ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ
Система забезпечення якості освітньої діяльності та вищої освіти
Кафедра комп'ютерних технологій і мехатроніки

ЗАТВЕРДЖЕНО
Гарант освітньо-професійної програми «Інженерія програмного забезпечення» першого(бакалаврського) рівня освіти:
зав. каф. КТМ, д.т.н., проф.
_____Ніконов О.Я.

СИЛАБУС
ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ
/
OBJECT-ORIENTED PROGRAMMING
SYLLABUS

освітній рівень	**бакалавр** / bachelor
галузь знань	**12 Інформаційні технології** / Information Technologies
спеціальність	**121 Інженерія програмного забезпечення** / Software Engineering
спеціалізація	**Програмне забезпечення систем** / Systems Software

Харків 2020

Автор: Мнушка Оксана Василівна, старший викладач кафедри комп'ютерних технологій і мехатроніки

Силабус розглянуто та затверджено на засіданні кафедри комп'ютерних технологій і мехатроніки, протокол № 20 від «28» серпня 2020 р.

СИЛАБУС

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ /

OBJECT-ORIENTED PROGRAMMING

SYLLABUS

освітній ступінь	**бакалавр** / bachelor
галузь знань	**12 Інформаційні технології /** Information Technologies
спеціальність	**121 Інженерія програмного забезпечення /** Software Engineering
освітня програма	**Програмне забезпечення систем /** Systems Software

Анотація курсу

1. Викладачі

1.1. Лектор: Мнушка Оксана Василівна

- Старший викладач кафедри комп'ютерних технологій та мехатроніки;
- педагогічний стаж – 17 років
- контактний телефон +38-057-707-37-43
- e-mail: mnushka.ov@gmail.com
- наукові інтереси: теоретичні та практичні питання програмної та комп'ютерної інженерії, комп'ютерні мережі, інформаційні технології керування, безпека даних, групові комунікації.

1.2. Асистент лектора: -

2. Дисципліна «Об'єктно-орієнтоване програмування»

– рік навчання: 2;

– семестр навчання: 3;

– кількість годин за семестр: 180, в т. ч.

лекційних: 16;

практичних занять: 64;

на самостійне опрацювання: 100;

– кількість аудиторних годин на тиждень

лекційних: 2 (раз на два тижні);

практичних занять: 4.

3. Час та місце проведення

– аудиторні заняття – відповідно до розкладу ХНАДУ, ауд. 313, 214;

– позааудиторна робота – самостійна робота студента із використанням інструментальних засобів розробки для ОС Linux та ОС Windows – Python, Visual Studio Code, plantUml, Microsoft Visual Studio Community Edition, Oracle Virtual Box, WSL.

4. Пререквізити та постреквізити навчальної дисципліни:

- **пререквізити**: «Алгоритмізація та програмування», окремі розділи з курсу «Вища математика»
- **постреквізити (**дисципліни та компетентності, які необхідні в професійній діяльності фахівця): Аналіз вимог до ПЗ, Управління ІТ проектами, Професійна практика програмної інженерії, Software Engineer, Software Architect

5. Характеристика дисципліни:

5.1. Призначення навчальної дисципліни: знайомство із технологіями розробки складних програмних систем на основі принципів об'єктно-орієнтованого дизайну та реалізація отриманих моделей засобами обраної мови програмування. Вивчення теоретичних засад побудови класів, в т. ч. інтерфейсів, для програмних систем та способів їх взаємозв'язків, побудова ієрархій класів, різні варіанти успадкування, в т. ч. множинне, спеціальні питання реалізації, пов'язані із конкретною мовою програмування. Опанування розглянутих питань є основою для вивчення питань побудови комплексних програмних систем на основі комплексного аналізу вимог (Software Requirements), вивчення їх життєвого циклу (Lifecycle) програмного забезпечення тощо. Знання та практичні навички, що отримуються при вивченні дисципліни є необхідною умовою підготовки кваліфікованого інженера-програміста (Software Engineer), системного архітектора (System Architect), архітектора програмного забезпечення (Software Architect).

