповідний заголовок колонки, але і тип колонки. Тобто якщо тип колонки буде чимось ніби int/decimal/string, то додасться колонка типу DataGridViewTextBoxColumn. А якщо така колонка матиме тип boolean, то додасться вже DataGridViewCheckBoxColumn. Можна видалити "зайві" колонки, поправити текст заголовка, а також тип колонки. От як це робиться. Після завдання значень властивостей *DataSource* і, за необхідності, *DataMember,* вже є колонки, що згенеровані по описаному вище алгоритму. Виділивши таблицю в дизайнерові, необхідно натиснути її "розумний ярлик" (smart tag). "Розумний ярлик" знаходиться у верхньому правому кутку елемента управління і надає доступ до меню, склад елементів якого можна охарактеризувати як "найбільш часто використовувані настройки".

Меню дозволяє робити з таблицею багато цікавих речей, але в даному сценарії найцікавішим є пункт Edit Columns... При виборі цього пункту відкривається діалог редагування колонок.

У ньому можна видалити зайві колонки (кнопка Remove), змінити заголовок колонки (властивість HeaderText), тип колонки (властивість ColumnType) і ряд інших властивостей кожної колонки. У списку Selected Columns зліва показуються всі колонки, причому їх порядок "зверху-вниз" відповідає порядку "зліва-направо" реальної таблиці. Парою кнопок із стрілками можна міняти їх порядок в цьому списку.

Резюме: у даному сценарії основну роботу виконує візуальний дизайнер MS Visual Studio.

2. Немає джерела даних, але вже під час розробки ми знаємо склад і тип колонок, і готові додавати їх.

У цьому сценарії значення властивостей DataSource і DataMember не встановлюються, а таблиця використовується у вільному, не

прив’язаному до даних режимі. У smart tag-меню необхідно вибрати пункт Add Column...

Продовження додатка Е

Оскільки джерела даних немає (властивості DataSource і DataMember виставлені в null), перемикач Databound column не працює. Отже, доступний тільки перемикач Unbound column і підлеглі йому поля. У цьому вікні задаються наступні властивості нової колонки:

1. Name (ім’я колонки) – це ім’я створюваної у формі змінної типу DataGridViewColumn або його спадкоємця, в яку поміщається посилання на колонку, що додається.

2. Type (тип колонки) – задає елемент управління, який буде відображатися у комірках колонки:

кнопка (тип DataGridViewButtonColumn);

прапорець (тип DataGridViewCheckBoxColumn);

список (тип DataGridViewComboBoxColumn);

зображення (тип DataGridImageColumn);

гіперпосилання (тип DataGridViewLinkColumn);

текстове поле (тип DataGridViewTextBoxColum).

3. HeaderText – текст, який буде показаний в заголовку даної колонки.

Три перемикачі, що залишилися, допомагають задати додаткові властивості колонки.

Після натиснення на кнопку Add даний діалог не закривається, а пропонує нові значення за умовчанням – Column2, Column3 і т. д.

Резюме: у даному сценарії програміст указує, які колонки і в якому вигляді він хоче бачити.

3. Є джерело даних, але воно доступне тільки під час виконання, а під час розробки нічого невідомо ні про нього, ні про склад колонок.

Сценарій, що змагається по простоті з сценарієм номер 1. Якщо властивість DataGridView.AutoGenerateColumns виставлено в true (а за умовчанням так і є), то під час виконання будь-яка зміна властивостей DataSource/DataMember викликає генерацію колонок по алгоритму сценарію 1. Можна також запустити (перезапустити) цей процес генерації і додавання, встановивши згадану властивість в false, а потім повернувши її в true.

Резюме: у даному сценарії програміст користується колекцією колонок, що заздалегідь згенеровані, вносячи мінімальні зміни.

4. Немає джерела даних, а склад/тип колонок з’ясовується динамічно, під час виконання, а під час розробки невідомий тип і, можливо, навіть кількість колонок.

Продовження додатка Е

У даному випадку немає іншого виходу, окрім як скласти власну колекцію колонок. Перший крок – визначити тип колонок, які хотілося б бачити в таблиці. Оскільки колонки всіх типів (і вбудовані, і користувальницькі) додаються в таблицю однаково, в даному розділі буде розглянута робота з найпоширенішим типом колонки – DataGridViewTextBoxColumn. Робота зі всіма іншими типами колонок абсолютно аналогічна.

Першим способом програмного додавання колонок є використання методу Add() колекції колонок. Цей метод перевантажений і дозволяє додавати як готову колонку (екземпляр класу DataGridViewColumn або його спадкоємця), так і пару "ім’я – заголовок":

```
grid.Columns.Add("MyColumnName","MyColumnHeaderText
");
grid.Columns.Add(new DataGridViewColumn(...));
```

Тут grid – це посилання на об’єкт класу DataGridView.

Другий спосіб – просто встановити властивість ColumnCount таблиці в яке-небудь значення більше нуля:

```
//якщо до цього була прив’язка до джерела даних
grid.DataSource = null;
grid.ColumnCount = 5;
```

При цьому будуть створені колонки, що ініціалізовані значеннями за умовчанням.

Резюме: у даному сценарії програміст повністю відповідає за створення нових колонок і за їх внесення до колекції. Настройка нових колонок також повністю лежить на ньому.

Усі інші варіації зводяться до комбінацій цих чотирьох базисних сценаріїв.

Додавання рядків

Властивість DataGridView.Rows забезпечує доступ до колекції рядків. Користуючись нею, можна додати рядки в таблицю. Але, на відміну від чотирьох можливих сценаріїв додавання колонок, у разі додавання рядків сценарій всього один. Додати рядки в DataGridView

можна або програмно, скориставшись методом Add() колекції рядків, або підключивши до нього деяке джерело даних.

