***Підсумковий проект з блоку тем ООП***

*ЗВЕРНУТИ УВАГУ!*

*Загальні завдання для всіх проектів*

Розробити консольний додаток. В нього повинен бути невеликий інтерфейс по типу:

*Варіант 1*Проект: Система Університету

Обов`язкові пункти для виконання:

1. Класи:

- Створення класів для представлення об'єктів університету, таких як Студент, Викладач, Предмет і т.д.

2. Наслідування, абстрактні класи:

- Використання наслідування для створення ієрархії класів, наприклад, Абстрактний клас "Особа", від якого успадковуються класи "Студент" та "Викладач".

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Застосування інкапсуляції для приховування деталей реалізації класів.

- Використання поліморфізму через перевизначення методів, наприклад, метод "вивестиІнформацію()" для кожного класу.

- Використання інтерфейсів для опису функціональності, такої як "МожеСтудіювати" для студентів або "МожеВикладати" для викладачів.

4. Перерахування. Вкладені класи:

- Використання перерахувань для представлення різних ступенів студентів або факультетів.

- Створення вкладених класів для представлення, наприклад, контактної інформації особи.

5. Object:

- Використання класу Object для перевизначення методів toString(), equals() та hashCode() для кращої роботи з об'єктами університету.

6. Обробка виключень:

- Введення обробки виключень для обробки можливих помилок під час роботи зі студентами, викладачами тощо.

7. Узагальнююче програмування:

- Використання узагальнень для створення універсальних класів чи методів, наприклад, для роботи зі списками студентів або предметів.

*Варіант 2*

Проект: Компанія з транспортних засобів

1. Класи:

- Створіть клас `Vehicle` для представлення загальних властивостей транспортних засобів, таких як марка, модель, рік випуску тощо.

2. Наслідування, абстрактні класи:

- Створіть абстрактний клас `Transport` як батьківський для різних видів транспортних засобів, наприклад, `Car`, `Bus`, `Bicycle`.

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Використовуйте інтерфейси для реалізації загальних функціональних можливостей транспортних засобів, таких як `Moveable`, `Stopable`, `Soundable`, і використовуйте поліморфізм для їхньої реалізації у різних класах.

4. Перерахування. Вкладені класи:

- Створіть перерахування `FuelType` для представлення різних типів палива (бензин, дизель, електроенергія тощо). Вкладіть це перерахування в клас `Car` для представлення типу палива автомобіля.

5. Object:

- Використовуйте методи класу `Object`, такі як `toString()`, `equals()`, для правильного виведення та порівняння об'єктів.

6. Обробка виключень:

- Реалізуйте механізм обробки виключень для валідації деяких властивостей транспортних засобів, наприклад, року випуску або швидкості.

7. Узагальнююче программування:

- Використовуйте узагальнення для створення універсальних методів, наприклад, метод для запуску транспортного засобу, який може бути використаний для будь-якого класу, що реалізує інтерфейс `Moveable`.

*Варіант 3*

Проект: Герої та Монстри

Уявіть ігровий світ, де існують герої та монстри, і вони мають різні характеристики та навички. У цьому проекті ви можете застосувати різні концепції об'єктно-орієнтованого програмування та узагальнюючого програмування.

1. Класи:

- Створіть класи `Hero` та `Monster`, які визначають характеристики героїв та монстрів відповідно. Наприклад, `Hero` може мати властивості, такі як ім'я, рівень, здоров'я та атака.

2. Наслідування, абстрактні класи:

- Використовуючи наслідування, створіть абстрактний клас `Character`, від якого будуть успадковуватись як герої, так і монстри. Цей клас може містити загальні характеристики, такі як ім'я та рівень, а також абстрактні методи для реалізації унікальних властивостей кожного персонажа.

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Використовуючи інкапсуляцію, приховуйте деякі характеристики та методи, аби не дозволяти прямий доступ до них. Використовуйте поліморфізм та інтерфейси для організації різних навичок та функціональностей для різних типів персонажів.

4. Перерахування. Вкладені класи:

- Використовуйте перерахування для визначення різних типів героїв та монстрів (наприклад, воїн, лучник, маг, зомбі, дракон). Вкладені класи можуть представляти додаткові властивості чи навички, що властиві лише певному типу персонажів.

5. Object:

- Використовуйте клас `Object` для створення методів `toString`, `equals` та `hashCode` для кращого представлення та порівняння об'єктів.

6. Обробка виключень:

- Реалізуйте обробку виключень для ситуацій, які можуть виникнути під час гри, наприклад, спроба атакувати мертвого персонажа або введення невірних даних.

