МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Робоча програма навчальної дисципліни

"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

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

Харків. Вид. ХНЕУ, 2008

Затверджено на засіданні кафедри інформаційних систем.
Протокол №1 від 28.08.2007 р.

Р78 Робоча програма 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. Лосев, В. М. Федорченко. — Харків: Вид. ХНЕУ, 2008. — 48 с. (Укр. мов.)

Подано тематичний план навчальної дисципліни та її зміст за модулями й темами, вміщено плани лекцій і лабораторних занять, матеріал щодо закріплення знань (індивідуальне навчально-дослідне завдання, самостійна робота, контрольні запитання), методичні рекомендації та оцінювання знань студентів.

Рекомендовано для студентів напряму підготовки "Комп’ютерні науки".

Вступ

Сьогоднішні умови господарювання вимагають від фахівців з економічного управління всебічного використання новітніх інформаційних технологій. Широкі можливості комп’ютеризованих засобів в 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 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 потребує 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 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.

Предметом вивчення дисципліни є принципи об‘єктно-орієнтованого програмування, а також методи їх використання при розробці додатків.

Необхідним елементом успішного засвоєння навчального матеріалу дисципліни є самостійна робота студентів з технічною літературою, та сучасними програмними засобами розробки програм.

Структура робочої програми навчальної дисципліни "Об‘єктно-орієнтоване програмування" наведена в табл.1.

Таблиця 1

Структура навчальної дисципліни

Характеристика дисципліни: підготовка бакалаврів	Галузь знань, напрями підготовки освітньо-кваліфікаційний рівень	Характеристика навчальної дисципліни
Кількість кредитів, відповідних до ECTS — 8; у тому числі: змістовних модулів — 3; самостійна робота; індивідуальне науково-дослідне завдання (ІНДЗ) ; курсове проектування.	Шифр та галузі знань 0501 "Інформатика та обчислювальна техніка"	Обов'язкова Рік підготовки: 2. Семестр: 3, 4
Кількість годин: усього — 288; за змістовними модулями: модулі 1; 2 — 90 год.; модуль 3 — 198 год.	Назва напряму підготовки: "Комп'ютерні науки"	Лекції: кількість годин — 72. Лабораторні заняття: кількість годин — 72. Самостійна робота: кількість годин — 90. Індивідуальна робота: кількість годин — 54
Кількість тижнів викладання дисципліни: 36. Кількість годин за тиждень — 4	Освітньо-кваліфікаційний рівень: бакалавр	Вид контролю: іспит

У процесі навчання студенти отримують необхідні знання під час проведення аудиторних занять: лекційних та лабораторних. Велике значення в процесі вивчення та закріплення знань має самостійна робота студентів, у тому числі робота над курсовим проектом, а також виконання індивідуального навчально-дослідного завдання.

Зміст усіх видів занять розроблено відповідно до положень Болонської декларації, враховано рекомендації щодо кредитно-модульної системи організації навчального процесу.

1. Кваліфікаційні вимоги до студентів в галузі інформаційних управляючих систем і технологій

Навчальна дисципліна "Об‘єктно-орієнтоване програмування" є базовою для підготовки бакалаврів напрямку підготовки "Комп'ютерні науки".

Необхідна навчальна база перед початком вивчення дисципліни:

з метою найкращого засвоєння матеріалу студенти повинні до початку вивчення дисципліни засвоїти теоретичні знання та опанувати практичні вміння з дисципліни "Основи програмування та алгоритмічні мови", а також мати навички роботи з персональним комп’ютером.

В результаті вивчення даної дисципліни студенти повинні:

знати:

принципи об‘єктно-орієнтованого програмування;

поняття класу та об‘єкту, співвідношення між ними;

типи відношень між класами;

порядок проектування класів;

життєвий цикл об‘єктів;

реалізацію основних концепцій об‘єктно-орієнтованого програмування у мові C#;

способи розробки графічного інтерфейсу користувача за допомогою технології Windows Forms.

уміти:

використовувати основні концепції об‘єктно-орієнтованого програмування при розробці програм;

розробляти об‘єктно-орієнтовані програми на мові C#;

використовувати головні можливості бібліотеки .Net при розробці додатків та графічного інтерфейсу користувача;

виконувати основні технологічні операції з розробки програм у інтегрованому середовищі розробки Microsoft Visual Studio.

2. Тематичний план навчальної дисципліни

При вивченні дисципліни "Об‘єктно-орієнтоване програмування" студент має ознайомитися з програмою дисципліни, її структурою, формами та методами навчання, видами та методами контролю знань.

Тематичний план дисципліни "Об‘єктно-орієнтоване програмування" складається з трьох модулів, кожний з яких об'єднує у собі відносно окремий самостійний блок дисципліни, який логічно пов'язує кілька навчальних елементів дисципліни за змістом та взаємозв'язками.

Навчальний процес здійснюється у таких формах: лекційні та лабораторні заняття, індивідуальна науково-дослідна робота, курсове проектування, самостійна робота студента. Структура залікового кредиту дисципліни наведена у табл. 2.

Таблиця 2

Структура залікового кредиту навчальної дисципліни

Тема	Кількість годин			
	Лекції	Лабора-торні заняття	Індивідуальна робота	Самостійна робота
1	2	3	4	5
Модуль 1. Використання головних концепцій ООП при розробці додатків на мові C#				
Тема 1. Введення до платформи Microsoft .Net та мови C#	4	6		1
Тема 2. Реалізація головних концепцій об‘єктно-орієнтованого програмування у мові C#	10	12		2
Тема 3. Основи використання мови XML під час розробки додатків для .Net	4	–		2
Разом годин за модулем	18	18	3	5
Модуль 2. Використання основних бібліотек .Net при розробці додатків на мові C#				
Тема 4. Основні бібліотеки .Net	12	18		2
Тема 5. Особливості застосування платформи .Net при розробці програмного забезпечення	6	–		2
Разом годин за модулем	18	18	6	4
Модуль 3. Використання концепцій ООП щодо розробки додатків з графічним інтерфейсом користувача на мові C#				
Тема 6 Основи використання технології Windows Forms	6	8		15

Закінчення табл. 2

1	2	3	4	5
Тема 7 Розробка та використання елементів управління	12	10		26
Тема 8 Використання графічних можливостей технології Windows Forms	4	6		12
Тема 9 Розробка багатопотокових програм	4	6		14
Тема 10 Використання додаткових можливостей платформи .Net	10	6		14
Разом годин за модулем	36	36	45	81
Всього годин	72	72	54	90

3. Зміст навчальної дисципліни за модулями та темами

МОДУЛЬ 1. ВИКОРИСТАННЯ ГОЛОВНИХ КОНЦЕПЦІЙ ООП ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 1. Введення до платформи Microsoft .Net та мови C#

Платформа Microsoft .Net: історія розвитку, архітектура, засоби розробки додатків, компіляція та виконання програм, бібліотека базових класів, система типізації.

Загальні відомості про мову C#: особливості використання, алфавіт, типи даних, порівняння типів-значень та типів-посилань, вбудовані типи-значення, вбудовані типі-посилання, одномірні та багатомірні масиви, операції, оператори, структура програми, коментарі, особливості використання функцій, механізми передачі параметрів, простори імен, основи використання бібліотеки базових класів .Net.

Культура програмування: вимоги до інтерфейсу користувача та вихідного тексту програми.

Тема 2. Реалізація головних концепцій об‘єктно-орієнтованого програмування у мові C#

Особливості розробки програмних систем. Прості та складні програмні системи. Ознаки складних програмних систем. Декомпозиція програмних систем. Способи декомпозиції. Об‘єктно-орієнтована

декомпозиція. Переваги об‘єктно-орієнтованого підходу до розробки програмних систем.

Поняття об‘єкта. Характеристики об‘єкта. Поняття класу. Співвідношення між класом та його об‘єктом. Складові частки об‘єктно-орієнтованої технології та їх використання щодо розробки програмних систем.

Визначення терміна "об‘єктно-орієнтоване програмування". Принципи об‘єктно-орієнтованого програмування. Абстракція. Інкапсуляція. Ієрархія. Агрегація та спадкування. Поліморфізм. Достоїнства та недоліки об‘єктно-орієнтованого програмування.