Метод Add() має чотири варіанти:

```
// додає один рядок, заповнюючи її значеннями за замовчуванням
int Add();
```

Продовження додатка Е

```
// додає один рядок, заповнюючи її значеннями з масиву values
int Add(params object[ ] values);
// додає декілька рядків, заповнюючи їх значеннями за замовчуванням
int Add(int count);
// додає заздалегідь створений рядок
int Add(DataGridViewRow dataGridViewRow);
```

DataGridView допускає наявність в одній колонці комірок різних типів. Для цього спочатку об'єкт типу DataGridViewRow повинен бути створений і заповнений окремо. І тільки тоді доданий до таблиці. При цьому кількість колонок у рядку повинна відповідати кількості колонок таблиці.

Нижче наведений приклад додавання перемикача в першу колонку третього рядка :

```
grid.DataSource = null;
// створимо 3 колонки типу DataGridViewTextBoxColumn
grid.ColumnCount = 3;
grid.Rows.Add();
grid.Rows.Add();
DataGridViewRow newRow = new DataGridViewRow();
// Створюємо комірку типу CheckBox
DataGridViewCheckBoxCell checkCell = new DataGridViewCheck-
BoxCell();
checkCell.Value = true;
// Додаємо як першу комірку нового рядка чарунку типу CheckBox
newRow.Cells.Add(checkCell);
// Решту комірок заповнюємо комірками типу TextBox
newRow.Cells.Add(new DataGridViewTextBoxCell());
newRow.Cells.Add(new DataGridViewTextBoxCell());
// цей рядок буде з перемикачем у першій колонці
```

```
grid.Rows.Add(newRow);
```

Використання класу DataTable

Як джерело даних для елементу управління DataGridView часто використовується об’єкт класу DataTable.

Клас DataTable становить одну таблицю з даними в оперативній пам’яті комп’ютера.

Продовження додатка Е

Якщо об’єкт DataTable створюється програмно, необхідно спочатку визначити його структуру, додавши об’єкти DataColumn у колекцію DataColumnCollection.

Об’єкт класу DataColumn – це один стовпець таблиці DataTable. DataTable містить колекцію об’єктів класу DataColumn, на яку посилається властивість Columns таблиці. Ця колекція стовпців разом з обмеженнями визначає схему, або структуру, таблиці.

Стовпці таблиці створюються, використовуючи конструктор класу DataColumn або викликаючи метод Add властивості Columns таблиці, яка є об’єктом DataColumnCollection. Метод Add також додає створений стовпець до колекції стовпців DataColumnCollection. При завданні типів даних для DataColumn використовуються типи .NET Framework

У наступному прикладі до об’єкта DataTable "Customers" додаються чотири стовпці з використанням методу Add. Також стовпець "CustID" визначається як первинний ключ таблиці "Customers", тобто дані в комірках цього стовпця мають бути унікальними.

```
DataTable workTable = new DataTable("Customers");
DataColumn workCol = workTable.Columns.Add("CustID",
typeof(Int32));
workTable.Columns.Add("CustLName", typeof(String));
workTable.Columns.Add("CustFName", typeof(String));
workTable.Columns.Add("Purchases", typeof(Double));
workTable.PrimaryKey=new DataColumn[ ] { workTable
.Columns["CustID"] };
```

Після створення об’єкта DataTable і визначення його структури з використанням стовпців і обмежень до створеної таблиці можна додавати нові рядки даних.

Щоб додати новий рядок, необхідно спочатку створити новий об’єкт класу DataRow за допомогою методу NewRow. Метод NewRow повертає рядок зі схемою об’єкта DataTable, згідно з тим, як вона визначена в

колекції DataColumnCollection таблиці. Максимальна кількість рядків, яка може зберігатися в об’єкті DataTable, рівна 16777216.

У наступному прикладі показано, як створити новий рядок шляхом виклику методу NewRow.

```
DataRow workRow = workTable.NewRow();
```

Після цього можна маніпулювати доданим рядком за допомогою індексу або імені стовпця, як показано в наступному прикладі.

Продовження додатка Е

```
workRow["CustLName"] = "Петренко";
```
або
```
workRow[1] = "Петренко";
```

Після вставки даних в новий рядок для додавання рядка в об’єкт DataRowCollection застосовується його метод Add, показаний в наступному коді.

```
workTable.Rows.Add(workRow);
```

Результатом передачі масиву значень типу Object у метод Add є створення нового рядку в таблиці та ініціалізація значень стовпців цього рядка, відповідними значеннями з масиву об’єктів. Дані з масиву зіставляються із стовпцями послідовно, з урахуванням порядку цих стовпців у таблиці.

У наступному прикладі відбувається додавання 10 рядків до таблиці "Customers".

```
DataRow workRow;
for (int i = 0; i <= 9; i++)
{
  workRow = workTable.NewRow();
  workRow[0] = i;
  workRow[1] = "CustName" + i.ToString();
  workTable.Rows.Add(workRow);
}
```

Використання класу DataView

Класс DataView дозволяє створювати різні представлення даних, які зберігаються в DataTable. Ця можливість часто використовується в додатках. За допомогою класу DataView можна представити дані таблиці

DataTable з різними порядками сортування і відфільтрувати їх за деякими критеріями.

DataView надає декілька способів сортування і фільтрації даних у DataTable:

1. Можна використовувати властивість Sort для завдання одного або декількох порядків сортування стовпців використовуючи параметри ASC (сортування за зростанням) та (або) DESC (сортування за убуванням).

