МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
„ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи з курсу

«Об’єктно-орієнтоване програмування»

для студентів спеціальності 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка усіх форм навчання

Харків

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
„ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи з курсу

«Об’єктно-орієнтоване програмування»

для студентів спеціальності 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка усіх форм навчання

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 15.02.2024

Харків
2024

Методичні вказівки до самостійної роботи з курсу «Об’єктно-орієнтоване програмування» 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка усіх форм навчання /уклад. О. В. Пугановський, – Харків: НТУ “ХПІ”, 2024. – 18 с .

Укладач: О. В. Пугановський

Рецензент І.Л. Красніков

Кафедра автоматизації технологічних систем та екологічного моніторингу

ВСТУП

Курс «Об'єктно-орієнтоване програмування» базується на попередньо вивчених основах програмування мовою C# або Java. При самостійному вивченні дисципліни основну увагу потрібно приділити переліку питань, що вивчаються. Теоретичні матеріали до курсу можуть бути отримані із джерел з вільним доступом або з лекційних матеріалів до курсу, доступ до яких надається студентам. Весь курс скомпільовано з матеріалів, що є у вільному доступі. Таким чином, для розширення знань чи для більш детального вивчення окремих питань, є можливість звернутись до Інтернет-ресурсів.

При освоєнні матеріалів курсу необхідно мати доступ до середовища програмування C# або Java. Вони можуть бути встановлені на локальному обладнанні або можна використати онлайн платформи через інтернет оглядач.

Самостійна підготовка передбачає вивчення теоретичного матеріалу, виконання лабораторних робіт та індивідуальне розрахункове завдання.

У даних методичних вказівках наведено вимоги до виконання індивідуального завдання.

ТЕМА 1. ПРЕДМЕТ ТА ЗАДАЧІ КУРСУ. ОСНОВНІ СТРУКТУРНІ ЕЛЕМЕНТИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Загальні відомості.

Курс призначено для вивчення принципів об'єктно-орієнтованого програмування (ООП). Для цього необхідно мати знання про структурні елементи обраної мови програмування. Курс побудовано на мові програмування C# і від студентів вимагається знання синтаксису та вміння використовувати основні оператори, команди та інструкції мови.

Вивчення курсу починається з розгляду питання про використання систем штучного інтелекту (ШІ) для створення програм і додатків.

https://chat.openai.com/

https://platform.openai.com

Стрімкий розвиток даної технології породжує велику кількість спекуляцій і потребує від майбутніх фахівців чіткого знання особливостей застосування даної технології.

Увага !!! Системи ШІ для широкого застосування, наприклад ChatGPT, можуть видавати доволі пристойні фрагменти коду початкового рівня. Такий код у більшості випадків буде коректно сприйнятий середовищем програмування але коректність його функціонування потребує додаткового аналізу. Вирішення завдань програмування більш високого рівня, потребує спеціалізованих систем ШІ.

Для вивчення теми об'єктно-орієнтованого програмування, потрібно звернути увагу на вивчення таких структурних елементів як:

- методи;
- делегати;
- події;
- класи;

На відміну від лінійного програмування, де інструкції виконуються одна за одною від початку алгоритму і до його кінця, об'єктно-орієнтоване програ-

мування передбачає використання окремих функціональних блоків, що взаємно пов'язані але можуть існувати не залежно від інших. Більш того, функції цих блоків можуть залежити від поточних обчислювальних процесів. Такий принцип дозволяє створювати програми з мінімальними витратами пам'яті системи та уніфікованими блоками, що можуть бути модифіковані під час роботи програми чи при об'єднанні створеного коду з новим.

Більшість сучасних мов належить до мов з підтрикою ООП. Під поняття «об'єкт» підпадає більшість елементів мови програмування, хоча в літературних джерелах, традиційним елементом ООП є класи. Наприклад, типи даних, що закладено в мови програмування, також є об'єктами. Або елементи форм у візуальному інтерфейсі.

Увага!!! Таким чином, об'єкт представляє деякий шаблон, що описує його властивості. З чого витікає важливий принцип:

«Для використання об'єкту потрібно зробити копію з шаблону» - екземпляр.

Важливими принципами ООП є :

- абстракція;
- асоціація;
- композиція;
- агрегація;
- наслідування;
- інкапсуляція;
- поліморфізм.

Абстракція – виділення окремих елементів роз'язання задачі у самостійні блоки.

Асоціація – принцип, згідно якому при описі об'єкту можна використовувати інші об'єкти. Останні, можуть також включати інші об'єкти і так далі. Тобто, утворюється «дерево» з використовуваних об'єктів.