Абстрактні типи даних. Проектування абстрактного типу даних. Синтаксис структур та класів у мові C#. Елементи класу. Доступ до елементів класу. Посилання this. Перевантаження методів класу.

Об‘єкти в програмі. Послідовність створення об‘єкта. Конструктори. Основні властивості конструкторів. Звільнення пам‘яті. Система "збору сміття". Статичні дані та методи: призначення, властивості, особливості використання.

Відношення агрегації. Реалізація агрегації у мові C#. Відношення спадкування. Синтаксис спадкування у мові C#. Ініціалізація об‘єкта базового класу. Варіанти використання спадкування. Перевизначення методів. Заборона спадкування. Рядкове представлення об‘єкта.

Реалізація принципу поліморфізму у мові C#. Раннє та пізнє зв‘язування. Віртуальні методи. Абстрактні класи та методи. Реалізація поліморфної поведінки на базі абстрактного класу. Правила застосування абстрактних класів. Інтерфейси. Реалізація поліморфної поведінки на базі інтерфейсу. Правила застосування інтерфейсів.

Тема 3. Основи використання мови XML під час розробки додатків для .Net

Введення до мови XML. Створення та відображення XML-документа. Структура XML-документа. Коректні XML-документи. Відображення XML-документа за допомогою каскадних таблиць стилів SAX та DOM. Використання мови XML у додатках .Net.

МОДУЛЬ 2. ВИКОРИСТАННЯ ОСНОВНИХ БІБЛІОТЕК .NET ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 4. Основні бібліотеки .Net

Принципи перевантаження операцій. Особливості використання функції operator. Індексатори. Властивості.

Види помилок у програмах. Проблеми традиційного підходу до обробки помилок. Механізм обробки виключень. Класи виключень стандартної бібліотеки .Net. Синтаксис обробки виключень. Перевірка на арифметичне переповнення.

Джерела та споживачі даних. Загальні відомості про потоки введення-виведення даних. Алгоритми роботи потоків введення-виведення даних. Основні класи стандартної бібліотеки .Net для підтримки введення-виведення даних.

Загальні відомості про контейнери. Основні елементи та структури даних стандартної бібліотеки контейнерів .Net. Типізовані контейнери.

Особливості реалізації рядкового типу даних у платформі .Net. Класи стандартної бібліотеки .Net для представлення рядків та особливості їхнього використання. Форматування рядків. Призначення та застосування регулярних виразів. Підтримка регулярних виразів у стандартній бібліотеці .Net. Спеціальні символи, яки використовуються у регулярних виразах.

Тема 5. Особливості застосування платформи .Net при розробці програмного забезпечення

Управління пам‘яттю для типів-значень та типів-посилань. Звільнення ресурсів, що "не управляються". Деструктори. "Небезпечний" програмний код. Вказівники та їх використання. Арифметика вказівників. Вказівники на структури та елементи класів. Застосування вказівників щодо оптимізації продуктивності програми.

Введення до атрибутів. Елементи програми до яких можливе застосування атрибутів. Визначені атрибути. Використання атрибутів умовної компіляції. Атрибути рівня модуля компіляції.

Збереження та відновлення стану об‘єктів у .Net. Серіалізація та десеріалізація. "Граф" об‘єктів при серіалізації. Створення класів, об‘єкти яких можливо серіалізувати. Процеси серіалізації та десеріалізації. Формати серіалізації. Серіалізація та десеріалізація об‘єктів у двійковому та XML-форматах.

МОДУЛЬ 3. ВИКОРИСТАННЯ КОНЦЕПЦІЙ ООП ЩОДО РОЗРОБКИ ДОДАТКІВ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА НА МОВІ C#

Тема 6. Основи використання технології Windows Forms

Загальні відомості про делегати. Оголошення та використання делегатів у мові C#. Анонімні методи. Групові делегати. Загальні відомості про події. Подія з погляду одержувача. Генерування подій.

"Традиційна" модель програмування на платформі .Net. Модель "Windows-програмування" на платформі .Net. Технологія Windows Forms. Форми. Загальна структура додатку з графічним інтерфейсом користувача на платформі .Net. Розробка додатків Windows Forms за допомогою інтегрованого середовища. Події рівня форми.

"Колекція" візуальних елементів управління форми. Використання базових візуальних елементів управління.

Тема 7. Розробка та використання елементів управління

Основи архітектури додатків Windows Forms. Модель подій у Windows Forms. Діалогові вікна. Основні візуальні елементи управління: властивості та використання. Компоненти форми для виключення помилкового введення даних користувачем.

Використання візуальних елементів управління "дерево" та "таблиця".

Тема 8. Використання графічних можливостей технології Windows Forms

Поняття контексту пристрою в операційній системі Windows. Особливості графічного виведення даних. Логічна система координат. Контекст пристрою у .Net. Простори імен GDI+. Обробка повідомлення перемальовування. Програмне генерування повідомлення перемальовування. Графічні об‘єкти GDI+. Використання кистей, пер та шрифтів. Робота з графічними зображеннями.

Тема 9. Розробка багатопотокових програм

Багатозадачні операційні системи. Основні поняття багатозадачності. Потоки виконання. Домени додатків у .Net. "Життєвий" цикл потоку виконання. Управління первинним потоком виконання. Синхронізація потоків виконання.

Тема 10. Використання додаткових можливостей платформи .Net

Поняття модуля компіляції у .Net. Структура модуля компіляції. Приватні та поділювані модулі компіляції. Підтримка міжмовної взаємодії у .Net. Глобальний кеш модулів компіляції. Створення поділюваних модулів компіляції.

Загальні відомості про розгортання додатків .Net. Просте розгортання. Проекти інсталяторів додатків.

Загальні відомості про локалізацію додатків .Net. Культури та регіони. Файли ресурсів. Створення та використання файлів ресурсів. Підтримка локалізації додатків у стандартній бібліотеці .Net.

Поняття про систему безпеки платформи .Net. Групи коду. Повноваження доступу коду та набори повноважень. Рівні політики безпеки. Управління політиками безпеки. Конфігураційні файли. Управління групами коду та повноваженнями.

4. Плани лекцій

МОДУЛЬ 1. ВИКОРИСТАННЯ ГОЛОВНИХ КОНЦЕПЦІЙ ООП ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 1. Введення до платформи Microsoft .Net та мови C#

1.1. Основні поняття платформи Microsoft .Net.

1.2. Основи мови C#.

Література: основна [1 – 3]; додаткова [4; 7].

Тема 2. Реалізація головних концепцій об‘єктно-орієнтованого програмування у мові C#

2.1. Основні положення об‘єктно-орієнтованого підходу.

2.2. Класи та об‘єкти, співвідношення між ними.

2.3. Створення та руйнування об‘єктів.

2.4. Повторне використання класів.

2.5. Реалізація поліморфізму в C#.

Література: основна [1 – 3]; додаткова [4; 5].

Тема 3. Основи використання мови XML під час розробки додатків для .Net

3.1. Основи використання мови XML під час розробки додатків для .Net.

Література: основна [1; 2]; додаткова [4].

МОДУЛЬ 2. ВИКОРИСТАННЯ ОСНОВНИХ БІБЛІОТЕК .NET ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 4. Основні бібліотеки .Net

4.1. Принципи перевантаження операцій.
4.2. Індексатори та властивості.
4.3. Обробка виключень.
4.4. Введення-виведення даних.
4.5. Контейнери.
4.6. Рядки та регулярні вирази.
Література: основна [1 – 3]; додаткова [4; 6; 7].

Тема 5. Особливості застосування платформи .Net при розробці програмного забезпечення

5.1. Управління пам‘яттю та вказівники.
5.2. Атрибути.
5.3. Збереження та відновлення стану об‘єктів у .Net.
Література: основна [1; 2]; додаткова [4; 6; 7].

