**Object-oriented programming (OOP).**

I. Read and memorize the following words and word combinations:

to invoke - активувати

generic - уніфікований

encapsulation – формування пакету

attribute – властивість, характерна риса

entity – 1. Суть, 2. Одиниця 3. Комплекс 4. Категорія 5. ціле

to override – 1. обходити 2. відкидати 3. відхиляти 4. нехтувати 5. ігнорувати 6. скасовувати

**Об'єктно-орієнтоване програмування (OOP)**

Об'єктно-орієнтоване програмування є наступником процесуального (структурного) програмування. Процесуальне програмування описує програми як групи багаторазових кодових одиниць (процедур), які визначають вхідні та вихідні параметри. Процедурні програми складаються з процедур, які викликають один одного.

Проблема з процедурним програмуванням полягає в тому, що багаторазове використання коду є складним і обмеженим - лише процедури можуть бути використані повторно, і важко зробити їх загальними та гнучкими. Нелегкий спосіб працювати з абстрактними структурами даних з різними реалізаціями.

Об'єктно-орієнтований підхід спирається на парадигму, що кожна програма працює з даними, що описують об'єкти (об'єкти або події) від реального життя. Наприклад: бухгалтерські програмні системи працюють з рахунками-фактурами, предметами, складами, наявністю, замовленнями на продажі тощо. Саме так виникли об'єкти. Вони описують характеристики (властивості) та поведінку (методи) таких об'єктів реального життя. Основними перевагами та цілями ООП є спрощення розробки складного програмного забезпечення та його легкість у обслуговуванні. OOP дозволяє легко повторно використовувати код, застосовуючи прості та загальноприйняті правила. Переваги парадигми ООП стають більш зрозумілими, якщо ви зрозумієте його основні принципи: *інкапсуляцію, успадкування та поліморфізм*.

**Encapsulation**

Інкапсуляція - це ідея, що атрибути сутності вкладені в цей суб'єкт. Це дає контекст атрибутам. Це також дозволяє програмісту обмежувати доступ до цих атрибутів таким чином, щоб ці атрибути модифіковані та / або використовувались лише так, як програміст має намір їх використовувати.

Нагадаємо, що об'єкт є насамперед дискретним або самодостатним. Ця характеристика є принципом інкапсуляції на роботі. Сховання - це інший термін, який іноді використовується для вираження автономної, захищеної природи об'єктів.

**Inheritance**

У структурованому програмуванні звичайно копіювати структуру, надати йому нове ім'я та додавати або змінювати атрибути, які роблять нову сутність (наприклад, запис облікового запису) відмінною від оригінального джерела. З часом цей підхід породжує велику кількість дубльованих коду, що може спричинити проблеми з обслуговуванням.

OOP вводить поняття спадщини, за допомогою чого спеціалізовані класи - без додаткового коду - можуть "копіювати" атрибути та поведінку вихідних класів, які вони спеціалізуються. Якщо деякі з цих атрибутів або поведінки потрібно змінити, ви перевизначите їх. Єдиним вихідним кодом, який ви змінюєте, це код, необхідний для створення спеціалізованих класів. Вихідний об'єкт називається батьком, а нова спеціалізація називається дитиною.

Припустимо, що ви пишете програму для роботи з людьми та хочете використовувати клас Person як основу (також називається супер класом) для нового класу, що називається Employee. Будучи дитиною Особи, працівник матиме всі атрибути класу Осіб разом з додатковими, такими як: платник податків, ідентифікаційний номер, номер працівника, зарплата. Спадкування полегшує створення нового класу Employee без необхідності копіювати весь код особи вручну.

**Polymorphism**

Поліморфізм - це поняття, що може існувати безліч різних реалізацій виконуваної одиниці, а різниця відбувається поза сценою без усвідомлення викликом.

Поліморфізм дозволяє розширювати комп'ютерні системи, створюючи нові спеціалізовані об'єкти, дозволяючи поточній частині системи взаємодіяти з новим об'єктом, не зачіпаючи конкретних властивостей нових об'єктів.

Наприклад, якщо вам потрібно було написати повідомлення на аркуші паперу, ви можете використовувати ручку, олівець, маркер або навіть олівець. Ви тільки вимагаєте, щоб використовуваний вами елемент міг вміститися у вашій руці, і він може зробити відмітку при натисканні на папір. Таким чином, дія листа допоможе вам зробити позначку на папері, а те, що маркування чи написання інструменту для використання є питанням рішення. Інший приклад - літак і космічний човник, обидва вони можуть бути названі як Flying objects. Але шлях і летить різні І.Е. існує різниця в реалізації. Але з точки зору глядача обидва об'єкти можуть літати.

**Abstraction**

Абстракція - це процес приховування всієї, крім відповідної інформації, відомостей про те, щоб зробити речі менш складними та більш ефективними для користувача. Наприклад, нам не потрібно знати, як годинник працює, щоб використовувати його, щоб сказати час. Абстракція дозволяє зосередитись на тому, що робить, замість того, як це робиться.