Композиція – принцип, при якому дочірній об’єкт існує, доки існує батьківський. Зменшує ресурс системи на прибирання «сміття» - не потрібних даних чи фрагментів коду.

Агрегація – принцип, коли дочірні об’єкти існують не залежно від батьківського і можуть бути використані іншими об’єктами. Полегшує розробку програм і зменшує об’єм програмного коду за рахунок багатократного використання одного і того об’єкту.

Наслідування – принцип використання властивостей об’єкту при створенні його екземплярів, чи створенні нових об’єктів, що включають властивості базового.

Інкапсуляція – дозволяє визначити, які властивості об’єктів можна використовувати (їм надають ознаку **public**) а до яких доступ ззовні заборонено (їм надають ознаки **private** чи **protected**).

Поліморфізм надає можливість автоматично визначати спосіб поведінки об’єктів при виконанні програми.

Статті, що пояснюють вище описане:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/127-urok-12-poniattia-ob-iektno-oriientovanoho-prohramuvannia-oop-klasy-i-ob-iekty

Сучасні мови програмування. Платформа NET.

Наразі існує велика кількість мов програмування, більшість з яких орієнтована на певний сегмент задач. За останні десятиріччя, стрімкий розвиток технологій Інтернет, стимулював поширення мови Java та Java Script. А у останні роки намітився перехід на мову Python. Ці мови орієнтовані на не візуальне використання але останнім часом цей недолік компенсується за рахунок сторонніх бібліотек, що є у вільному доступі. Однією з мов програмування, що має універсальне застосування є мова C#. Це потужний інструмент, призначений для розробки, як серверних так і клієнтських додатків. Програмний код може включати елементи деяких інших мов та може бути портований на різні платформи.

Кросплатформовий принцип досягається за рахунок проміжної мови Microsoft Intermediate Language (MSIL), яка була переіменована при створенні стандарту ECMA-335. Наразі її називають Common Intermediate Language (CIL), що входить до складу CLR (common language runtime). Ця мова є основою платформи NET Framework. Тобто, усі мови програмування, що орієнтовані на технологію NET перетворюють код програми у інструкції CIL а далі операційна система перетворює їх у інструкції для виконання. Цим досягається незалежність від операційної системи. Необхідною умовою є встановлення у системі NET компонентів.

Порівняння мов можна подивитись, наприклад:

https://foxminded.ua/vidminnosti-mov-prohramuvannia/

https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/download/10490/8923

Вимоги до розробки програмних продуктів.

При розробленні програмних продуктів, висувать ряд вимог на основі яких можна стверджувати про якість створеного програмного засобу.

https://ela.kpi.ua/bitstream/123456789/38101/1/Rozrobka_ta_analiz_KP.pdf

https://e-tk.lntu.edu.ua/mod/resource/view.php?id=8419

ТЕМА 2. ПРИНЦИПИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Загальні відомості.

У даній темі розглядаються наступні питання. Класи як основний елемент ООП, об'єднання класів у бібліотеки. Принцип спадкування та його використання при створенні додатків. Відмінності у мовах програмування щодо спадкування. Принцип інкапсуляції, модифікатори та області видимості. Використання "*get - set*" при створенні класів і структур. Використання спадкування та інкапсуляції при створенні додатків та бібліотек. Поліморфізм як окремий підхід у створенні коду, абстрактні класи та інтерфейси як елементи поліморфного коду.

Клас представляє собою опис, що може вміщувати різні поля і методи. Під полем розуміють опис змінної. Клас може містити посилання на інші класи та зв'язки з іншими класами, згідно принципів ООП, що описані в першій лекції.

При вивченні класів необхідно звернути увагу на такі питання як конструктори за замовчуванням, перевантаженя конструкторів, ініціалізатори, деструктори. Властивості та методи класів, використання у властивостях пари "*get - set*". Області видимості змінних та модифікатори доступу. Використання ключового слова *this*

Використання об'єктів класу та функціоналу класу. Для звернення до функціональності класу – полів, методів (а також інших елементів класу) застосовується точкова нотація точки – після об'єкта класу ставиться крапка, а потім елемент класу. Використання ланцюгів виклику конструкторів класу.

Особлива увага приділяється використанню модифікаторів доступу при створенні бібліотек класів.

Повна інформація до курсу міститься у додаткових матеріалах, що розміщені на ресурсі кафедри, доступ надається тільки студентам, що вивчають дисципліну. Посилання на відкриті інтернет-джерела наведено нижче.