Продовження додатка Е

2. Властивість ApplyDefaultSort можна використовувати для автоматичного створення порядку сортування за зростанням на підставі стовпця або стовпців первинного ключа таблиці. Властивість ApplyDefaultSort застосовується лише якщо властивість Sort є посиланням із значенням null або порожнім рядком, і якщо в таблиці визначений первинний ключ.
3. Властивість RowFilter можна використовувати для фільтрації даних шляхом завдання підмножин рядків на підставі значень їх стовпців.

Є два способи створення представлення даних DataView. Можна використовувати конструктор DataView або створити посилання на властивість DefaultView таблиці DataTable.

У наступному прикладі коду демонструється як створювати об’єкт класу DataView за допомогою конструктора.

DataView custDV = new DataView(workTable);

Тут workTable є посиланням на створену раніше таблицю DataTable.

Далі наведений ще один варіант використання конструктора класу DataView.

DataView custDV = new DataView(workTable,"CustLName= ‘Зайченко’", "CustID", DataViewRowState.CurrentRows);

Тут "CustLName= ‘Зайченко’" є значенням властивості RowFilter, "CustID" – це значення властивості Sort, DataViewRowState.CurrentRows – значеня перерахування DataViewRowState, яке означає, що об’єкт DataView використовує всі поточні рядки таблиці DataTable.

У наступному прикладі коду демонструється, як отримувати посилання на об’єкт DataView за допомогою властивості DefaultView таблиці "Customers".

DataView custDV = workTable.DefaultView;

У наступному прикладі об’єкт DataView створюється на основі таблиці "Contact" (посилання contacts), а потім встановлюється значення властивості RowFilter для повернення рядків, що містять контакти з прізвищем "Іванченко".

Продовження додатка Е

DataView view = contacts.DefaultView;
view.RowFilter = "LastName=‘Іванченко’";

Фільтр для об’єкта DataView можна очистити після його завдання двома різними способами за допомогою властивості RowFilter:

1. Встановити властивість RowFilter у значення null.
2. Встановити значення властивості RowFilter рівним порожньому рядку.

Об’єкт DataView надає декілька способів сортування і повернення рядків даних, впорядкованих по певних критеріях.

Після створення об’єкта DataView для завдання сортування можна використовувати його властивість Sort.

У наступному прикладі об’єкт DataView також створюється на основі таблиці "Contact", потім виконується сортування по прізвищах у зростаючому порядку, а потім по іменах в убиваючому порядку:

DataView view = contacts.DefaultView;
view.Sort = "LastName desc, FirstName asc";

Дані сортування для об’єкта DataView можна очистити після їх завдання за допомогою властивості Sort. Існує два способи очищення даних сортування:

1. Встановити властивість Sort у значення null.
2. Встановити значення властивості Sort рівним порожньому рядку.

Використання класу ErrorProvider

Компонент ErrorProvider призначений для перевірки даних, які користувач увів у елемент управління форми. Використання компоненту ErrorProvider – кращий варіант вибору порівняно з відображенням повідомлення про помилку у відповідному вікні, оскільки після закриття цього вікна повідомлення про помилку більш не відображується.

Компонент ErrorProvider відображає значок помилки поруч з відповідним елементом управління, наприклад текстовим полем; при наведенні курсору миші на значок помилки з'являється візуальна "підказка", в якій відображується рядок повідомлення про помилку.

Закінчення додатка Е

Компонент ErrorProvider може використовуватися для відображення значків помилок при введенні користувачем неправильних даних. У формі має бути принаймні два елементи управління для переходу між ними та виклику в такий спосіб коду перевірки.

Для відображення значка помилки в разі введення в елемент управління недопустимих даних необхідно:

1. Додати два елементи управління, наприклад текстових полів, на форму.
2. Додати компонент ErrorProvider на форму.
3. Вибрати перший елемент управління і додати код до його обробника події Validating. Для правильного виконання цього коду метод-обробник має бути зв'язаний з подією.

Наступний код перевіряє допустимість даних, що були введені користувачем, і якщо ці дані є неправильними (в даному випадку нецілими числами), викликає метод SetError класу ErrorProvider. Перший аргумент методу SetError вказує, поруч з яким елементом управління розташовувати значок помилки. Другий аргумент визначає текст повідомлення про помилку, що буде відображуватися.

```
protected void textBox1_Validating (object sender,
  System.ComponentModel.CancelEventArgs e)
{
  try
  {
    int x = Int32.Parse(textBox1.Text);
    errorProvider1.SetError(textBox1, "");
```

```
  }
  catch (Exception e)
  {
    errorProvider1.SetError(textBox1, "Введіть ціле число!");
  }
}
```

Додаток Ж

Угоди щодо запису тексту програм мовою програмування C#

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 простірів імен, класів, інтерфейсів, полів, методів, об’єктів починаються з великої літери:

int FirstNumber, SecondNumber;

int GetResult();

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

const int STRING_LENGTH = 100;

Додаток И

Приклад опису архітектури програми (в скороченні)

Архітектура програми наведена на рис. 2.1.