МОДУЛЬ 3. ВИКОРИСТАННЯ КОНЦЕПЦІЙ ООП ЩОДО РОЗРОБКИ ДОДАТКІВ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА НА МОВІ C#

Тема 6. Основи використання технології Windows Forms

6.1. Делегати та події.
6.2. Введення до Windows Forms.
6.3. Основи використання елементів управління.
Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 7. Розробка та використання елементів управління

7.1. Використання основних елементів управління.
7.2. Використання "дерев" у графічному інтерфейсі користувача.
7.3. Використання таблиць у графічному інтерфейсі користувача.
7.4. Розробка MDI-додатків.
7.5. Розробка елементів управління.
Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 8. Використання графічних можливостей технології Windows Forms

8.1. Використання графічних можливостей Windows Forms.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 9. Розробка багатопотокових програм

9.1. Реалізація багатопотоковості у .Net.

9.2. Синхронізація потоків.

Література: основна [1; 2]; додаткова [4; 7].

Тема 10. Використання додаткових можливостей платформи .Net

10.1. Модулі компіляції.

10.2. Розгортання додатків.

10.3. Локалізація додатків.

10.4. Система безпеки.

Література: основна [1; 2]; додаткова [4].

5. Плани лабораторних занять

Лабораторне заняття — це організаційна форма навчального заняття, на якому студенти під керівництвом викладача формують уміння й навички з практичного застосування основних теоретичних положень навчальної дисципліни шляхом виконання завдань до лабораторних робіт.

Лабораторні заняття з дисципліни "Об‘єктно-орієнтоване програмування" проводяться в спеціально обладнаному навчальному класі з використанням комп’ютерного устаткування пристосованого до начального процесу.

З метою підвищення якості навчального процесу, під час проведення лабораторного заняття призначається ще один викладач і навчальна група ділиться на дві підгрупи. Кожний студент працює самостійно, виконуючи індивідуальне завдання для лабораторного дослідження.

За результатами виконаної на занятті лабораторної роботи студенти оформлюють індивідуальні звіти з її виконання та захищають ці звіти перед викладачем. Результати виконання лабораторних досліджень оцінюються викладачем.

Тематика проведення лабораторних занять наведена у табл. 3.

Таблиця 3

План проведення лабораторних занять

Назва теми	Назва лабораторного заняття та питання що опрацьовуються	Кількість годин	Література
1	2	3	4
МОДУЛЬ 1. Використання головних концепцій ООП при розробці додатків на мові C#			
Тема 1. Введення до платформи Microsoft .Net та мови C#	Основи використання мови C#	6	Основна [1 – 3]; додаткова [4; 7]
Тема 2. Реалізація головних концепцій об'єктно-орієнтованого програмування у мові C#	Розробка додатків з використанням базових елементів ООП	4	Основна [1 – 3]; додаткова [4; 5]
	Застосування концепції повторного використання класів при розробці додатків	8	Основна [1 – 3]; додаткова [4; 5]
МОДУЛЬ 2. Використання основних бібліотек .Net при розробці додатків на мові C#			
Тема 4. Основні бібліотеки .Net	Використання перевантаження операцій при розробці додатків	4	Основна [1 – 3]; додаткова [4; 6; 7]
	Використання основних бібліотек .Net	6	Основна [1 – 3]; додаткова [4; 6; 7]
	Використання регулярних виразів при розробці додатків	8	Основна [1 – 3]; додаткова [4; 6; 7]
Модуль 3. Використання концепцій ООП щодо розробки додатків з графічним інтерфейсом користувача на мові C#			
Тема 6. Основи використання технології Windows Forms	Використання делегатів та подій при розробці додатків	8	Основна [1 – 3]. додаткова [4; 8; 9; 10]
	Використання основних концепцій Windows Forms		

Закінчення табл. 3

1	2	3	4
Тема 7. Розробка та використання елементів управління	Використання елементів управління у додатках WinForms	10	Основна [1 – 3]; додаткова [4; 8; 9; 10]
Тема 8. Використання графічних можливостей технології Windows Forms	Використання графіки у додатках WinForms	6	Основна [1 – 3]; додаткова [4; 8; 9; 10]
Тема 9. Розробка багатопотокових програм	Розробка багатопотокових програм	6	Основна [1; 2]; додаткова [4; 7]
Тема 10. Використання додаткових можливостей платформи .Net	Розгортання та локалізація додатків .Net	6	Основна [1; 2]; додаткова [4]
Разом годин		72	

6. Самостійна робота студентів

6.1. Основні форми самостійної роботи студентів

Для опанування матеріалу дисципліни "Об‘єктно-орієнтоване програмування" окрім лекційних та лабораторних занять, тобто аудиторної роботи, значну увагу необхідно приділяти самостійній роботі.

Основні форми самостійної роботи студента:

1. Вивчення лекційного матеріалу.
2. Вивчення окремих тем або питань, що передбачені для самостійного опрацювання.
3. Вивчення основних термінів та понять за темами дисципліни.
4. Підготовка до лабораторних занять.
5. Контрольна перевірка кожним студентом знань за питаннями для самодіагностики.
6. Підготовка до проміжного та підсумкового модульного контролю.
7. Систематизація вивченого матеріалу перед іспитом.
8. Виконання індивідуального навчально-дослідного завдання.
9. Оформлення звітів з лабораторних робіт.
10. Робота з опрацювання та вивчення рекомендованої літератури.
11. Робота над курсовим проектом.

6.2. Питання для самостійного опрацювання

МОДУЛЬ 1. ВИКОРИСТАННЯ ГОЛОВНИХ КОНЦЕПЦІЙ ООП ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 1. Введення до платформи Microsoft .Net та мови C#

1. Історія та етапи розвитку платформи Microsoft.Net.
2. Переваги платформи Microsoft .Net над аналогами.
3. Призначення та особливості віртуальної машини.
4. Призначення основних просторів імен бібліотеки класів платформи Microsoft .Net.
5. Використання мови XML у програмних проектах .Net.
6. Варіанти об‘явлення та використання головного методу програми на мові C#.
7. Способи використання директиви using.
8. Форматування даних при їх виведенні.

9. Використання утілити ILDasm.
10. Налагодження програм у середовищі Visual Studio.
11. Використання засобів SDK щодо розробки програм.

Література: основна [1 – 3]; додаткова [4; 5].

Тема 2. Реалізація головних концепцій об‘єктно-орієнтованого програмування у мові C#

1. Призначення та використання перечислень.
2. Призначення та використання структур.
3. Використання списків аргументів змінної довжини.
4. Розробка рекурсивних методів.
5. Призначення та елементи класу Array.
6. Використання аргументів командного рядку.
7. Використання внутрішніх класів.
8. Пізнє та раннє зв‘язування.

Література: [1 – 4; 6; 11; 13; 15].

Тема 3. Основи використання мови XML під час розробки додатків для .Net

1. Документування вихідного коду програми.
2. Документаційні XML-теги.
3. Генерація XML-документації за допомогою середовища Visual Studio.

Література: основна [1; 2]; додаткова [4].

МОДУЛЬ 2. ВИКОРИСТАННЯ ОСНОВНИХ БІБЛІОТЕК .NET ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 4. Основні бібліотеки .Net

1. Перетворення типів, яки визначаються користувачем.
2. Призначення та використання елементів класу System.Exception.
3. Використання вкладених блоків try.
4. Розробка власних класів виключень.
5. Генерування користувальницьких виключень.
6. Перехоплення користувальницьких виключень.
7. Особливості буферизованих потоків введення-виведення даних.
8. Хеш-таблиці та хеш-функції.
9. Сортування вмісту контейнера.

10. Розроблення користувальницьких контейнерів.

11. Засоби форматування рядків.

Література: основна [1 – 3]; додаткова [4; 6; 7].

Тема 5. Особливості застосування платформи .Net при розробці програмного забезпечення

1. Використання інтерфейсу IDisposable.

2. Використання програмного стеку.

3. Динамічна область пам‘яті та особливості її використання програмою.

4. Алгоритм роботи "збирача сміття".

5. Поняття про "покоління" об‘єктів.

6. Приведення типів вказівників.

7. Розробка користувальницьких атрибутів.

8. Обмеження на застосування атрибутів.

9. Критерії вибору формату серіалізації.

10. Налагодження параметрів серіалізації за допомогою атрибутів.

Література: основна [1; 2]; додаткова [4; 6; 7].