5.2. Мета вивчення дисципліни: вивчення основних принципів об'єктно-орієнтованого програмування та основ об'єктно-орієнтованого дизайну програмного забезпечення, вивчення основних абстрактних структур даних та особливостей їх використання їх під час розробки програмного забезпечення, вступ до проектування програмного забезпечення та вивчення поширених шаблонів проектування, елементи мови UML, що мають відношення до ООП.

5.3. **Задачі вивчення дисципліни:** основними завданнями вивчення дисципліни Об'єктно-орієнтоване програмування є формування сукупності знань та вмінь для використання методів об'єктно-орієнтованого дизайну та принципів S.O.L.I.D для розробки прикладного програмного забезпечення автоматизованих

інформаційних систем засобами обраної мови об’єктно-орієнтованого програмування.

По завершені вивчення дисципліни студенти повинні володіти наступними *компетентностями:*

– здатність до абстрактного мислення, аналізу та синтезу.

– здатність застосовувати знання у практичних ситуаціях.

– здатність до пошуку, оброблення та аналізу інформації з різних джерел.

– здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

– здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.

– здатність розробляти архітектури, модулі та компоненти програмних систем.

– володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних.

– здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв’язання завдань інженерії програмного забезпечення.

– здатність до алгоритмічного та логічного мислення.

Результати навчання:

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки

– знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення.

– уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення.

– вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання.

– знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.

– уміння документувати та презентувати результати розробки програмного забезпечення.

5.4. Зміст навчальної дисципліни: відповідає робочій програмі.

5.5. План вивчення дисципліни:

Результати навчання	**Навчальна діяльність**	**Робочий час студента (год.)**	**Оцінювання (бал.)**
1	**2**	**3**	**4**
Тема 1. Основні принципи об'єктно-орієнтованого програмування та дизайну (SOLID)			
Загальні та спеціальні компетентності: – здатність до абстрактного мислення, аналізу та синтезу; – здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення *Результати навчання:* – знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення. – вибирати вихідні дані для проектування,	**Лекція 1.** Основні принципи об'єктно-орієнтованого програмування та дизайну (SOLID) План лекції: • Принципи SOLID • Об'єкт • Клас • Діаграма класів • UML-нотація • Відносини між класами Список рекомендованих джерел: Основний: 1-8 Додатковий: 1-5 Інтернет-ресурси: 1-6 **Практична робота 1.** Створення простого класу, робота із атрибутами класу. *Мета роботи* – отримати практичні навички використання елементів об'єктно-орієнтованого дизайну, розробки та реалізації діаграми класу у відповідності до принципів S.O.L.I.D. *Завдання:* розробити клас, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Основні принципи об'єктно-орієнтованого програмування та дизайну, основні принципи SOLID.	2 4 2	2 4 1

керуючись формальними методами опису вимог та моделювання. –			
Тема 2. Реалізація відносин між об'єктами і класами. Агрегація та композиція			
Загальні та спеціальні компетентності: – здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення. – здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування. *Результати навчання*: – уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. – вибирати вихідні дані для проектування, керуючись формальними	**Лекція 2.** Реалізація відносин між об'єктами і класами. Агрегація та композиція План лекції: • Класи у Python • Асоціація • Агрегація • Композиція • Приклади Список рекомендованих джерел: Основний: 1-8 Додатковий: 1-5 Інтернет-ресурси: 1-6 **Практична робота 2.** Агрегація, композиція *Мета роботи* – отримати практичні навички використання агрегації та композиції для побудови класів. *Завдання* - розробити клас, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу. Приведіть програмний код та результати виконання. Дозволяється використовувати різні види відносин між класами. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Діаграми класів, взаємовідносини між класами	2 4 2	2 4 1