Класи: https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/128-urok-13-klasy-v-si-sharp-oholoshennia-klasiv-ta-stvorennia-ob-iektiv

Методи: https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/129-urok14-metody-v-si-sharp-riznytsia-mizh-prostymy-i-statychnymy

Конструктори і показчик this:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/131-urok-15-konstruktory-v-si-sharp-pokazhchyk-this

Властивості та аксесори:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/132-urok-16-vlastyvosti-v-si-sharp-aksesor-get-i-set-avtomatychni-vlastyvosti

Спадкування:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/133-urok-17-uspadkuvannia-v-si-sharp-konstruktor-bazovoho-klasu

Поліморфізм:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/136-urok-19-polimorfizm-v-si-sharp-shcho-tse-take

Інкапсуляція:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/142-urok-24-inkapsuliatsiia-v-si-sharp-modyfikatory-dostupu

Віртуальні методи:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/137-urok-20-virtualni-metody-v-si-sharp-perevyznachennia-metodiv

Абстракції:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/138-urok-21-abstraktni-klasy-metody-i-vlastyvosti-v-si-sharp

Інтерфейси:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/139-urok-22-interfeisy-v-si-sharp-mnozhynne-uspadkuvannia

Перевантаження:

https://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c/140-urok-23-perevantazhennia-metodiv-u-si-sharp

ТЕМА 3 ВСТУП ДО ПАТЕРНІВ ПРОЄКТУВАННЯ

Вступ до патернів проектування, Взаємодія класів і об’єктів, класифікація патернів. https://refactoring.guru/uk/design-patterns/what-is-pattern

Перед вивченням наступної теми, потрібно засвоїти основні принципи побудови UML-діаграм та такі поняття як спадкування, агрегація, композиція, реалізація та асоціація.

По суті патерни проєктування – це стратегічний підхід до створення програмних продуктів або шаблони алгоритмів для вирішення поставлених завдань. При такому підході до написання коду, програмний продукт розглядається не як набір окремих складових а як єдина система призначена для досягнення поставленої мети. Використання патернів полегшує розуміння взаємозв’язків усередині програми та можливості по мінімізації та оптимізації коду.

Pattern matching – принцип за яким виконують співставлення деякого значення з певним шаблоном. За виконанням умови відбувається виконання послідовності дій. Мовою C# передбачено виконання різних типів співставлянь. Детальну інформацію можна знайти у вільному доступі, наприклад: https://krypton.com.ua/rozdil-9-pattern-matching/

Відповідно, можна виділити окремі принципи:

- type pattern – паттерн типів;
- constant pattern - зіставлення з деякою константою;
- паттерн властивостей дає змогу порівнювати зі значеннями певних властивостей об'єкта;
- патерни кортежів;

- позиційний патерн застосовується до типу, у якого визначено метод деконструктора.

Часто Pattern matching не відносять до патернів проєктування, хоча використання цього принципу дуже поширене в програмуванні.

Класифікацію патернів, та деякі принципи їх застосування наведено в : https://refactoring.guru/uk/design-patterns/classification

ТЕМА 4. ПАТЕРНИ ПРОЄКТУВАННЯ

Класично, патерни поділяють на групи:

- Патерни, що породжують.
- Структурні патерни.
- Патерни поведінки

Ще є окремий принцип, який відносять до патернів але який по суті ним не є – принцип SOLID

Таким чином, разом з Pattern matching маємо п’ять груп патернів проєктування.

До патернів, що порроджують входять патерни:

- Абстрактна фабрика (Abstract Factory)
- Будівельник (Builder)
- Фабричний метод (Factory Method)
- Прототип (Prototype)
- Одинак (Singleton)

Опис патернів, що породжують у вільному доступі:

https://refactoring.guru/uk/design-patterns/creational-patterns

https://krypton.com.ua/rozdil-2-paterny-shho-porodzhuyut/

Патерни поведінки складаються з:

- ланцюжок обов'язків (Chain of responsibility)

- Команда (Command)
- Інтерпретатор (Interpreter)
- Ітератор (Iterator)
- Посередник (Mediator)
- Зберігач (Memento)
- Спостерігач (Observer)
- Стан (State)
- Стратегія (Strategy)
- Шаблонний метод (Template method)
- Відвідувач (Visitor)

Патерни поведінки розглядають виконувані завдання з точки зору зв'язків між об'єктами і розподілу обов'язків між ними. Для цього можуть використовуватися механізми, засновані як на спадкуванні, так і на композиції.

Патерни поведінки – це один з найбільших класів патернів. Це зумовлено сучасними вимогами до створення програмних продуктів і в першу чергу необхідністю створення більш універсальних та інтелектуальних додатків. Крім того патерни поведінки допомагають створювати найбільш ефективну та безпечну взаємодію між елементами програми.