7. Узагальнене програмування:

- Використовуйте узагальнене програмування для створення універсальних методів чи класів, які можна використовувати з різними типами даних.

*Варіант 4*

Проект: Електронна бібліотека "BookLibrary"

1. Класи:

- Створення класу `Book` для представлення книги з властивостями, такими як автор, назва, рік видання, тощо.

2. Наслідування, абстрактні класи:

- Визначення абстрактного класу `PrintedMaterial`, який є базовим класом для книги, газети, журналу та інших друкованих матеріалів.

3. Інкапсуляція, Поліморфізм, Інтерфейси:

- Використання інкапсуляції для захисту даних про книги.

- Використання поліморфізму та інтерфейсів для створення можливості додавання різних видів друкованих матеріалів до бібліотеки (книги, газети, журнали).

4. Перерахування, Вкладені класи:

- Визначення перерахування `Genre` для класифікації жанрів книг.

- Вкладення класу `Author` у клас `Book` для представлення автора з властивостями, такими як ім'я, прізвище, рік народження.

5. Object:

- Використання методів класу `Object`, таких як `toString()`, для забезпечення зручного виводу інформації про об'єкти.

6. Обробка виключень:

- Реалізація обробки виключень для валідації введених даних про книгу (наприклад, рік видання не може бути від'ємним).

7. Узагальнююче програмування:

- Використання узагальненого класу або методу для забезпечення гнучкості та повторного використання коду (наприклад, узагальнений клас `Library<T>` для управління різними видами друкованих матеріалів у бібліотеці).

*Варіант 5*

Проект: Керування Бібліотекою Мультимедійних Файлів

1. Класи:

- Створити основний клас `MediaFile`, який представляє мультимедійний файл. Цей клас може мати поля, такі як `назва`, `розмір`, `тип`, тощо.

2. Наслідування, абстрактні класи:

- Створити абстрактний клас `AudioFile`, який розширює `MediaFile`. Він може мати додаткові характеристики, специфічні для аудіофайлів.

3. Інкапсуляція, Поліморфізм, Інтерфейси:

- Використовуйте інкапсуляцію для захисту даних класів. Реалізуйте інтерфейс `Playable`, який містить методи для відтворення та призупинення мультимедійних файлів.

4. Перерахування, Вкладені класи:\*

- Створити перерахування `FileType` зі значеннями `AUDIO`, `VIDEO`, `IMAGE`, тощо. Додайте це перерахування до класу `MediaFile`. Вкладіть клас `Metadata`, який містить метадані для мультимедійного файлу (автор, рік випуску, тощо).

5. Object:

- Перевизначте методи `toString()`, `equals()` та `hashCode()` для класів, щоб забезпечити правильну роботу об'єктів при використанні у колекціях та інших структурах даних.

6. Обробка виключень:

- Реалізуйте можливість обробки виключень при роботі з файлами: наприклад, виникнення виключення, коли спробуєте відтворити файл, який не існує.

7. Узагальнююче програмування:\*

- Застосуйте узагальнюючі класи або методи для деяких операцій, наприклад, введення метаданих, що може бути використано для будь-якого типу `MediaFile`.

Цей проект дозволяє вам використовувати різні аспекти об'єктно-орієнтованого програмування та демонструє важливі концепції, такі як успадкування, інкапсуляція, поліморфізм, узагальнююче програмування та обробка виключень.

*Варіант 6*

Проект: "Електронний магазин":

1. Класи:

- Створення класів для представлення різних об'єктів у магазині, таких як Товар, Замовлення, Користувач і т.д.

2. Наслідування, абстрактні класи:

- Використання наслідування для реалізації спільних властивостей серед класів, наприклад, різних видів товарів.

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Використання інкапсуляції для приховання деяких деталей реалізації класів.

- Реалізація поліморфізму через використання інтерфейсів для об'єктів різних типів (наприклад, різні види оплати).

4. Перерахування. Вкладені класи:

- Використання перерахувань для класифікації товарів (наприклад, Категорії товарів: Електроніка, Одяг, Книги).

- Вкладені класи можна використовувати для опису додаткових деталей товару (наприклад, Технічні характеристики для електроніки).

5. Object:

- Використання методів класу Object, таких як equals(), hashCode() та toString() для правильного порівняння об'єктів, отримання хеш-кодів та отримання рядкового представлення об'єкта.

6. Обробка виключень:

- Додавання обробки виключень для сценаріїв помилок, таких як недостатній залишок на складі, неправильний формат оплати тощо.

7. Узагальнююче программування:

- Використання узагальненого класу для забезпечення гнучкості у роботі з різними типами товарів.

Такий проект дозволить вам вдосконалити розуміння та навички у багатьох аспектах об'єктно-орієнтованого програмування та Java.