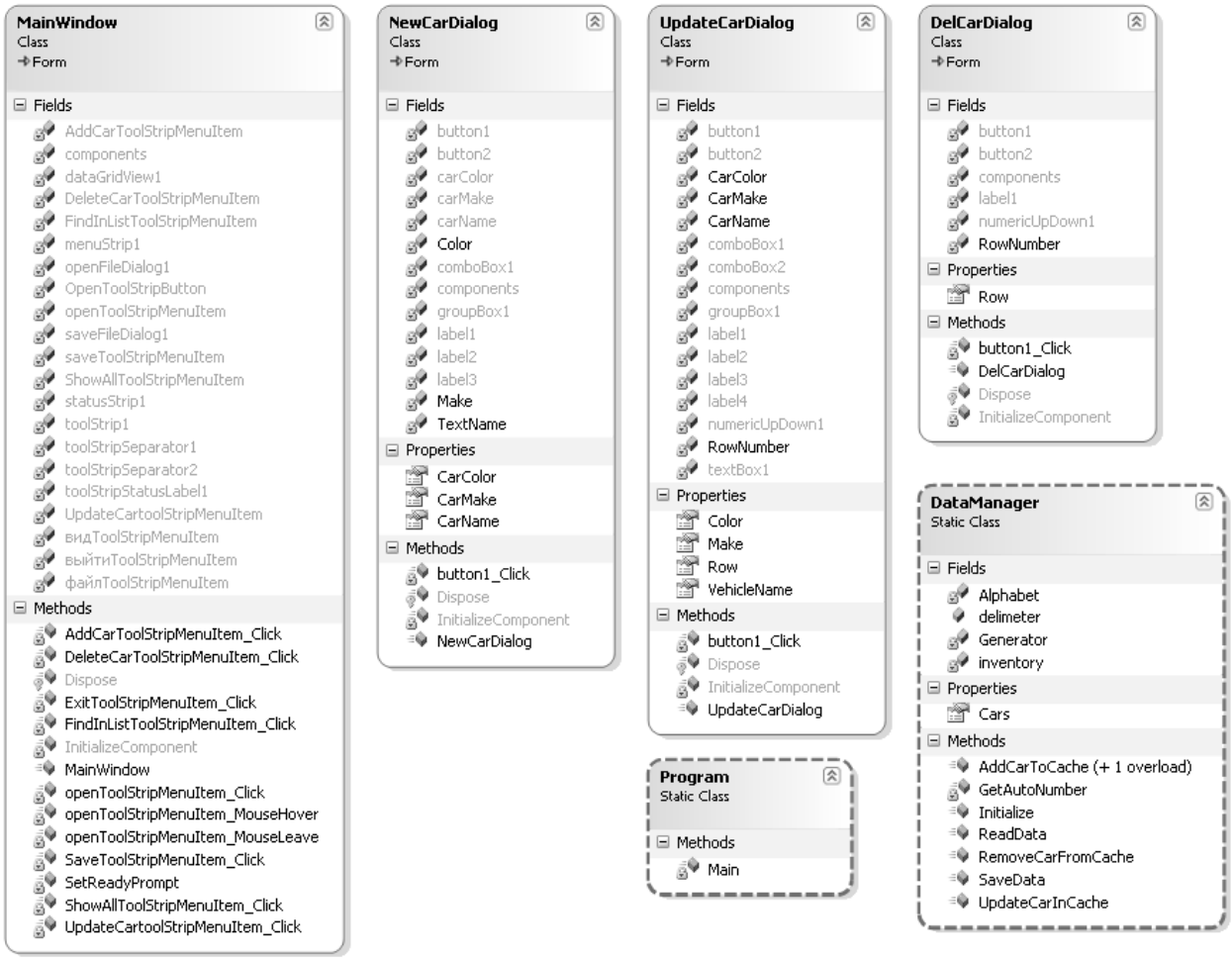

Рис. 2.1 - Діаграма класів

Програма складається з шести класів. Чотири з них: MainWindow, NewCarDialog, UpdateCarDialog, DelCarDialog є похідними від класу System.Windows.Forms.Form, тобто мають графічний інтерфейс.

Клас MainWindow становить головне вікно програми.

Поля даного класу відповідають пунктам меню, панелі інструментів програми та іншим елементам управління головного вікна програми. Кожне з них – це посилання на об’єкт класу із простору імен System. Windows.Forms. Найважливішим елементом управління в головному вікні є таблиця, яка відображує дані про автомобілі. Вона є об’єктом класу System.Windows.Forms.DataGridView (поле dataGridView1).

Продовження додатка И

Методи класу головним чином є обробниками подій елементів управління головного вікна програми. Метод InitializeComponent призначений для ініціалізації полів, які відповідають елементам графічного інтерфейсу. В класі MainWindow також є конструктор без параметрів у якому, зокрема, викликається метод InitializeComponent.

Клас NewCarDialog є діалоговим вікном для введення даних про новий автомобіль. Майже всі поля даного класу відповідають елементам управління діалогового вікна. Поля TextName, Make, Color призначені для збереження даних про назву, марку та колір автомобіля відповідно. У класі NewCarDialog також є відкрити властивості CarName, CarMake, CarColor, які призначені для доступу до значень полів TextName, Make, Color з інших класів.

У конструкторі даного класу викликається метод InitilizeComponent, у якому відбувається ініціалізація полів – елементів графічного інтерфейсу користувача.

Обробник події натискання на командну кнопку button1_Click призначений для присвоювання полям TextName, Make, Color значень, що були введені користувачем у цьому діалоговому вікні.

Клас UpdateCarDialog є діалоговим вікном для оновлення даних про автомобіль. Призначення його полів, методів та властивостей аналогічно відповідним елементам класу NewCarDialog. Крім того він має поле RowNumber для збереження номеру запису про автомобіль, який необхідно оновити.

Відкрита властивість Row призначена для доступу до значення поля RowNumber з інших класів.

Клас DelCarDialog є діалоговим вікном, що призначено для видалення даних про автомобіль. Він має поле RowNumber для збереження номеру запису про автомобіль, який необхідно видалити. Відкрита властивість Row призначена для доступу до значення поля RowNumber з інших класів.

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

Його головним полем (inventory) є об'єкт класу System.Data.Data Table, який становить кеш у оперативній пам'яті. Цей кеш має структуру двовимірної таблиці. У кожному з його рядків

Закінчення додатка И

зберігаються дані про деякий автомобіль. Для доступу до значення поля inventory з інших класів призначена відкрита властивість Cars.