МОДУЛЬ 3. ВИКОРИСТАННЯ КОНЦЕПЦІЙ ООП ЩОДО РОЗРОБКИ ДОДАТКІВ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА НА МОВІ C#

Тема 6. Основи використання технології Windows Forms

1. Інтерфейси зворотного виклику.

2. Коваріантність делегатів.

3. Реалізація групових викликів.

4. Реєстрація подій у середовищі Visual Studio.

5. Призначення та застосування класу Application.

6. Функціональні можливості класу Control.

7. Використання "колекції" візуальних елементів управління форми.

8. Застосування оброблювачів подій рівня форми.

9. Розміщення візуальних елементів управління на формі.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 7. Розробка та використання елементів управління

1. Ієрархія класів-предків класу Form та його функціональні можливості.

2. Основні властивості та події класу Control.

3. Склад та призначення основних файлів проекту типу Windows Application у середовищі Visual Studio.

4. Використання класу Timer.

5. Використання візуального елементу управління CheckedListBox.

6. Основні елементи класу TreeView та особливості його використання.

7. Основні елементи класу DataGridView та особливості його використання.

8. Основні елементи класу DataTable та особливості його використання.

9. Спадкування форм.

10. Основні властивості перерахування DialogResult.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 8. Використання графічних можливостей технології Windows Forms

1. Особливості застосування системи координат GDI+.

2. Допоміжні типи даних простору імен System.Drawing.

3. Перевірка влучення у задану область графічного зображення.

4. Використання штрихових кистей.

5. Використання градієнтних кистей.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 9. Розробка багатопотокових програм

1. Одержання списку процесів за допомогою стандартної бібліотеки .Net.

2. Одержання інформації про процес за допомогою стандартної бібліотеки .Net.

3. Управління процесами за допомогою стандартної бібліотеки .Net.

4. Одержання списку доменів процесу за допомогою стандартної бібліотеки .Net.

5. Програмне створення доменів.

6. Використання делегатів щодо багатопотокових програм.

7. Основні елементи класу Thread та особливості його використання.

8. Застосування атрибутів для синхронізації потоків виконання.

Література: основна [1; 2]; додаткова [4; 7].

Тема 10. Використання додаткових можливостей платформи .Net

1. Структура маніфесту модуля компіляції.
2. Використання "суворих" імен модуля компіляції.
3. Варіанти розгортання додатків .Net.
4. Створення інсталяторів за допомогою середовища Visual Studio.
5. Сортування даних в залежності від культури.
6. Програмне переключення між культурами.
7. Програмне застосування повноважень коду.

Література: основна [1; 2]; додаткова [4].

6.3. Контрольні запитання для самодіагностики

МОДУЛЬ 1. ВИКОРИСТАННЯ ГОЛОВНИХ КОНЦЕПЦІЙ ООП ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 1. Введення до платформи Microsoft .Net та мови C#

1. Охарактеризуйте структуру програми на мові C#.
2. Перелічіть вбудовані типи даних C#.
3. Призначення методів класу Math.
4. Порядок створення проекту консольного додатка C# у середовищі Visual Studio.
5. Порядок відкриття проекту у середовищі Visual Studio.
6. Порядок добавлення файлу до проекту у середовищі Visual Studio.
7. Порядок компіляції та запуску програми на виконання у середовищі Visual Studio.
8. Порядок налагодження програми у середовищі Visual Studio.
9. Порядок використання утиліти Object Browser**.**

Література: основна [1 – 3]; додаткова [4; 7].

Тема 2. Реалізація головних концепцій об‘єктно-орієнтованого програмування у мові C#

1. Пояснити визначення методу: *public static void Main(string[ ] args) { }*
2. Принципи об’єктно-орієнтованого підходу.
3. Синтаксис опису класу.
4. Особливості статичних елементів класу.

5. Специфікатори доступу до елементів класу у C#.
6. Порядок ініціалізації об'єкта класу.
7. Призначення та використання посилання this.
8. Агрегація й наслідування.
9. Синтаксис наслідування у C#.
10. Порядок виклику конструкторів при наслідуванні.
11. Перевантаження "базового" методу.
12. Перевизначення "базового" методу.
13. Принцип поліморфізму.
14. Переваги концепції поліморфізму.
15. Поняття про абстрактні класи та їх призначення.
16. Правила використання абстрактних класів.
17. Поняття про інтерфейси та їх призначення.
18. Правила використання інтерфейсів.

Література: основна [1 – 3]; додаткова [4; 5].

Тема 3. Основи використання мови XML під час розробки додатків для .Net

1. Структура XML-документа.
2. Поняття коректного XML-документа.
3. Відображення XML-документа за допомогою каскадних таблиць стилів.
4. Поняття про SAX та DOM.
5. Порядок використання мови XML у додатках .Net.

Література: основна [1; 2]; додаткова [4].

МОДУЛЬ 2. ВИКОРИСТАННЯ ОСНОВНИХ БІБЛІОТЕК .NET ПРИ РОЗРОБЦІ ДОДАТКІВ НА МОВІ C#

Тема 4. Основні бібліотеки .Net

1. Проблеми "традиційного підходу" до обробки помилок під час виконання програми.
2. Переваги обробки виключень у порівнянні з "традиційним підходом" до обробки помилок.
3. Механізм обробки виключень.
4. Який клас є базовим для всіх помилок і виключень в .Net ?
5. Особливості байтових і символьних потоків вводу-виводу.

6. Базові класи байтових потоків вводу-виводу .Net і їх основні методи.

7. Базові класи символьних потоків .Net і їх основні методи.

8. Призначення основних класів символьних потоків введення-виведення .Net.

9. Призначення основних класів байтових потоків введення-виведення .Net.

10. У чому різниця між буферізованими й небуферізованими потоками потоків введення-виведення .Net ?

11. Склад бібліотеки контейнерів .Net.

12. Основні інтерфейси бібліотеки контейнерів .Net.

13. Коротка характеристика структури даних "динамічний масив" і її реалізації в .Net.

14. Коротка характеристика структури даних "двозв'язний список" і її реалізації в .Net.

15. Поняття про хеш-таблицю й хеш-функцію.

16. Коротка характеристика структури даних "асоціативний масив" і її реалізації в .Net.

17. Ітератори і їхнє використання.

18. Засоби розбору рядків на лексеми бібліотеки .Net.

19. Особливості реалізації рядкового типу даних у платформі .Net.

20. Класи стандартної бібліотеки .Net для представлення рядків та особливості їхнього використання.

Література: основна [1 – 3]; додаткова [4; 6; 7].

Тема 5. Особливості застосування платформи .Net при розробці програмного забезпечення

1. Принципи управління пам'яттю для типів-значень та типів-посилань.

2. Як звільнити ресурс, що "не управляється"?

3. Застосування вказівників щодо оптимізації продуктивності програми.

4. Елементи програми до яких можливо застосування атрибутів.

5. Використання атрибутів умовної компіляції.

6. Створення класів, об'єкти яких можливо серіалізувати.

7. Формати серіалізації у .Net.

Література: основна [1; 2]; додаткова [4; 6; 7].

МОДУЛЬ 3. ВИКОРИСТАННЯ КОНЦЕПЦІЙ ООП ЩОДО РОЗРОБКИ ДОДАТКІВ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА НА МОВІ C#

Тема 6. Основи використання технології Windows Forms

1. Оголошення та використання делегатів у мові C#.
2. Поняття групового делегата.
3. Поняття події.
4. Загальна структура додатка з графічним інтерфейсом користувача на платформі .Net.
5. Порядок добавлення елемента управління на форму.
6. Ієрархія класів простору імен System.Windows.Forms.
7. "Колекція" візуальних елементів управління форми.
8. Використання компонента MenuStrip.
9. Використання компонента ToolStrip.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 7. Розробка та використання елементів управління

1. Обробка подій пунктів меню.
2. Обробка подій кнопок панелі інструментів.
3. Акселератори та підказки, яки спливають.
4. Властивості та використання компонента StatusStrip.
5. Властивості та використання компонента Button.
6. Властивості та використання компонента TextBox.
7. Властивості та використання компонента Label.
8. Властивості та використання компонента GroupBox.
9. Властивості та використання компонента RadioButton.
10. Властивості та використання компонента CheckBox.
11. Властивості та використання компонента TabControl.
12. Властивості та використання компонента TreeView.
13. Файлове введення-виведення. Стандартні діалоги SaveFileDialog, OpenFileDialog.
14. Використання компонента ErrorProvider.
15. Основні властивості елемента управління "дерево".
16. Основні властивості елемента управління "таблиця".