методами опису вимог та моделювання.			
Тема 3. Реалізація відносин між об'єктами і класами. Спадкування.			
Загальні та спеціальні компетентності: – здатність до абстрактного мислення, аналізу та синтезу. – здатність застосовувати знання у практичних ситуаціях. – здатність розробляти архітектури, модулі та компоненти програмних систем. *Результати навчання:* – уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. – вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання. – знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.	**Лекція 3.** Реалізація відносин між об'єктами і класами. Спадкування. План лекції: • Повторення – агрегація, композиція, асоціація • Проблема дублювання коду • Одиночне успадкування • Ієрархія одиночного успадкування • Проблеми використання успадкування • Приклади Список рекомендованих джерел: Основний: 1-8 Додатковий: 1-5 Інтернет-ресурси: 1-6 **Практична робота 3.** Успадкування *Мета роботи* – отримати практичні навички використання успадковування для побудови ієрархії класів. *Завдання:* розробити ієрархію класів, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу. Приведіть програмний код та результати виконання. Можна використовувати різні види відносин між класами. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Успадкування, проблеми великих ієрархій	2 4 2	2 4 1
Тема 4. Реалізація відносин між об'єктами і класами. Поліморфізм.			

Загальні та спеціальні компетентності: – здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування. – здатність розробляти архітектури, модулі та компоненти програмних систем. *Результати навчання:* – аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки – знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення.	**Лекція 4.** Реалізація відносин між об'єктами і класами. Поліморфізм. План лекції: • Повторення – одиночне успадкування • Поліморфізм та його типи - ad-hoc, параметричний, підтипів • Качина типізація (duck typing) • Приклади Список рекомендованих джерел: Основний: 1-8 Інтернет-ресурси: 4-6	2	2
	Практична робота 4. Поліморфізм *Мета роботи* – отримати практичні навички використання поліморфізму. *Завдання:* розробити клас, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу. Приведіть програмний код та результати виконання. Можна використовувати різні види відносин між класами. Додати поліморфну функції згідно завдання.	4	4
	Завдання для самостійної роботи: самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Поліморфізм у різних мовах програмування	2	2

Тема 5. Реалізація відносин між об'єктами і класами. Спадкування множинне.			
Загальні та спеціальні компетентності: – здатність розробляти архітектури, модулі та компоненти програмних систем. – здатність до абстрактного мислення, аналізу та синтезу. – здатність застосовувати знання у практичних ситуаціях. *Результати навчання:* – аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки – знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.	**Лекція 5.** Реалізація відносин між об'єктами і класами. Спадкування множинне. План лекції: • Повторення – поліморфізм • Множинне успадкування • Ромбовидні діаграми класів • Алгоритм MRO у Python • Приклади Список рекомендованих джерел: Основний: 1-8 Інтернет-ресурси: 7-9	2	2
	Практична робота 5. Множинне спадкування *Мета роботи* – отримати практичні навички використання множинного успадкування та міксінів (домішок). *Завдання:* розробити клас, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу. Приведіть програмний код та результати виконання. Можна використовувати різні види відносин між класами: оголосити ієрархію класів згідно варіанта; записати лінеаризації для всіх класів, за необхідності міняти порядок оголошення предків; написати програму, інстанціювати клас, що знаходиться на нижньому рівні ієрархії; ініціалізація базових класів є зоною відповідальності класу, що інстанціює ієрархію (як мінімум одне приватне поле даних має бути у кожному класі); написати метод, який відображає внутрішній стан кожного об’єкта в ієрархії (значення приватного поля даних); додати міксін – клас, що буде друкувати mro ієрархії у вигляді B->A->…	4	4
	Завдання для самостійної роботи: самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Спадкування множинне у різних мовах програмування	2	2
Тема 6. Перевантаження операторів.			
Загальні та спеціальні компетентності: – здатність ідентифікувати, класифікувати та	**Лекція 6.** Перевантаження операторів. План лекції: • Повторення – множинне успадкування, ромбовидні ієрархії, алгоритм MRO • Перевантаження операторів • Замикання	2	2