У даному класі є допоміжні поля Generator, Alphabet та delimeter.

Поле Generator є об'єктом класу System.Random та становить генератор псевдовипадкових чисел. Поле Alphabet є масивом для збереження великих літер латинського алфавіту.

Ці поля використовуються при формуванні псевдовипадкового рядку алфавітно-цифрових символів.

Поле delimeter необхідно для організації введення даних у текстового файла до кешу inventory.

Метод Initialize призначений для додавання даних у масив Alphabet та встановлення загальних параметрів кешу inventory, зокрема кількості та типів його стовпцов.

У методі GetAutoNumber з використанням полів Generator та Alphabet формується рядок алфавітно-цифрових символів, який використовується як ідентифікатор запису в кеші inventory.

Методи AddCarToCache, RemoveCarFromCache, UpdateCarInCache класу DataManager призначені для додавання даних про автомобіль до кешу inventory, видалення цих даних з кешу та оновлення даних про автомобіль в кеші відповідно.

Метод SaveData необхідний для збереження даних, що знаходяться в кеші, в текстовий файл. Метод ReadData – для читання даних про автомобілі з текстового файла в кеш.

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 у 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 порадника користувача (в скороченні)

2.2 Порадник користувача

2.2.1 Призначення програми

Програма призначена 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.

2.2.2 Виконання програми

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. 2.2).

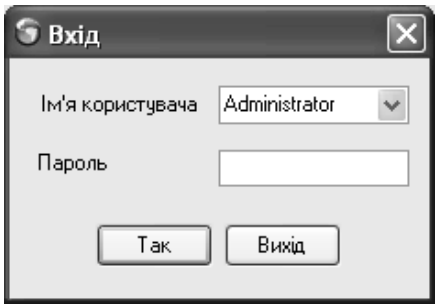

Рис. 2.2 - Вікно входу в програму

Продовження додатка К

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. 2.3) та додаткових діалогових вікон.

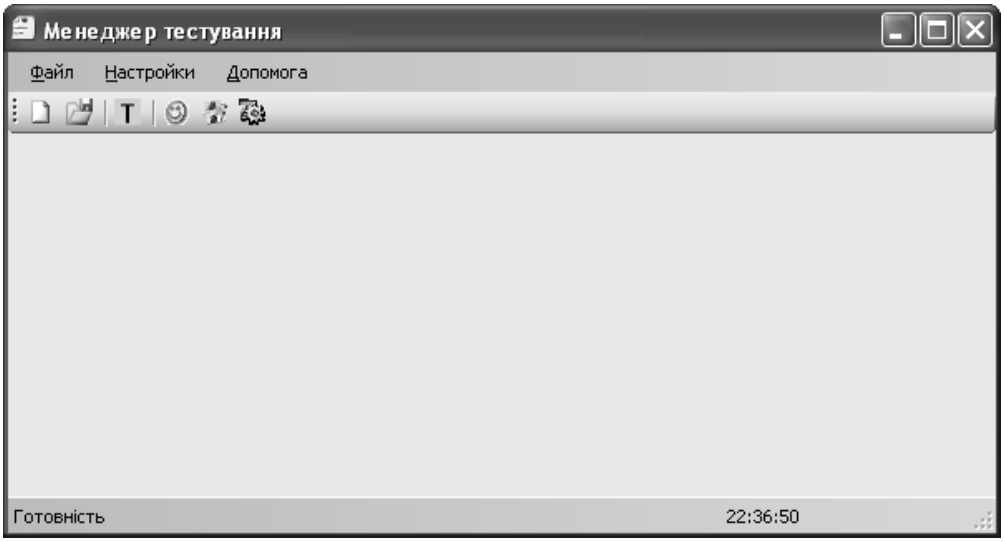

Рис. 2.3 - Головне вікно програми

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";

Продовження додатка К

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. 2.4, де користувачеві необхідно вибрати тип тесту, ввести його назву, вибрати шкалу оцінювання та ввести назву дисципліни, до якої відноситься тест.

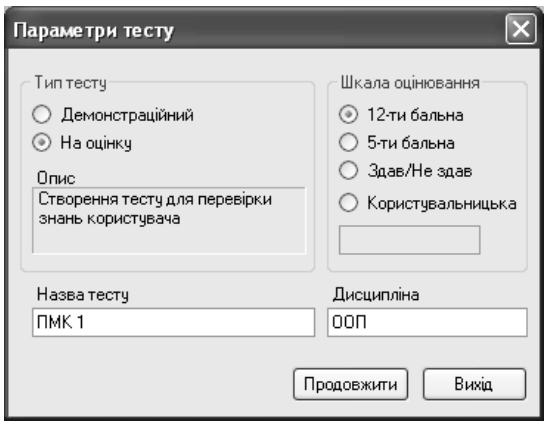

Рис. 2.4 - Вікно параметрів нового тесту

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. В даному випадку група елементів управління "Шкала оцінювання" буде недоступною.

Продовження додатка К

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. 2.5).