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 8. Використання графічних можливостей технології Windows Forms

1. Поняття контексту пристрою у .Net.
2. Поняття про логічну систему координат.
3. Простори імен GDI+ та їх призначення.
4. Обробка повідомлення перемальовування.
5. Програмне генерування повідомлення перемальовування.
6. Використання кистей.
7. Використання пер.
8. Використання шрифтів.
9. Робота з графічними зображеннями.

Література: основна [1 – 3]; додаткова [4; 8; 9; 10].

Тема 9. Розробка багатопотокових програм

1. Поняття процесу й потоку виконання.
2. Первинні й вторинні потоки виконання.
3. "Життєвий цикл" потоку виконання.
4. Способи створення нового потоку виконання в програмі на C#.
5. Для чого використовується синхронізація потоків виконання ?
6. Основні види об'єктів синхронізації.

Література: основна [1; 2]; додаткова [4; 7].

Тема 10. Використання додаткових можливостей платформи .Net

1. Структура модуля компіляції.
2. Поняття про приватні та поділювані модулі компіляції.
3. Призначення глобального кеша модулів компіляції.
4. Види проектів інсталяторів.
5. Поняття про культури та регіони.
6. Призначення та використання файлів ресурсів.
7. Поняття про систему безпеки платформи .Net.
8. Поняття про групи коду.
9. Рівні політики безпеки.
10. Призначення та використання конфігураційних файлів.

Література: основна [1; 2]; додаткова [4].

6.4. Індивідуальне навчально-дослідне завдання

Індивідуальне навчально-дослідне завдання (ІНДЗ) передбачає: систематизацію, закріплення, розширення теоретичних знань і практичних навичок із дисципліни та застосування їх при вирішенні конкретних виробничих ситуацій; розвиток навичок самостійної роботи з науково-технічною документацією.

ІНДЗ з дисципліни "Об‘єктно-орієнтоване програмування" видається студенту викладачем на початку вивчення дисципліни. ІНДЗ виконується студентом самостійно. Студент має надати ІНДЗ для перевірки наприкінці семестру, але не пізніше терміну проведення підсумкового модульного контролю. Оцінка за виконання ІНДЗ враховується при виставленні загальної оцінки з дисципліни.

Тематика ІНДЗ має носити проблемний характер. Студент має право самостійно обрати тему та зміст роботи з обов'язковим її узгодженням з викладачем.

У протилежному випадку тема має бути запропонована викладачем (варіанти тем наведені нижче).

У процесі виконання ІНДЗ студент має опрацювати не менше п'яти літературних джерел з посиланнями на використання певної інформації з них у тексті роботи. При цьому робота має носити творчий характер і бути спрямованою на вирішення певної проблеми чи на висловлення особистого погляду автора роботи на питання, яке розглядається в роботі.

ІНДЗ складається з: титульної сторінки, змісту, вступу, основної частини, висновків, списку використаної літератури, додатків до індивідуального завдання (при необхідності).

Вступ має розкривати актуальність обраної студентом теми, її проблематику, мету написання роботи.

Основна частина роботи (може включати декілька підрозділів) має включати характеристику сучасного стану проблеми, погляд різних авторів на цю проблему, позитивні та негативні наслідки проблеми.

Висновки мають включати обґрунтовані відомості студента щодо досягнення мети роботи.

Обсяг ІНДЗ повинен становити у друкованому варіанті 10 – 15 сторінок. Орієнтовна кількість сторінок у розділах: вступ — 1с.; основна частина — 8 – 12 с.; висновки — 1 – 2 с.

6.4.1. Тематика ІНДЗ

1. ООП: історія виникнення, основоположники, необхідність.
2. Основні принципи ООП.
3. Класи і об‘єкти, співвідношення між ними.
4. Створення об‘єктів, конструктори.
5. Руйнування об‘єктів, "збірання сміття".
6. Перевантаження операцій.
7. Спадкування.
8. Агрегація.
9. Поліморфізм.
10. Статичні дані.
11. Перевантаження методів.
12. Статичні методи.
13. Перевизначення методів.
14. Використання посилання this.
15. Абстрактні класи.
16. Інтерфейси.
17. Інкапсуляція.
18. Використання конструкторів при спадкуванні.
19. Управління доступом до елементів класу.
20. Клас System.Object бібліотеки .Net: призначення, основні елементи, використання.
21. Класи та структури.

6.5. Курсове проектування

6.5.1. Завдання курсового проектування

Курсове проектування є завершальним етапом вивчення 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.

6.5.2.Організація курсового проектування

Відповідно до навчального плану вивчення дисципліни "Об‘єктно-орієнтоване програмування" включає лекційні та лабораторні заняття. Завершується вивчення дисципліни написанням і захистом курсового проекту. Студенти виконують курсовий проект у 4 семестрі.

Керівництво курсовим проектуванням здійснюється викладачами кафедри інформаційних систем, яки приймають участь у викладанні цієї дисципліни.

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

Студентові надається право вільного вибору теми проекту з урахуванням його схильностей і можливостей найбільш повно застосувати отримані знання.

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

У завданні приводиться тема курсового проекту, вихідні дані до проекту, зміст пояснювальної записки, завдання на розробку додатку, строки початку й закінчення роботи над курсовим проектом, обумовлені графіком навчального процесу.

Студент розробляє зміст курсового проекту, обговорює його з керівником, підготовляє вхідні дані і приступає до проектування. У процесі проектування студент повинен регулярно відвідувати консультації керівника, подавати на перевірку йому робочі матеріали.

Курсовий проект студент повинен виконувати самостійно. Оформлений відповідно до пред'явлених вимог проект студент здає на перевірку керівникові за тиждень до строку захисту.

Захист курсових проектів організовується кафедрою інформаційних систем у комісіях за тиждень до екзаменаційної сесії за графіком, затвердженим завідувачем кафедрою.

Здійснюється захист із демонстрацією вирішення завдання на ПК на контрольному прикладі та презентацією розроблених проектних рішень. Тільки після захисту курсового проекту студент допускається до здачі екзамену з дисципліни "Об‘єктно-орієнтоване програмування".

6.5.3.Структура, зміст і обсяг курсового проекту

Курсовий проект складається з пояснювальної записки та графічного матеріалу, підготовленого у вигляді презентації, яка демонструється при захисті проекту. Обсяг пояснювальної записки становить близько 40 сторінок надрукованого на ПК тексту. Таблиці, діаграми, відеограми, машинограми, вихідні документи можна винести в додаток.

Рекомендується така структура пояснювальної записки:

Титульний аркуш.

Завдання на курсове проектування.

Зміст.

Вступ (≈2 – 3 стор.).

Перший розділ. Містить коротку характеристику призначення розробки, постановку завдання, вимоги до програми та програмної документації, етапи розробки та порядок тестування додатка (15 стор.).

Другий розділ. Містить розроблені студентом проектні рішення, структуру розробленого додатка, відомості про призначення, умови виконання, настройку та перевірку програми, повідомлення користувачу, що можуть з‘явитися у процесі роботи програми (20 стор.).

Висновки (1 стор.).

Використана література.

Додатки.

6.5.4. Методичні рекомендації щодо оформлення проекту

Важливе значення при роботі над курсовим проектом має його оформлення, до якого пред’являються певні вимоги. Весь матеріал курсового проекту треба розташувати в певній послідовності.

Титульний аркуш оформляється за встановленою формою.

У змісті приводяться заголовки розділів, підрозділів із зазначенням сторінок, з яких вони починаються. При цьому заголовки повинні бути наведені в суворій відповідності з текстом.

Текстовий матеріал курсового проекту друкується на ПК на папері формату А4 (210х297 мм.). Текст повинен відповідати правилам граматики й стилістики.