формулювати вимоги до програмного забезпечення. – здатність до алгоритмічного та логічного мислення. *Результати навчання:* – вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання. – знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань. – уміння документувати та презентувати результати розробки програмного забезпечення.	• Декоратори • Приклади Список рекомендованих джерел: Основний: 1-8 Інтернет-ресурси: 9-14 **Практична робота 6.** Перевантаження операторів *Мета роботи* – отримати практичні навички використання перевантаження операторів, замикань та декораторів. *Завдання:* розробити клас, що відповідає варіанту, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразіть діаграму класу. Приведіть програмний код та результати виконання. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Перевантаження операторів як засіб спрощення написання та читання програмного забезпечення	4 2	4 2
Тема 7. Обробка виняткових ситуацій			
Загальні та спеціальні компетентності: – здатність застосовувати знання у практичних ситуаціях. – здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення.	**Лекція 7.** Обробка виняткових ситуацій План лекції: • Повторення – перевантаження операторів, замикання, декоратори • Винятки • Стек викликів, стек блоків • Обробка виняткових ситуацій стандартними засобами • Обробка виняткових ситуацій кастомізованими засобами • Приклади Список рекомендованих джерел: Основний: 1-8 Додатковий 1-9 Інтернет-ресурси: 18-19	2 4	2 4

Результати навчання: – знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань. – уміння документувати та презентувати результати розробки програмного забезпечення.	**Практична робота 7.** Обробка виняткових ситуацій *Мета роботи* – отримати практичні навички використання механізмів обробки виняткових ситуацій. *Завдання*: розробити клас для обробки виняткових ситуацій, що відповідає варіанту, проаналізувати завдання та визначити всі можливі виняткові ситуації при роботі із класом, визначити та обґрунтувати перелік атрибутів та методів (інтерфейс класу). Відобразити діаграму класу (або ієрархію). Приведіть програмний код та результати виконання. Результати мають містити приклади обробки всіх заданих виняткових ситуацій. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Виняткові ситуації та проблеми їх обробки	2	2
Тема 8. Метакласи, елементи функційного програмування			
Загальні та спеціальні компетентності: – здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення. – володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних. – здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення.	**Лекція 8.** Метакласи, елементи функційного програмування План лекції: • Повторення – винятки та обробка виняткових ситуацій • Метакласи • Елементи функційного програмування • Лямбда-функції • Приклади Список рекомендованих джерел: Основний: 1-8 Додатковий: 1-5 Інтернет-ресурси: 20-22 **Практична робота 8.** Метакласи. *Мета роботи* – отримати практичні навички використання метакласів для генерації класів у Python. *Завдання:* на вибір студента для індивідуального завдання із однієї практичної роботи №3, №4, №6 за допомогою метакласів додати: функцію вимірювання часу виконання методів; додати методи за допомогою яких можна друкувати всі значення атрибутів класу; функцію логування роботи. Для індивідуального завдання із роботи №6 реалізувати декілька конструкторів. Показати приклад роботи програми. Відобразити діаграму класу (або ієрархію). Приведіть програмний код та результати виконання. Результати мають містити	2 4	2 4

Результати навчання: – аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки – знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення.	приклади обробки всіх заданих виняткових ситуацій. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Метакласи, метапрограмування та інші технології керування класами у Python	2	2
Тема 9. Абстрактні типи даних із використанням ООП			
Загальні та спеціальні компетентності: – здатність до пошуку, оброблення та аналізу інформації з різних джерел. – здатність розробляти архітектури, модулі та компоненти програмних систем. – володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для	Лекція не передбачена робочою програмою дисципліни. Список рекомендованих джерел: Основний: 6 Додатковий: 1 Інтернет-ресурси: 23 **Практична робота 9.** Реалізація абстрактних структур даних (АТД). Зв'язані списки *Мета роботи* – знайомство із особливостями реалізації АТД. *Задання:* відповідно до індивідуального завдання розробити програму реалізації зв'язаного списку для роботи із екземплярами класів користувача. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Особливості реалізації зв'язаних списків.	4 2	4 2