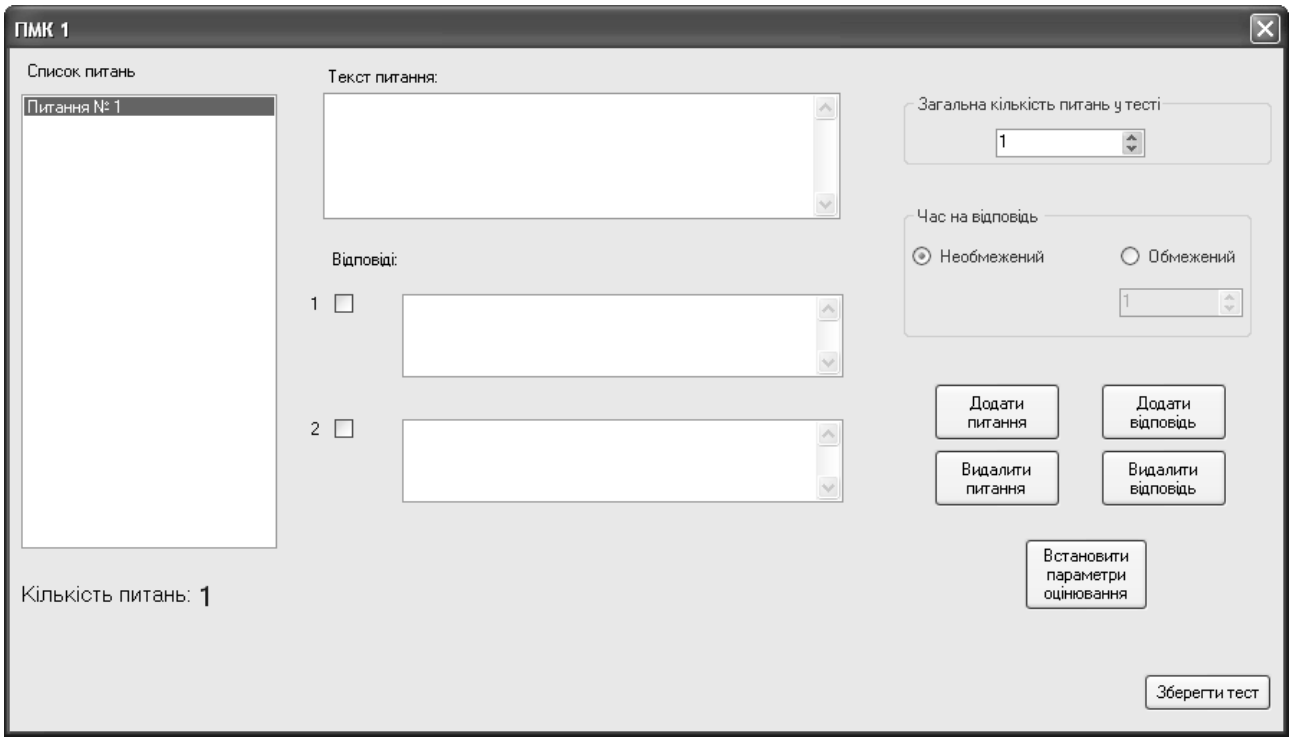

Рис. 2.5 - Вікно редагування нового тесту

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.

Редагування тесту

Для редагування існуючого тесту необхідно обрати пункт меню "Змінити тест". Після виконання цієї дії з'являється стандартне діалогове вікно відкриття файла (рис. 2.6).

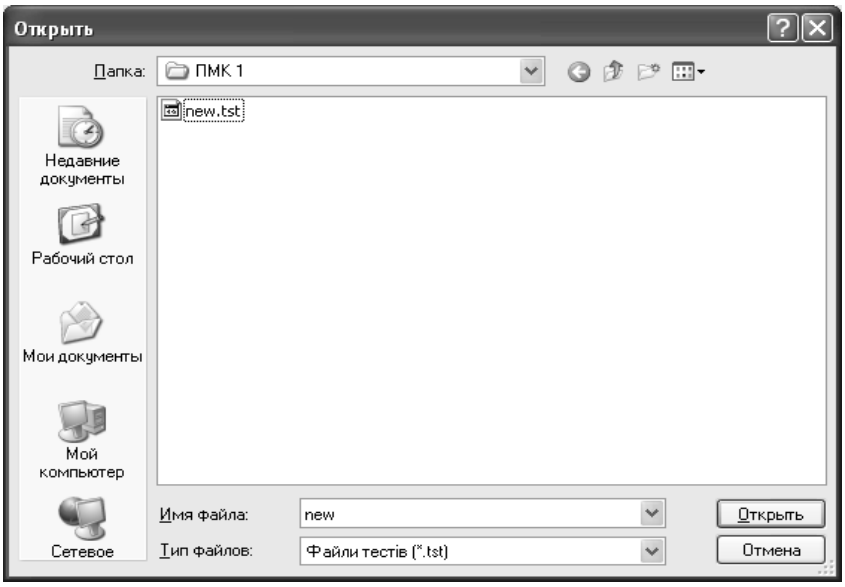

Рис. 2.6 - Відкриття файла існуючого тесту

Після відкриття існуючого тесту робота з ним не відрізняється від пункту порадника користувача "Створення нового тесту".

Виконання тестування

Для запуску режиму тестування необхідно вибрати пункт меню "Почати тест". Після цього з'являється діалогове вікно "Настройки тестування" (рис. 2.7).

Продовження додатка К

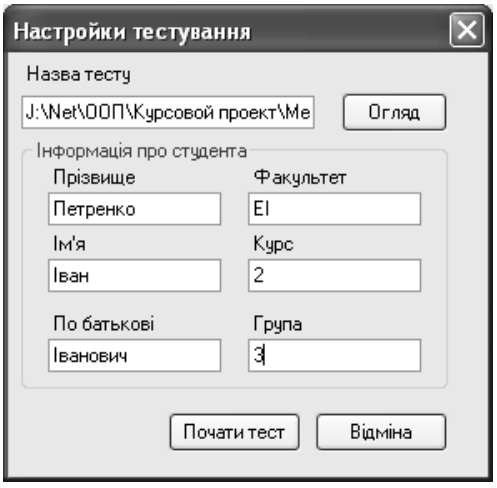

Рис. 2.7 - Вікно настройок тестування

У ньому студентові необхідно вибрати файл із тестом, натиснувши на кнопку "Огляд" та ввести персональні дані.