При написанні текстового матеріалу сторінки повинні бути відформатовані наступним чином: ліве поле — 30 мм., праве — 10 мм., верхнє та нижнє — 20 мм.

Абзац повинен починатися з відстані 25 мм від лівого краю сторінки.

Не дозволяється розміщати заголовки й підзаголовки в нижній частині сторінки, якщо на ній не більше 4 рядків наступного тексту.

Кожний розділ курсового проекту повинен починатися з нової сторінки, назви підрозділів, параграфів, пунктів — з абзацу.

Підкреслення найменувань розділів, підрозділів, параграфів не допускається. Відстань між заголовками розділів, підрозділів, параграфів і наступним текстом повинна бути на 5 мм більше відстані між рядками тексту. Заголовки структурних елементів проекту "ЗМІСТ", "ВСТУП", "ВИСНОВКИ", "ВИКОРИСТАНА ЛІТЕРАТУРА" і заголовки розділів треба

писати великими друкованими літерами без крапки наприкінці. При друку назви розділів центруються.

Заголовки підрозділів, параграфів і пунктів треба починати з великої букви також без крапки наприкінці. Переноси в середині слова в заголовках не допускаються.

Розділи, підрозділи, параграфи, пункти проекту треба нумерувати арабськими цифрами. Розділи мають порядкову нумерацію, наприклад: 1., 2., 3., і т. д. 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., і т. д. Номер параграфа включає номер розділу, підрозділу, порядковий номер параграфа, і розділяються вони крапкою, наприклад: 1.1.1., 1.1.2., 1.1.3., і т. д.

Сторінки курсового проекту 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, у завданні на курсовий проект і змісті 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, позначаються словом "Рис.", їм привласнюється порядковий номер (у межах номера розділу). Рисунки потрібно виконувати на одній сторінці й розташовувати відразу після згадування в тексті.

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 проекту. Над правим верхнім кутом таблиці міститься напис "Таблиця" із вказівкою її порядкового номера. Таблиця повинна мати найменування, яке розташовується на наступному рядку після слова "Таблиця".

Перерахування, при необхідності, можуть бути наведені усередині пунктів, їх варто нумерувати порядковою нумерацією арабськими

цифрами з дужкою й писати малими літерами з абзацу.

Формули необхідно виділяти з тексту в окремий рядок, залишаючи нижче й вище формули один вільний рядок. Формули треба нумерувати порядковою нумерацією в межах розділу проекту арабськими цифрами в круглих дужках у крайньому правому положенні на рядку. Пояснення значень символів числових коефіцієнтів формули потрібно приводити безпосередньо під формулою в тій же послідовності, у якій вони подані.

Значення кожного символу й числового коефіцієнта необхідно давати з нового рядка. Перший рядок пояснення починати зі словами "де" без двокрапки.

Використана в процесі роботи над курсовим проектом спеціальна література вказується наприкінці проекту перед додатком. Літературу необхідно приводити за абеткою.

Кожна література відбивається в списку в наступному порядку: порядковий номер, у списку прізвище й ініціали автора, назва книги (для статті — її заголовок, назва збірника, журналу, його номер) видавництво, місце й рік випуску.

У тексті пояснювальної записки повинні бути посилання на літературу. При цьому наводиться її порядковий номер, записаний у квадратні дужки.

Додаток потрібно оформляти як продовження проекту. Кожен додаток повинен починатися з нової сторінки й мати змістовний заголовок, написаний великими друкованими літерами. У правому верхньому куті над заголовком повинно бути написане: "Додаток". Додатки позначаються послідовно буквами: А, Б, В, Г, Д, З, К, Л, М, за винятком букв Г, Є, І, Ї, Й, О, Ч, Ь українського алфавіту.

7. Індивідуально-консультативна робота

Індивідуально-консультативна робота здійснюється за графіком індивідуально-консультативної роботи у формі індивідуальних занять, консультацій, перевірки виконання індивідуальних завдань, перевірки та захисту завдань, що винесені на поточний контроль тощо.

З теоретичної частини дисципліни індивідуально-консультативна робота проводиться у вигляді:

1) індивідуальних консультацій, на яких студент отримує відповідь від викладача на конкретні запитання або пояснення певних теоретичних положень чи аспектів їх практичного застосування;

2) групових консультацій, на яких викладач розглядує типові приклади з використання концепцій об‘єктно-орієнтованого програмування.

З практичної частини дисципліни індивідуально-консультативна робота проводиться у вигляді:

1) індивідуальних консультацій, на яких викладач розглядає лабораторні завдання, стосовно яких виникли запитання у студента;

2) групових консультацій, на яких викладач розглядає практичні ситуації, які потребують колективного обговорення.

Індивідуально-консультативна робота для комплексної оцінки засвоєння програмного матеріалу проводиться у вигляді:

1) індивідуального захисту самостійних та індивідуальних завдань;

2) підготовки рефератів для виступу на науковому семінарі;

3) підготовки рефератів для виступу на науковій конференції.

8. Методики активізації процесу навчання

При викладанні навчальної дисципліни "Об‘єктно-орієнтоване програмування" для активізації навчально-пізнавальної діяльності студентів передбачено застосування таких навчальних технологій, як: проблемні лекції, робота в малих групах, семінари-дискусії, презентації (табл. 4).

Таблиця 4

Використання навчальних технологій для активізації процесу навчання

Методики активізації процесу навчання	Практичне застосування навчальних технологій
1	2
Проблемні лекції направлено на розвиток логічного мислення студентів, коло питань теми обмежується двома-трьома ключовими моментами, використовується досвід закордонних навчальних закладів з роздачею студентам під час лекцій друкованого матеріалу та виділенням головних висновків з питань, що розглядаються. При читанні лекцій студентам даються питання для самостійного розміркування, на які лектор відповідає сам, не чекаючи відповідей студентів	Проблемна лекція з питання: "Основні концептуальні засади об‘єктно-орієнтованого програмування" (за темою 2). Проблемна лекція з питань: Реалізація принципу поліморфізму в у мові програмування C# (за темою 2). Проблемна лекція з питань: "Технологія Windows Forms та її застосування щодо розробки сучасних додатків" (за темою 6)

Закінчення табл. 4

1	2
Робота в малих групах дає змогу структурувати лабораторні заняття за формою і змістом, створює можливості для участі кожного студента в роботі за темою заняття, забезпечує формування особистісних якостей та досвіду соціального спілкування	Робота в малих групах при розробленні комплексного додатку з використанням основних бібліотек .Net. (Лабораторна робота за модулем 2)
Семінари-діскусії передбачають обмін думками і поглядами учасників з приводу даної теми, а також розвивають мислення, допомагають формувати погляди і переконання, виробляють вміння формулювати думки й висловлювати їх, вчать оцінювати пропозиції інших людей, критично підходити до власних поглядів	Проблемне повідомлення та дискусія при роботі студентів з питання: "Розробка багатопотокових програм та синхронізація потоків виконання." (Лабораторна робота за модулем 2)
Презентації — виступи перед аудиторією, що використовуються для представлення певних досягнень, результатів роботи групи	Презентація розробленої постановки задачі для розв’язання на ПК, архітектури додатка, демонстрація роботи розробленого додатка на даних контрольного прикладу (Лабораторна робота за модулем 3)

9. Система поточного та підсумкового контролю знань студентів

Система оцінювання знань, вмінь та навичок студентів передбачає виставлення оцінок за усіма формами проведення занять.

Перевірка та оцінювання знань студентів може проводитись в наступних формах:

1. Оцінювання роботи студентів в процесі лабораторних занять.
2. Оцінювання виконання індивідуального завдання.
3. Проведення проміжного контролю.
4. Проведення модульного контролю.
5. Розроблення курсового проекту.
6. Проведення підсумкового іспиту.

Загальна модульна оцінка складається з поточної оцінки, яку студент отримує під час лабораторних занять, оцінки за виконання

індивідуального завдання та оцінки за виконання модульної контрольної роботи.

Загальна оцінка з дисципліни визначається як середнє арифметичне модульних оцінок та оцінки за результатами підсумкового іспиту.