Класифікацію патернів поведінки з прикладами добре розкрито на сайтах:
https://refactoring.guru/ru/design-patterns/behavioral-patterns
https://krypton.com.ua/rozdil-3-paterny-povedinky/

Структурні патерни.

Цей клас патернів відноситься до об’єктних. В їх основі – дослідження процесу створення більших об’єктів з менших. Спрощуючи, можна представити програму, як набір класів і об’єктів, що входять до складу інших класів і структур, створюючи своєрідну ієрархію.

Вони складаються з патернів:

- Адаптер (Adapter)
- Міст (Bridge)

- Компонувальник (Composite)
- Декоратор (Decorator)
- Фасад (Facade)
- Пристосуванець (Flyweight)
- Заступник (Proxy)

Опис патернів даного класу наведено на сайті:

https://refactoring.guru/ru/design-patterns/structural-patterns

https://krypton.com.ua/rozdil-4-strukturni-paterny/

Принцип SOLID – це акронім :

S: Single Responsibility Principle (Принцип єдиної відповідальності).

O: Open-Closed Principle (Принцип відкритості-закритості).

L: Liskov Substitution Principle (принцип підстановки Барбари Лісков).

I: Interface Segregation Principle (Принцип поділу інтерфейсу).

D: Dependency Inversion Principle (Принцип інверсії залежностей).

Найкраще розглядати ці принципи окремо.

S: Клас повинен бути відповідальний лише за щось одне. Якщо клас відповідає за вирішення декількох завдань, його підсистеми, що реалізують рішення цих задач, виявляються пов'язаними один з одним. Зміни в одній такій підсистемі ведуть до змін в іншій. Цей принцип можна застосувати не тільки до класів, а й до компонентів програмного забезпечення в більш широкому сенсі.

O: Програмні суті (класи, модулі, функції) повинні бути відкриті для розширення, але не для модифікації. Цей принцип зачіпає основний принцип ООП – інкапсуляцію. При створенні об’єктів необхідно виділяти елементи, що можуть бути використані для створення інших але не можуть бути модифіковані ззовні, всі інші елементи необхідно закрити для доступу.

L: Мета цього принципу полягає в тому, щоб класи-спадкоємці могли б використовуватися замість батьківських класів, від яких вони утворені, не порушуючи роботу програми. Якщо виявляється, що в коді перевіряється тип класу, значить принцип підстановки порушується.

I: Створюйте вузькоспеціалізовані інтерфейси, призначені для конкретного клієнта. Клієнти не повинні залежати від інтерфейсів, які вони не використовують. Цей принцип спрямований на усунення недоліків, пов'язаних з реалізацією великих інтерфейсів.

D: Об'єктом залежності повинна бути абстракція, а не щось конкретне. Модулі верхніх рівнів не повинні залежати від модулів нижніх рівнів. Обидва типи модулів повинні залежати від абстракцій. Абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій.

В процесі розробки програмного забезпечення існує момент, коли функціонал додатка перестає поміщатися в рамках одного модуля. Коли це відбувається, доводиться вирішувати проблему залежностей модулів. В результаті, наприклад, може виявитися так, що високорівневі компоненти залежать від низькорівневих компонентів.

Досить детальне пояснення на прикладах:

https://dou.ua/lenta/articles/solid-principles/

Приклади кодів на патернах:

https://refactoring.guru/ru/design-patterns/examples

https://refactoring.guru/uk/design-patterns/csharp

РОЗРАХУНКОВЕ ЗАВДАННЯ

Індивідуальне домашнє завдання (ІДЗ) призначене для демонстрації навичок отриманих під час вивчення предмета. Завдання оформлюється за стандартом НТУ «ХПІ» на аркушах формату А4. Об'єм завдання повинен складати не менше 12 аркушів. Зміст завдання:

Індивідуальне завдання – частина загальної оцінки засвоєння матеріалу курсу. Номер варіантів – номер в академічному журналі групи. Основа завдання – алгоритм, реалізований мовою C# або будь-якою сучасною мовою.

Завдання складається з наступних частин:

1. Зміст
2. Теоретичні основи заданого алгоритму;
3. Теоретичні відомості про патерн, команди, методи, класи та подібне, використані при створенні програми;
4. Код програми з коментарями і поясненнями.
5. Перелік використаних джерел – книжок, підручників, сайтів;
6. Скріншот екрану перевірки на плагіат.
7. **Показник унікальності текстової частини повинен бути більше 75 %** (додаток Б)

Шрифт основного тексту – Times New Roman, 14 пт.