Для початку тестування потрібно натиснути кнопку "Почати тест". Після цього програма починає функціонувати в режимі тестування та керування передається відповідному вікну (рис. 2.8).

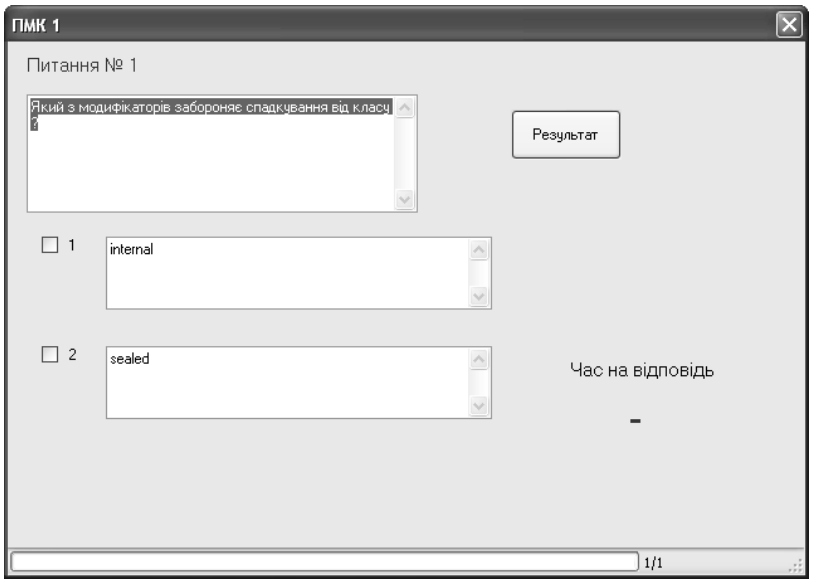

Рис. 2.8 - Вікно тестування

Відповідь на будь-яке питання тесту здійснюється шляхом проставляння "галочки" напроти правильної відповіді.

Якщо розробником тесту встановлені обмеження на максимальний час відповіді на питання, у даному вікні відображується час, що залишився для відповіді.

Продовження додатка К

Смуга прогресу, яка знаходиться в нижній частині вікна, показує студентові, на яку кількість питань він уже дав відповідь, що дозволяє ефективно розподіляти час тестування.

По завершенні тестування у вікні з’являється кнопка "Результат", що виводить на екран інформаційне вікно "Результат тестування" (рис. 2.9).

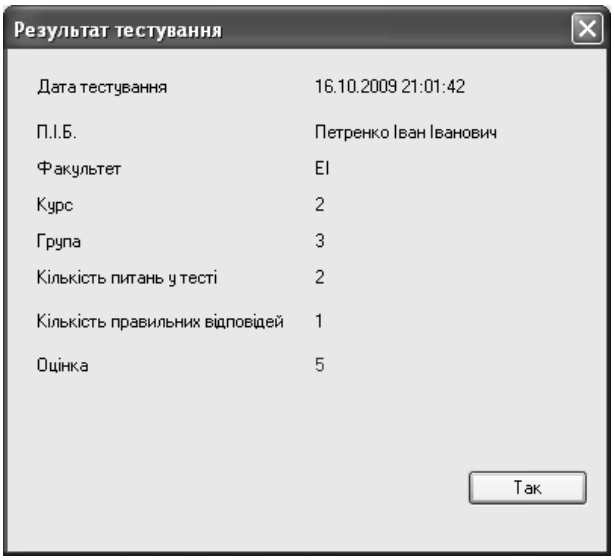

Рис. 2.9 - Діалогове вікно виведення результатів тестування

Після завершення тестування автоматично генерується звіт про проведене тестування у вигляді текстового файла з тією ж інформацією, що представлена у вікні "Результат тестування".

Вихід з програми

Для виходу з програми необхідно вибрати пункт меню "Вихід".

Зміна вигляду інтерфейсу користувача

Для цього призначений пункт меню "Вид".

Пункт цього меню "Панель інструментів" відображує або скриває панель інструментів головного вікна програми.

За допомогою пункту меню "Рядок стану" виконуються аналогічні дії з рядком стану головного вікна програми.

Пункт меню "Годинник" призначений для включення або відключення відображення годинника у панелі стану.

При виборі пункту меню "Менеджер облікових записів" з’являється вікно менеджера облікових записів (рис. 2.10).

Продовження додатка К

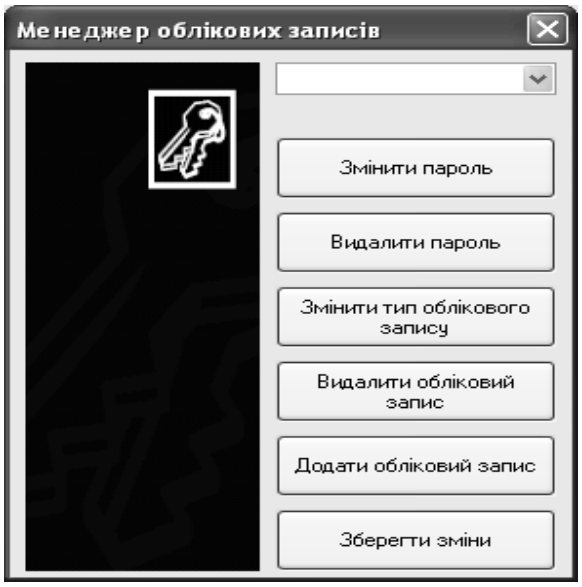

Рис. 2.10 - Діалогове вікно менеджера облікових записів

Використання елементів управління цього вікна дозволяє змінити пароль існуючого користувача, видалити пароль облікового запису користувача, змінити його тип, видалити пароль користувача, додати нового користувача та зберегти цю інформацію.