Порядок поточного оцінювання знань студентів

Поточне оцінювання здійснюється під час проведення лабораторних занять і має на меті перевірку рівня підготовленості студента до виконання конкретної роботи. Об'єктами поточного контролю є:

1) активність та результативність роботи студента протягом семестру над вивченням програмного матеріалу дисципліни, відвідування занять;

2) виконання індивідуального навчально-дослідного завдання;

3) виконання проміжного контролю;

4) виконання модульного контрольного завдання.

Контроль систематичного виконання самостійної роботи та активності на лабораторних заняттях

Оцінювання проводиться за 12-бальною шкалою за такими критеріями:

1) розуміння, ступінь засвоєння теорії та методології проблем, що розглядаються;

2) ступінь засвоєння матеріалу дисципліни;

3) ознайомлення з рекомендованою літературою, а також із сучасною літературою з питань, що розглядаються;

4) уміння поєднувати теорію з практикою при розробці програм, розв'язанні задач, проведенні розрахунків при виконанні завдань, винесених для самостійного опрацювання, та завдань, винесених на розгляд в аудиторії;

5) логіка, структура, стиль викладу матеріалу в письмових роботах і при виступах в аудиторії, вміння обґрунтовувати свою позицію, здійснювати узагальнення інформації та робити висновки.

Оцінка "відмінно" (10 – 12 балів) ставиться за умови відповідності звіту з виконаного лабораторного завдання студента та його усної відповіді при захисті завдання усім п’ятьом зазначеним критеріям. Відсутність тієї або іншої складової знижує оцінку на відповідну кількість балів.

При оцінюванні лабораторних завдань увага також приділяється якості, самостійності та своєчасності здачі виконаних завдань викладачу (згідно з графіком навчального процесу). Якщо якась із вимог не буде виконана, то оцінка на розсуд викладача буде знижена.

Проміжний модульний контроль

Проміжний модульний контроль рівня знань передбачає виявлення опанування студентом матеріалу лекційного модуля та вміння застосовувати його для вирішення практичної ситуації і проводиться у вигляді тестування. При цьому тестове завдання може містити як запитання, що стосуються теоретичного матеріалу, так і запитання, спрямовані на вирішення невеличкого практичного завдання.

Тестове завдання містить запитання одиничного і множинного вибору різного рівня складності. Для оцінювання рівня відповідей студентів на тестові завдання використовується критерій співвідношення кількості правильних відповідей та загальної кількості тестових запитань:

оцінка "відмінно" (12 балів) — 100 – 95% правильних відповідей;

оцінка "відмінно" (11 балів) — 94,99 – 90% правильних відповідей;

оцінка "відмінно" (10 балів) — 89,99 – 85% правильних відповідей;

оцінка "добре" (9 балів) — 84,99 – 80% правильних відповідей;

оцінка "добре" (8 балів) — 79,99 – 75% правильних відповідей;

оцінка "добре" (7 балів) — 74,99 – 70% правильних відповідей;

оцінка "задовільно" (6 балів) — 69,99 – 60% правильних відповідей;

оцінка "задовільно" (5 балів) — 59,99 – 50% правильних відповідей;

оцінка "задовільно" (4 бала) — 49,99 – 40% правильних відповідей;

оцінка "незадовільно" (3 бали) — 39,99 – 35% правильних відповідей;

оцінка "незадовільно" (2 – 1 бали) — менш як 35% правильних відповідей.

Тести для проміжного контролю обираються із загального переліку тестів за відповідними модулями.

Критерії оцінювання індивідуального навчально-дослідного завдання

Виконання ІНДЗ є додатковою частиною самостійної роботи студента над навчальною дисципліною "Об‘єктно-орієнтоване програмування". Мета ІНДЗ — поглиблення теоретичних знань, які були набуті студентами в процесі вивчення дисципліни та придбання

практичних вмінь та навичок з використання концепцій об‘єктно-орієнтованого програмування під час розроблення програм.

ІНДЗ має бути виконано і подано на кафедру не пізніше зазначеної в навчальному плані дати.

ІНДЗ оцінюється за критеріями:

самостійності виконання;

логічності та деталізації плану;

повноти й глибини розкриття теми;

наявності ілюстрацій (таблиці, рисунки, схеми тощо);

використання наукової інформації, відображення практичного досвіду;

якості оформлення.

Проведення поточно-модульного контролю

Поточно-модульний контроль здійснюється та оцінюється за двома складовими: практичний модульний контроль і лекційний (теоретичний) модульний контроль. Оцінка за практичну складову модульного контролю виставляється за результатами оцінювання знань студента під час лабораторних занять та виконання ІНДЗ. Лекційний модульний контроль здійснюється у формі тестів.

Для підведення підсумків роботи студентів із змістовного модуля виставляється підсумкова оцінка з поточно-модульного контролю, яка враховує оцінки за практичний модульний контроль і лекційний модульний контроль.

Таким чином, після вивчення тем 1 – 5 (модуль 1) студенти денної форми навчання виконують завдання до модуля 1; відповідно, після вивчення тем 6 – 7 (модуль 2) — завдання до модуля 2; після вивчення тем 8 – 12 (модуль 3) — завдання до модуля 3.

Приклад тестових завдань за модулем 1

1. Є фрагмент програми:

```
class A {
  public int x;
  float y;
……..
}
```

Який з модифікаторів доступу встановлений для поля y ?

a. *private*
b. *public*
c. *internal*
d. *protected*
e. *protected internal*

2. Яким є результат виконання цього фрагменту коду?

```
string myStr = "test";
switch (myStr)
{
case "hi" : Console.WriteLine("Hi!"); break;
case "hello" : Console.WriteLine("Hello!"); break;
case "test" : Console.WriteLine("Testing!"); break;
default : Console.WriteLine("Default"); break;
}
```

a. ***Testing!***
b. *Під час виконання буде генеровано виключення.*
c. *Виводу даних не буде.*
d. ***Testing!***
Default!
e. *Код не буде компілюватися.*

3. Спадкоємцем якого класу є клас Insect?

```
class Insect
{ … }
```

a. *Класу object.*
b. *Класу base.*
c. *Класу System.*
d. Ніякого.
e. Класу Insects.

4. Є послідовність типів.Net: *Single, Int16, Int32, Int64.*

Яка з послідовностей типів мови C# відповідає даній послідовності типів.Net ?

a. **byte, int, uint, float**
b. **byte, int, double, long**
c. **float, short, int, long**
d. **long, int, byte, double**
e. **short, byte, int, float**

Приклад тестових завдань за модулем 2

1. Який клас є базовим для всіх виключень .Net ?

a. **System.Exception**
b. **System.SystemException**
c. **System.RuntimeException**
d. **System.Error**
e. **System.ApplicationException**

2. Блоки catch

a. Завжди повинні йти за блоком try.
b. Одержують управління при появі виключення та виконуються за порядком їх наведення.
c. Тільки один із блоків, які ідуть за try-блоком, може одержати управління при появі виключень.
d. Завжди генерують виключення.
e. Після коректування причини, яка викликала виключення, повертають управління до точки його появлення.

3. Який з класів .Net є потоком виведення даних?

a. TextReader
b. TextWriter
c. WriterText
d. Writer
e. Reader

4. Які методи визначені у класі System.Math ?

a. Sin
b. Tg
c. Ln
d. MaxValue
e. Min

Приклад тестових завдань за модулем 3

1. Яке з подій не підтримується візуальними елементами управління Windows Forms?