Шрифт коду – Courier New, 14 пт

Поля: верхнє і нижнє 2 см, ліве 2,5 см, праве 1,5 см.

Міжрядковий інтервал 1,5. Відступи відсутні !!!

Відступ на початку рядка (абзац) 1,25 см.

Вирівнювання – по ширині сторінки.

Титульний аркуш оформлюється відповідно до додатку А.

Теоретична частина повинна містити основні теоретичні відомості по темі завдання, середовищу розробки та обгрунтуванню до реалізації завдання.

Реалізація завдання містить програмний код, описи алгоритму, зображення екрану і т.і. Для реалізації завдання студент обирає будь-яку мову ООП.

Основою завдань є розробка програм та програмних модулів з використанням патернів програмування, що були вивчені. Студенти заздалегідь, узгоджують тему для демонстрації роботи того чи іншого патерну. Наприклад, «Програма обчислення об'єму геометричних тіл. Патерн «Factory Method».

Студент має змогу обрати довільний приклад для демонстрації використання патернів програмування. Основним критерієм є відсутність плагіату у результатах розробки. Виконане завдання проходить перевірку на плагіат на відкритих ресурсах. Після чого студент захищає роботу.

ВИМОГИ !!!

У процесі захисту роботи студент повинен продемонструвати:

- знання патернів програмування, їх ознаки та області застосування;
- принципи ООП, мету їх застосування при створенні програмних продуктів;
- володіння сучасною мовою програмування;
- розуміння алгоритму і принципу роботи створеної програми.

Теми домашнього завдання обирають у відповідності зі списком академічної групи. Перелік тем:

1. Фабричний метод (Factory Method)
2. Абстрактна фабрика (Abstract Factory)
3. Одинак (Singleton)
4. Прототип (Prototype)
5. Будівельник (Builder)
6. Стратегія (Strategy)
7. Спостерігач (Observer)
8. Команда (Command)
9. Шаблоновий метод (Template Method)
10. Итератор (Iterator)

11. Стан (State)

12. Ланцюжок Обов'язків (Chain of responsibility)

13. Інтерпретатор (Interpreter)

14. Посередник (Mediator)

15. Зберігач (Memento)

16. Відвідувач (Visitor)

17. Декоратор (Decorator)

18. Адаптер (Adapter)

19. Фасад (Facade)

20. Компоновщик (Composite)

21. Заступник (Проксі)

22. Міст (Bridge)

23. Пристосуванець (Flyweight)

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Повний посібник з C# 10 та .NET 6 [Електронний ресурс] // Krypton. – 2022. – Режим доступу до ресурсу: https://krypton.com.ua/tutorial/ci-10-net-6/.

2. Паттерни проектування у C# та .NET [Електронний ресурс] // Krypton. – 2022. – Режим доступу до ресурсу: https://krypton.com.ua/tutorial/ci-10-net-6/.

3. C# практичні / http://www.e-helper.com.ua/c_sharp_programmind_labs

4. C# лекції / http://www.e-helper.com.ua/c_sharp_programming_lections

5. Підручник з Umbrello UML Modeller https://docs.kde.org/trunk5/uk/umbrello/umbrello/index.html

6. Booch G. Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide (Object Technology Series): 2nd Edition, Addison-Wesley Professional, 2005, 494 p.

7. Troelsen A. Japikse P. C# 9 .NET 5: F : 10 , A , 2021, 1411 p.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до самостійної роботи з курсу «Об'єктно-орієнтоване програмування»
для студентів спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка усіх форм навчання

Укладачі: ПУГАНОВСЬКИЙ Олег Валентинович

Відповідальний за випуск : О.М. Дзевочко
Роботу до видання рекомендував: І.Л. Красніков

Редактор О.І.Шпільова

План 2024 р., поз .Підп. до друку . Формат 60х84 1/16.
Папір офсет-ний. ГарнітураTimes.
Ум. друк. арк. 1. Наклад 25 прим. Зам. №__. Ціна договірна.

Видавець НТУ"ХПІ", 61002, Харків вул. Кипичова, 2
Свідоцтво про державну реєстрацію ДК № 5478 от 21.08.2017 г.

Надруковано з готового оригінал-макету у друкарні ФОП В.В. Петров Єдиний державний реєстр юридичних осіб та фізичних осіб – підпри-ємців.
Запис №24800000000106167 від 08.01.2009 р.
61144, м. Харків, вул. Гв. Широнінців, 79в, к. 137, тел. (057)78-17-137.
e-mail: bookfabrik@mail.ua