Отримання довідкової інформації

Для цього є меню програми **"**Допомога**"**. Воно містіть пункт **"**Про програму", при виборі якого з'являється діалогове вікно "Про програму". Воно надає коротку інформацію про програмний продукт, його версію та розробника (рис. 2.11).

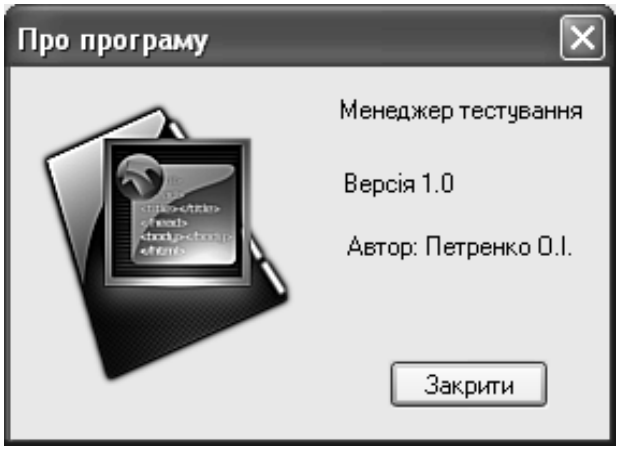

Рис. 2.11 - Вікно інформації про програму

Закінчення додатка К

2.2.3 Повідомлення оператору

Під час роботи програми можуть з’являтися інформаційні вікна щодо виникнення певних ситуацій, про які необхідно повідомити користувача. Далі наведений опис цих інформаційних повідомлень.

При вході в програму виводиться інформація, що повідомляє оператора про його роль при роботі в програмі (рис. 2.12).

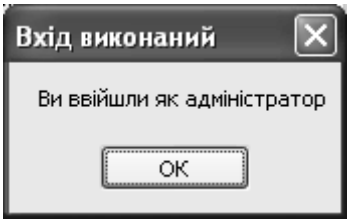

Рис. 2.12 - Повідомлення при вході в програму

При створенні нового тесту необхідно вказувати всі його атрибути у відповідних полях уведення, інакше подальша робота з тестом буде неможлива (рис. 2.13).

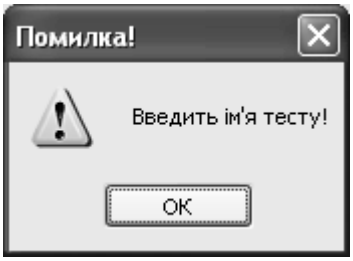

Рис. 2.13 - Повідомлення про неповну вказівку атрибутів нового тесту

При створенні тесту, якщо не обрано жодної правильної відповіді, виводиться відповідне діалогове вікно (рис. 2.14).

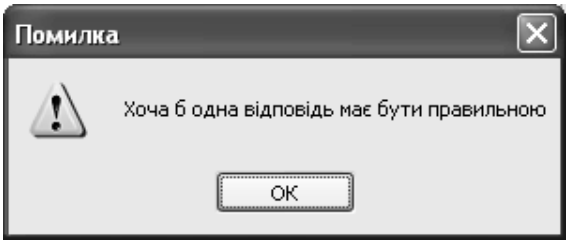

Рис. 2.14 - Повідомлення про відсутність правильної відповіді

Додаток Л

Приклад списку використаних джерел

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андреев, В. В. Как организовать делопроизводство на предприятии / В. В. Андреев. – М. : ИНФРА-М, 1997. – 94 с.

2. Белов, А. В. Финансы и кредит: учеб. / А. В. Белов, В. Н. Николаев ; КНУ им. Т. Г. Шевченко. – К. : Университет, 2004. – 215 с.

3. Агафонова, Н. Н. Гражданское право : учеб. пособие / Н. Н. Агафонова, Т. В.Богачева, Л. И. Глушкова ; под общ. ред. А. Г. Калпина ; МОН Украины. – 2-е изд., перераб. и доп. – Х. : Фактор, 2000. – 542 с.

4. Елементи інформатики : довідник / В. С. Височанський, А. І. Кардаш, В. С. Костев, В. В. Черняхівський. – К. : Наук, думка, 2003. – 192 с.

5. Коротковолновые антенны : учеб. пособие / Г. 3. Айзенберг, С. П. Белоусов, Я. М. Журбин и др. ; под общ. ред. А. А. Стогния. – 2-е изд. – М. : Радио и связь, 2003. – 192 с.

6. Нойман, Э. Происхождение и развитие сознания : пер. с англ. – К. : Ваклер ; М. : Реал-бук, 1998. – 462 с.

7. Набіулін С. Н. Вибір раціональних методів моделювання інформаційних систем // Управління розвитком. – 2009. – №4. – С.41-43.

8. Основные направления исследований, основанные на семантическом анализе текстов [Электронный ресурс] / С.-Петерб. гос. ун-т, фак. прикладной математики – процессов управления. – Режим доступа : \www/ URL: http://apmath.spdu/ru/ru/staff/tuzov/onapr.html/

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації
до виконання курсового проекту
з навчальної дисципліни

"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

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

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

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

Редактор **Голінська О. Г.**

Коректор **Муштай Т. О.**

План 2010 р. Поз. № 198.
Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 3,25. Обл.-вид. арк. 4,06. Тираж прим. Зам. №_____
Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а
Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи
Дк № 481 від 13.06.2001 р.

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

для студентів напряму підготовки
"Комп’ютерні науки"
всіх форм навчання