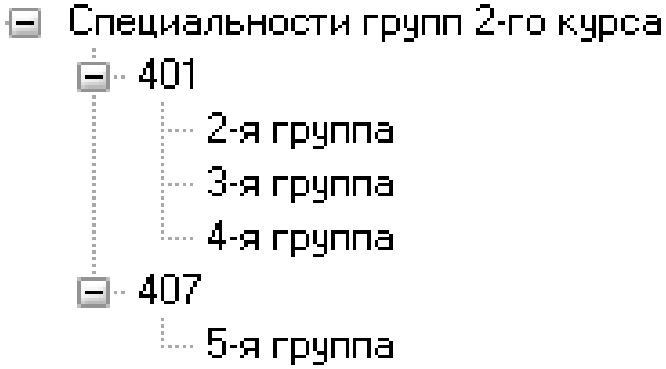
a. MouseDown
b. MouseEnter
c. MouseHover
d. MouseLeave
e. Усі події підтримуються

2. Оберіть правильне визначення обробника події:

```
Button temp = new Button();
temp.Click +=new EventHandler(BtnHandler);
```

a. **public void BtnHandler() { }**
b. **public void BtnHandler(Button sender) { }**
c. **public void BtnHandler(object sender) { }**
d. **public void BtnHandler(Button sender, EventArgs e) { }**
e. **public void BtnHandler(object sender, EventArgs e) { }**

3. Напишіть вихідний код на C# для "конструювання" елементу управління "дерево", який наведений на рисунку:



4. Яке з просторів імен не існує?

a. System.Drawing
b. System.Drawing.Drawing2D
c. System.Drawing.Drawing3D
d. System.Drawing.Text
e. System.Drawing.Printing

Проведення підсумкового іспиту

Іспит здійснюється на комп‘ютерах за екзаменаційними білетами. Екзаменаційний білет складається з двох завдань. Результаті іспиту оцінюються за 12-бальною системою. Підсумкова оцінка за іспит є сумою оцінок за кожне завдання.

Приклад екзаменаційного білета з дисципліни

Міністерство освіти і науки України
Харківський національний економічний університет

Напрям підготовки "Комп’ютерні науки".
Дисципліна: "Об‘єктно-орієнтоване програмування" (2 курс).

ЕКЗАМЕНАЦІЙНИЙ БІЛЕТ №1

Завдання 1.

Є діаграма класів: геометрична фігура (Shape) — абстрактний базовий клас, прямокутник (Rectangle) та прямокутний трикутник (Triangle) — похідні класи.

Клас Shape має абстрактний метод GetArea(), який обчислює та повертає значення площі геометричної фігури. Клас Rectangle має поле A — довжина прямокутника та B — ширина прямокутника. Клас Triangle має поля C, D — катети прямокутного трикутника. Кожний з похідних класів перевизначає метод GetArea() базового класу.

Розробити програму, яка використовує принцип поліморфізму при обчисленні площ прямокутника та прямокутного трикутника. Для цього програма повинна:

1. Створити динамічний масив (об‘єкт класу System.Collections.ArrayList).

2. Створити через посилання на тип Shape по одному об‘єкту класів Rectangle та Triangle (значення довжини та ширини прямокутника, катетів прямокутного трикутника ввести з консолі) та додати їх до динамічного масиву.

3. За допомогою оператора циклу foreach для кожного елемента динамічного масиву через посилання на тип Shape викликати метод GetArea().

4. Вивести значення площ прямокутника та прямокутного трикутника на консоль.

Завдання 2.

Задано матрицю цілих чисел, яка складається з 4 рядків та 4 стовпчиків. Розробити програму з графічним інтерфейсом користувача для знаходження добутку та кількості додатних (>0) елементів матриці.

Программа повинна:

1. Для введення елементів матриці використовувати необхідну кількість компонентів TextBox.

2. Дозволяти користувачу вводити тількі цілочисельні значення елементів матриці в інтервали [-100, 100], інакше повідомляти його про відповідну помилку (компонент ErrorProvider).

3. Виводити вихідні дані (елементи матриці) та результати роботи програми в багаторядковий компонент TextBox.

4. Для організації взаємодії з користувачем використовувати панель меню (компонент MenuStrip) з меню "Файл" (пункти: "Вийти з програми") та меню "Матриця" (пункти: "Обчислити", "Очистити матрицю").

Розроблені до кожного завдання проекти (кожний в своїй папці з іменами "Task 1", "Task 2") зберегти разом в окремій папці на диску.

Затверджено на засіданні кафедри "Інформаційних систем"
протокол №4 від 07.12.2007 р.

Зав. кафедрою_____проф. В. С. Пономаренко

Екзаменатор _____доц. Ю. Е. Парфьонов

Критерії оцінювання екзаменаційної роботи

Для оцінки рівня виконання студентами завдань використовуються відповідні критерії.

Кожне завдання оцінюється від 0 до 6 балів відповідно до наступної шкали (табл. 5).

Таблиця 5

Шкала оцінювання рівня виконання завдань

6 балів	Завдання виконано в повному обсязі. Програма працює правильно на всіх тестах. Інтерфейс та вихідний код програми задовольняють встановленим вимогам.
5 балів	Завдання виконано в повному обсязі. Програма працює правильно. Є невеликі зауваження до організації інтерфейсу користувача та (або) вихідного коду програми.
4 бала	Завдання в основному виконано. Програма працює правильно, але одна з її функціональних можливостей реалізована з порушенням вимог, вказаних в завданні.
3 бала	Завдання виконано, але не в повному обсязі. Програма працює, але не реалізована одна з функціональних вимог, вказаних в завданні, або дві з функціональних можливостей програми реалізовані з порушенням вимог, вказаних в завданні.
2 бала	Завдання не виконано. Програма запускається, але не реалізовані дві з функціональних вимог, вказаних в завданні.
1 бал	Програма не запускається, але є програмний код, розроблений студентом, який відповідає постановці завдання; або програма запускається, але не реалізовані більше двох з функціональних вимог, вказаних в завданні.
0 балів	Програма відсутня або не відповідає постановці завдання.

Підсумкова оцінка з дисципліни згідно з методикою переведення показників успішності знань студентів Університету в систему оцінювання за шкалою ECTS конвертується в підсумкову оцінку за шкалою ECTS (табл. 6).

Таблиця 6

Переведення показників успішності знань студентів у систему оцінювання за шкалою ECTS

Відсоток студентів, які зазвичай успішно досягають успішної оцінки	Оцінка за шкалою ECTS		Оцінка за бальною шкалою, що використовується в ХНЕУ	Оцінка за національною шкалою
10	відмінне виконання	A	12 – 11	відмінно
25	вище середнього рівня	B	10	
30	взагалі робота правильна, але з певною кількістю помилок	C	9 – 7	добре
25	непогано, але із значною кількістю помилок	D	6	задовільно
10	виконання задовольняє мінімальні критерії	E	5 – 4	
–	потрібне повторне перескладання	FX	3	незадовільно
–	повторне вивчення дисципліни	F	2 – 1	

10. Рекомендована література

Основна

1. Троелсен Э. C# и платформа.Net / Пер. с англ. — СПб.: Питер, 2007 — 796 с.

2. Троелсен Э. Язык программирования C# 2005 и платформа .Net 2.0 / Пер. с англ. — М.: Издательский дом "Вильямс", 2007 — 1168 с.

3. Шилдт Г. C#: учебный курс. — СПб.: Питер, 2003 — 512 с.

Додаткова

4. Нейгел К., Ивьен Б., Глинн Дж. и др. C# 2005 для профессионалов / Пер. с англ. — М.: Издательский дом "Вильямс", 2006 — 1376 с.

5. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Пер. с англ. — М.: Издательский дом "Вильямс", 2005 — 776 с.

6. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер-класс / Пер. с англ. — СПб. : Питер, 2007 — 656 с.

7. Либерти Джесс. Программирование на C#./Пер. с англ. — СПб.: Символ-плюс, 2005 — 684 с.

8. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1 / Пер. с англ. — М.: Издательско-торговый дом Русская Редакция, 2002. — 576 с.

9. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2 / Пер. с англ. — М.: Издательско-торговый дом Русская Редакция, 2002. — 624 с.

10. Лабор В. В. Си Шарп: Создание приложений для Windows. — Мн.: Харвест, 2003. — 384 с.

Ресурси мережі Internet

11. www.microsoft.com — сайт компанії Microsoft.

12. www.intuit.ru — Internet-інститут інформаційних технологій.

13. www.c-sharpcorner.com — матеріали з C#.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Робоча програма навчальної дисципліни

"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

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

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

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

Редактор **Гергеша А. В.**

Коректор **Мартовицька–Максимова В. А.**

План 2008 р. Поз. №212.

Підп. до друку Формат 60 × 90 1/16. Папір MultiCopy. Друк Riso.

Ум.-друк. арк. 3,0. Обл.-вид. арк. 3,45. Тираж прим. Зам. №

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

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

Робоча програма
навчальної дисципліни

"ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

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