*Варіант 7*

Проект: Система Магазину електроніки

1. Класи:

- Створіть класи для різних видів товарів (наприклад, телефони, ноутбуки, планшети), представляючи їх як об'єкти.

2. Наслідування, абстрактні класи:

- Використовуйте наслідування для створення загального класу "Електроніка", а також конкретних класів для кожного типу товару.

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Застосуйте інкапсуляцію та створіть інтерфейси для операцій, які можна виконувати з електронікою (наприклад, ввімкнути, вимкнути, відобразити характеристики).

- Реалізуйте ці інтерфейси у конкретних класах та використовуйте поліморфізм для їх виклику.

4. Перерахування. Вкладені класи:

- Використовуйте перерахування для визначення можливих брендів електроніки.

- Розгляньте можливість використання вкладених класів для детального опису параметрів певних товарів.

5. Object:

- Перевизначте методи `toString`, `equals` та `hashCode` у класах для ефективної роботи з об'єктами.

6. Обробка виключень:

- Додайте обробку виключень для можливих ситуацій, таких як введення некоректних даних або намагання провести операції з товаром, який відсутній на складі.

7. Узагальнююче программування:

- Використайте узагальнення для створення класів або методів, які можуть працювати з різними типами електроніки.

Цей проект дозволяє застосувати різні концепції об'єктно-орієнтованого програмування та розглянути їхнє використання в практичному контексті, що пов'язаний із створенням системи магазину електроніки.

*Варіант 8*

Проект: Компанія тваринного притулку

1. Класи:

- Створіть клас `Animal`, який включає в себе основну інформацію про тварину (ім'я, вік, вид, стан здоров'я тощо).

2. Наслідування, абстрактні класи:

- Створіть абстрактний клас `Pet`, який успадковує від класу `Animal` та містить додаткові характеристики тварин, які є домашніми улюбленцями (наприклад, тип шерсті, характер).

3. Інкапсуляція, Поліморфізм, Інтерфейси:

- Визначте інтерфейс `AdoptionService`, який має методи для прийому та віддачі тварин в адопцію.

- Створіть клас `AdoptionCenter`, який реалізує цей інтерфейс та взаємодіє з тваринами, щоб здійснювати процес адопції.

4. Перерахування, Вкладені класи:

- Визначте перерахування `AnimalType` для різних типів тварин (собака, кіт, птах тощо).

- Створіть вкладений клас `MedicalRecord` в класі `Animal`, який містить інформацію про медичні процедури та вакцинації тварини.

5. Object:

- Перевизначте методи `toString()`, `equals()`, та `hashCode()` для класів `Animal` та `MedicalRecord` для кращого представлення та порівняння об'єктів.

6. Обробка виключень:

- Реалізуйте механізм обробки виключень для ситуацій, пов'язаних із здоров'ям тварини чи неправильним введенням даних.

7. Узагальнююче програмування:

- Використовуйте узагальнені колекції для зберігання та обробки різних типів тварин.

Цей проект може слугувати базою для системи управління тваринним притулком, де можна додавати, редагувати та видаляти тварин, а також взаємодіяти з процесом їх адопції.

*Варіант 9*

Проект: Керування Медичним Центром

1. Класи:

- Створіть класи, що представляють сутності у медичному центрі, такі як Пацієнт, Лікар, Медичний Персонал, Медичне Обладнання тощо.

2. Наслідування, абстрактні класи

- Створіть ієрархію класів для лікарів, використовуючи абстрактний клас "Лікар" та конкретні підкласи для різних спеціальностей лікарів.

3. Інкапсуляція. Поліморфізм. Інтерфейси:

- Використовуйте інкапсуляцію для обмеження доступу до даних пацієнтів та лікарів. Застосуйте поліморфізм та інтерфейси для реалізації різних видів медичних послуг та процедур.

4. Перерахування. Вкладені класи

- Використовуйте перерахування для визначення статусів пацієнтів, таких як "Зареєстрований", "Прийнято на прийом" і т.д. Вкладені класи можуть бути використані для представлення додаткової інформації про аналізи або призначені ліки.

5. Object:

- Використовуйте клас Object для реалізації методів equals та hashCode для об'єктів, таких як Пацієнт та Лікар, щоб можна було використовувати їх у колекціях.

6. Обробка виключень:

- Реалізуйте механізм обробки виключень для ситуацій, таких як призначення лікарських засобів на підставі алергій пацієнта, відсутність лікаря для прийому тощо.

7. Узагальнююче програмування:

- Використовуйте узагальнені класи або методи для універсалізації деяких функцій, наприклад, для зберігання списку пацієнтів чи призначення лікарських засобів.