2. Вставте слова: переоцінка, наміри, інструменти, можливості панацеї, рішення, додатки, абстрактні причини, завдання

Є ряд важливих ... чому за останні два десятиліття об'єктно-орієнтоване програмування стало домінуючою парадигмою програмування. Об'єктно-орієнтоване програмування дуже добре від найменших до найбільш складних .... Це забезпечує форму ... що резонує з технікою, яку люди використовують для вирішення проблем у їх повсякденному житті. І для більшості домінуючих об'єктно-орієнтованих мов існує все більша кількість бібліотек, які допомагають у розвитку ... для багатьох областей. Об'єктно-орієнтоване програмування є лише останнім у довгій серії ..., які були запропоновані, щоб допомогти вирішити «програмну кризу». На думку, криза програмного просто означає, що наше уявлення та завдання, які ми хочемо вирішити за допомогою комп'ютерів, майже завжди перевершують нашу ... Але, хоча об'єктно-орієнтовані методи сприяють створенню складних програмних систем, важливо пам'ятати, що OOP не є .... Програмування комп'ютера є однією з найскладніших завдань, що колись виконували люди; для того, щоб стати досвідченим програмуванням, потрібні таланти, творчість, інтелект, логіка, вміння створювати абстракції та використовувати їх, а також досвід, навіть якщо є найкращим .... Об'єктно-орієнтоване програмування є новим способом мислення про те, що означає обчислити, про те, як ми можемо структурувати інформацію та спілкуватися з нашими ... як між собою, так і з машиною. Щоб стати володарем об'єктно-орієнтованих прийомів, потрібно повне ... традиційного розроблення програмного забезпечення.

3. Прочитайте та перекладіть слова, що означають заперечення:

незвичайний; невикористаний; неуточнений; необмежена; незадовільний; нечасто; нерозподілений; незалежний; непрямий; не відрізняється; неможливо; недолік; диз'юнкція; декодувати; незалежно; марно

4. Читайте та перекладіть дієслова, що означає повторення:

retype; перекомпілювати; переробляти; повторне використання; повторно підкреслити; перемістити; перечитувати; переписати; повторно; змінити порядок; знову з'являється; замінити; перезапустити; перемотати; огляд; повернутися

5. Перекладіть пропозиції, звертаючи увагу на засоби модальності:

1. Перш ніж обговорювати самі моделі, необхідно вивчити чинники, які можуть вплинути на результати. 2. Щоб скоротити час, необхідний для виконання інструкцій, бажано виконувати якомога більше операцій паралельно. 3. Передбачається надання компенсацій для всіх початкових граничних умов, які повинні бути застосовані. 4. Перш за все потрібно визначити потреби людей та конвенції, а потім максимально ефективно конвертувати їх в програми. 5. Іноді система мікрокомп'ютерів повинна використовуватись як комп'ютер загального призначення. 6. Інженери повинні мати справу з розвитком існуючих систем, а також проектуванням нових систем. 7. Правильно розроблені інформаційні системи можуть розглядатися як чорні ящики. 8. Комп'ютер дозволив механізувати більшу частину інформаційного обміну та обробки, які складають нервову систему нашого суспільства. 9. Архітектура повинна забезпечувати адекватну гнучкість, щоб підтримувати зростаючу тенденцію розподілених систем. 10. Розвиваються інформаційні системи та постійно додаються нові. Архітектура повинна підтримувати таке зростання.

 6. Прочитайте текст та перекладіть його зі словником.

 Що таке UML?

UML є графічною мовою для візуалізації, вказування, конструювання та документування артефактів програмно-інтенсивної системи. UML дає вам стандартний спосіб написати схеми системи, яка охоплює концептуальні речі, такі як бізнес-процеси та системні функції, а також конкретні речі, такі як класи, написані на певній мові програмування, схеми баз даних та багаторазові компоненти програмного забезпечення.

**Model**

Модель є спрощення реальності.

Модель містить креслення системи.

Модель може бути структурною, що підкреслює організацію системи, або може бути поведінковою, підкреслюючи динаміку системи.

Ми будуємо моделі, щоб ми могли краще зрозуміти систему, яку ми розробляємо. Завдяки моделюванню ми досягаємо чотирьох цілей. 1. Моделі допомагають нам візуалізувати систему так, як вона є або як ми цього хочемо.

2. Моделі дозволяють вказувати структуру або поведінку системи.

3. Моделі дають нам шаблон, який допомагає нам у побудові системи.

4. Моделі документують прийняті нами рішення.

Ми будуємо моделі складних систем, тому що ми не можемо осягнути таку систему в цілому. **Principles of Modeling**

Існує чотири основні принципи моделі

1. Вибір того, які моделі створюють, глибоко впливають на те, як атакується проблема та як формується рішення.

2. Кожна модель може бути виражена на різних рівнях точності.

 3. Кращі моделі пов'язані з реальністю.

4. Жодної моделі недостатньо.

Кожну нетривіальну систему найкраще підійти через невеликий набір майже незалежних моделей.

**Об'єктно-орієнтоване моделювання**

 У програмному забезпеченні існує кілька способів підходу до моделі. Два найбільш поширених шляху є

1. Алгоритмічна перспектива

2. Об'єктно-орієнтована перспектива

Алгоритмічна перспектива Традиційний підхід до розробки програмного забезпечення має алгоритмічну перспективу. У такому підході основним будівельним блоком всього програмного забезпечення є процедура або функція. Ця думка змушує розробників зосередитися на питаннях контролю та розкладання більших алгоритмів на менші. Оскільки вимоги змінюються і система зростає, системи, побудовані з алгоритмічним фокусом, виявилися дуже важкими для підтримки.

**Об'єктно-орієнтована перспектива**

Сучасний погляд на розробку програмного забезпечення має об'єктно-орієнтовану перспективу. У цьому підході головним будівельним блоком всіх програмних систем є об'єкт або клас. Клас - це опис набору загальних об'єктів. Кожен об'єкт має ідентичність, стан і поведінку. Об'єктно-орієнтований розвиток забезпечує концептуальну основу для збірки систем з компонентів, що використовують таку технологію, як Java Beans або COM +.