зберігання, видобування та опрацювання даних. *Результати навчання:* – аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки – знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення. – уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення.	**Практична робота 10.** Реалізація абстрактних структур даних. Дерева. *Мета роботи* – знайомство із особливостями реалізації АТД - дерева. *Задання:* відповідно до індивідуального завдання розробити програму реалізації дерева та методів роботи із ним – формування та обхід дерева різними способами **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Реалізація алгоритмів пошуку на деревах **Практична робота 11.** Реалізація абстрактних структур даних. Черги. *Мета роботи* – знайомство із особливостями реалізації АТД – односпрямованою чергою. *Задання:* відповідно до індивідуального завдання розробити програму реалізації роботи із односпрямованою чергою. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Призначення черги, реалізація стандартних алгоритмів по роботі із чергами. **Практична робота 12.** Реалізація абстрактних структур даних. Стек. *Мета роботи* – знайомство із особливостями реалізації АТД – стеком. *Задання:* відповідно до індивідуального завдання розробити програму реалізації роботи із стеком. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Стек та алгоритми на стеках	4 2 4 2 4 2	4 2 4 2 2 2
Тема10. Використання ООП та ООД при розробці програм			
Загальні та спеціальні компетентності: – здатність до пошуку, оброблення та аналізу інформації з різних джерел. – здатність розробляти	Лекція не передбачена робочою програмою дисципліни. Список рекомендованих джерел: Основний: 6 Додатковий: 1 Інтернет-ресурси: 23 **Практична робота 13.** Шаблони проектування. Патерни, що породжують	4	4

архітектури, модулі та компоненти програмних систем. – володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних. *Результати навчання:* – аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки – знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення. – уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. – уміння	*Мета роботи* – знайомство із особливостями реалізації шаблонів, що породжують *Задання:* розробити програму для роботи із абстрактною фабрикою **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Патерни, що породжують **Практична робота 14**. Шаблони проектування. Структурні патерни *Мета роботи* – знайомство із особливостями реалізації структурних патернів *Задання:* згідно варіанту розробити програму для роботи із мостом, компанувальником, адаптером. **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Структурні патерни **Практична робота 15**. Шаблони проектування. Поведінкові патерни. Ітератор. *Мета роботи* – знайомство із особливостями реалізації поведінкового патерна Ітератор *Задання:* згідно варіанту розробити програму для роботи із Ітератором для заданого класу користувача **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* Поведінкові патерни **Практична робота 16**. Шаблони проектування. Поведінкові патерни. Відвідувач, шаблонний метод, спостерігач. *Мета роботи* – знайомство із особливостями реалізації поведінкових патернів відвідувач, шаблонний метод, спостерігач *Задання:* згідно варіанту розробити програму для роботи із одним із заданих патернів - відвідувач, шаблонний метод, спостерігач для заданого класу користувача **Завдання для самостійної роботи:** самостійне опрацювання літературних джерел, які зазначені у списку. *Питання, винесені на самостійне опрацювання:* поведінкові патерни у програмах на Python	2 4 2 4 6 4 6	2 4 2 4 2 4 2

документувати та презентувати результати розробки програмного забезпечення.			
Курсова робота		30	
Підготовка до іспиту		30	
Разом		180/6 кредитів	100
Підсумковий контроль		екзамен	

6. Список рекомендованих джерел:

6.1. Основний

1. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: пер. с англ. / Буч Г. и др. М.: ООО «И.Д. Вильямс», 2008. – 720 с.
2. Лутц М. Изучаем Python, том 1, 5-е изд.: пер. с англ. СПб.: “Диалектика”, 2019. – 832 с.
3. Лутц М. Изучаем Python, том 2, 5-е изд. : пер. с англ. СПб. : “Диалектика”, 2020. – 720 с.
4. Хеллман Д. Стандартная библиотека Python 3: справочник с примерами, 2-е изд. : пер. с англ. СПб. : ООО “Диалектика”, 2019. – 1376 с.
5. Хайнеман Д., Поллис Г., Сеяков С. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд.: пер. с англ. СПб.: ООО “Альфа-книга”, 2017. — 432 с.
6. Приемы объектно-ориентированного программирования. Паттерны проектирования [Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес] . СПб: Питер, 2010. – 368 с.
7. Рамальо Л. Python. К вершинам мастерства. пер. с англ. Слинкин А. А. – М.: ДМК-Пресс, 2016. –768 с.
8. Мнушка, О.В. Об’єктно-орієнтоване програмування : конспект лекцій для студентів за спеціальністю 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки» - Харків, ХНАДУ, 2020.
9. Мнушка, О.В. Методичні вказівки для практичних робіт з дисципліни «Об’єктно-орієнтоване програмування» для студентів за спеціальністю 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки» - Харків, ХНАДУ, 2020.
10. Мнушка, О.В. Методичні вказівки для самостійної роботи з дисципліни «Об’єктно-орієнтоване програмування» для студентів за спеціальністю 121

«Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» - Харків, ХНАДУ, 2020.

11. Мнушка, О.В., Методичні вказівки по виконанню курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» для студентів спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» галузі знань 12 «Інформаційні технології» – Харків, ХНАДУ, 2020. – 20 с.

6.2. Допоміжна література

1. Кормен Т. Х. Алгоритмы: вводный курс. : пер. с англ. М. : ООО «И.Д. Вильямс», 2014. – 208 с.: ил.
2. Дейтел П., Дейтел Х. Python: Искусственный интеллект, большие данные и облачные вычисления. СПб.: Питер, 2020. – 864 с.
3. Прикладна інформатика: підручник / Б.С. Бусигін , Г.М. Коротенко, Л.М Коротенко. Дніпропетровськ: Національний гірничий університет, 2004. – 559 с.
4. Седер Н. Python. Экспресс-курс. 3-е изд. СПб.: Питер, 2019. – 480 с.
5. Прохоренок Н. А., Дронов В.А. Python 3 и PyQt 5. Разработка приложений. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2018. – 832 с.
6. Мартелли А., Рейвенскрофт А., Холден С. Python. Справочник. Полное описание языка, 3-е издание. СПб.: ООО "Диалектика", 2019. - 896 с.

6.3. Інформаційні ресурси

1. Welcome to Python.org.–URL: https://www.python.org.
2. Data model. URL: https://docs.python.org/3.8/reference/datamodel.html
3. S.O.L.I.D principles: what are they and why projects should use them. URL: https://medium.com/@mari_azevedo/s-o-l-i-d-principles-what-are-they-and-why-projects-should-use-them-50b85e4aa8b6
4. The clean code blog. URL: https://blog.cleancoder.com/
5. SOLID. URL: https://habr.com/ru/post/348286/
6. Association vs. Dependency vs. Aggregation vs. Composition. URL: https://nirajrules.wordpress.com/2011/07/15/association-vs-dependency-vs-aggregation-vs-composition/
7. Композиция или наследование: как выбрать? URL: https://habr.com/ru/post/325478/
8. Об'єктно-орієнтоване програмування.URL: http://programming.in.ua/programming/basisprogramming/25-oop.html
9. Утиная типизация в Python-3 примера. URL: https://nuancesprog.ru/p/9142/
10. Duck typing. URL: https://devopedia.org/duck-typing
11. Ликбез по типизации в языках программирования. URL: https://habr.com/ru/post/161205/

12. The Python 2.3 Method Resolution Order. URL: https://www.python.org/download/releases/2.3/mro/
13. Порядок разрешения методов в Python. URL: https://habr.com/ru/post/62203/
14. What is a Mixin. URL: https://chrisbartos.com/articles/what-is-a-mixin/
15. PEP-318. Decorators for Functions and Methods. URL: https://www.python.org/dev/peps/pep-0318/
16. Понимаем декораторы в Python'е, шаг за шагом. Шаг 1. URL: https://habr.com/ru/post/141411/
17. Понимаем декораторы в Python'е, шаг за шагом. Шаг 2. URL: https://habr.com/en/post/141501/
18. Мир Python: исключения . URL: https://ru.hexlet.io/courses/advanced_python/lessons/python_exceptions/theory_unit
19. Исключения. URL: https://wombat.org.ua/AByteOfPython/exceptions.html
20. Python Metaclasses. URL: https://realpython.com/python-metaclasses/
21. Продвинутый Python, часть 3: классы и метаклассы. URL: https://ru.hexlet.io/blog/posts/prodvinutyy-python-chast-3-klassy-i-metaklassy
22. Метаклассы и метапрограммирование в Python. URL: https://gitjournal.tech/metaklassy-i-metaprogrammirovanie-v-python/
23. Патерни проектування на Python. URL: https://refactoring.guru/uk/design-patterns/python

7. Контроль та оцінювання результатів навчання: включає весь спектр письмових, усних, практичних контрольних процедур у залежності від компетентністних характеристик (знання, уміння, комунікація, автономність і відповідальність) результатів навчання, досягнення яких контролюється. Вимірювання рівня досягнення результатів навчання здійснюється коефіцієнтом засвоєння або експертно за критеріями, що корелюються з дескрипторами НРК. Вибір, конкретизація та деталізація критеріїв оцінювання з урахуванням специфіки освітніх програм та їх компонентів здійснюється кафедрами на основі загальних критеріїв, наведених у СТВНЗ 7.1-01:2015 Положення про організацію освітнього процесу в ХНАДУ.

Під час вивчення дисципліни «ООП» викладачем здійснюється поточний та підсумковий контроль. Поточний контроль та оцінювання передбачає:

- перевірку рівня засвоєння теоретичного матеріалу (тестування за матеріалами лекції, який здійснюється на початку кожної наступної лекції);

- захист практичних робіт (проходить під час наступної практичної роботи);

8. Політика навчальної дисципліни:

8.1. Відвідування лекційних та практичних занять: відвідування лекційних та практичних занять є обов'язковим. Допускаються пропуски занять з таких поважних причин, як хвороба (викладачу надається копія довідки від медичного закладу), участь в олімпіаді, творчому конкурсі тощо за попереднього домовленістю та згодою викладача за умови дозволу деканату (надаються документи чи інші матеріали, які підтверджують заявлену участь у діяльності студента).

8.2. Відпрацювання пропущених занять: відпрацювання пропущених занять є обов'язковим незалежно від причини пропущеного заняття. Лекційне заняття має були відпрацьоване до наступної лекції на консультації викладача. Відпрацювання лекційного матеріалу передбачає вивчення пропущеного теоретичного матеріалу та складання тесту за цим матеріалом. Практичне заняття відпрацьовується під час консультації викладача (розклад консультацій на сайті університету).

8.3. Правила поведінки під час занять: повинні відповідати Морально-етичному кодексу учасників освітнього процесу Харківського національного автомобільно-дорожнього університету (З додатком згідно наказу ХНАДУ від 08 листопада 2019 № 147). Обов'язковим є:

– прагнути отримувати глибокі знання у відповідній області: сумлінно вчитися, не пропускати заняття без поважної причини, брати участь у навчальній та науково-дослідній роботах;

– прагнути максимально використовувати надані можливості з придбання теоретичних знань і практичних навичок з обраної спеціальності;

– виконувати вимоги, передбачені розпорядком дня університету, навчальними програмами, у суворо встановлені терміни;

– не користуватися забороненими допоміжними матеріалами і технічними засобами при проходженні процедур контролю знань, умінь і навичок, спиратися виключно на отримані знання;

не вчиняти дій, що перешкоджають здійсненню навчального процесу.

8.4. За порушення академічної доброчесності здобувачі вищої освіти можуть бути притягнені до академічної відповідальності у відповідності до Правил академічної доброчесності учасників освітнього процесу Харківського національного автомобільно-дорожнього університету (СТВНЗ 67.0-01:2019):

– повторне проходження оцінювання (контрольна робота, іспит, залік тощо);
– повторне проходження відповідного освітнього компонента освітньої програми;
– відрахування з університету;
– позбавлення академічної стипендії;
– позбавлення наданих університетом пільг з оплати навчання.