Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму №2

з дисципліни

«Архітектура програмного забезпечення»

Тема: «Створення діаграми класів»

|  |  |
| --- | --- |
| Прийняв: | Виконав: |
| Кузнєцов  Олександр  Вікторович | студент 3-го курсу  гр. ІП-51 ФІОТ  Зарічковий Олександра Антаолійович |

Київ – 2017

**ЗМІСТ:**

1. Мета РОБОТИ 3
2. Постановка задачі 4
3. Результати та пояснення 5
4. ЗАПИТАННЯ ТА ВІДПОВІДІ 8
5. Висновок 9

# Мета РОБОТИ

Набути навичок побудови діаграми класів, створення пакетів і групування класів в пакети.

# Постановка задачі

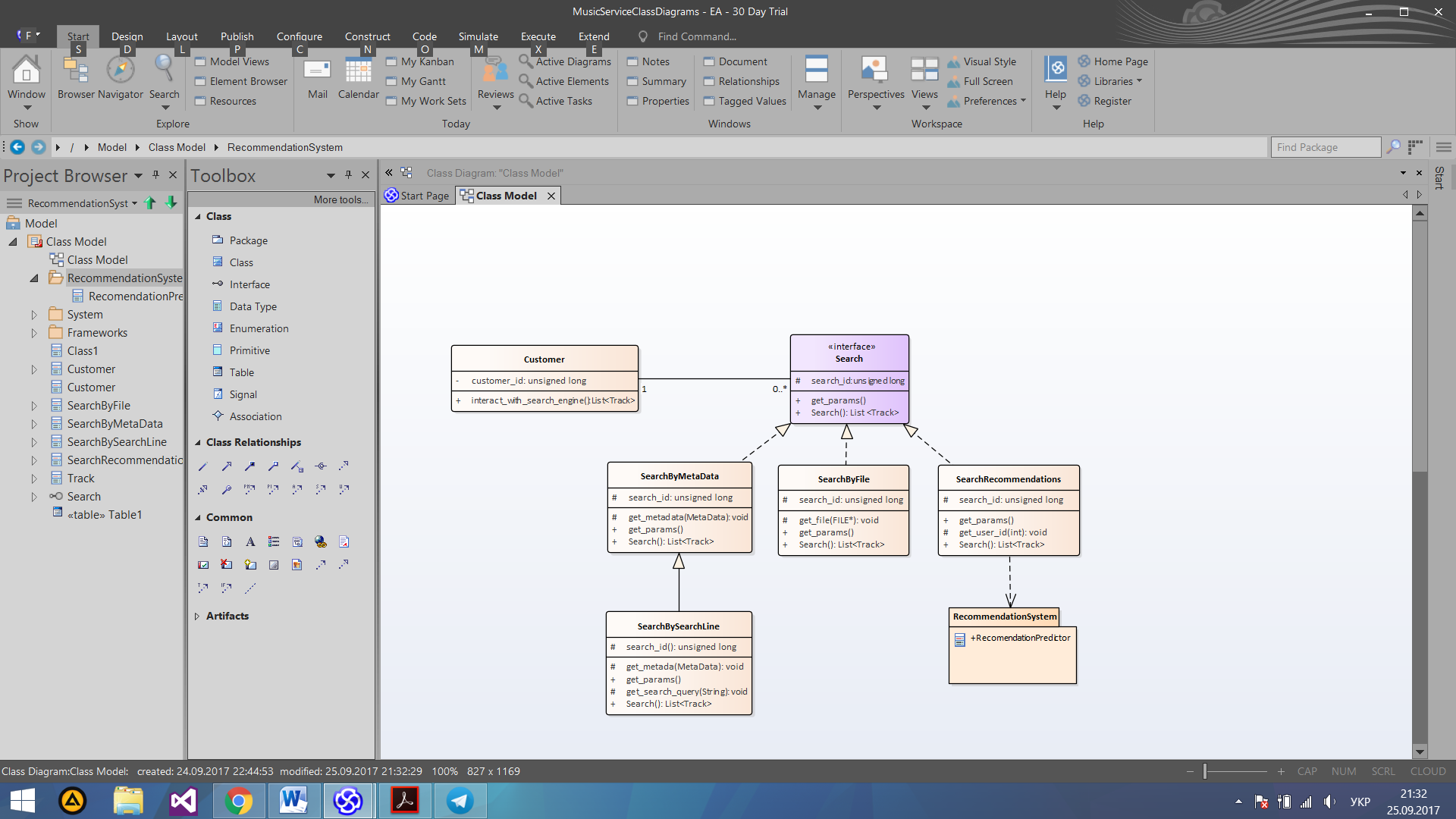
* Побудувати діаграму класів для одного із сценаріїв діаграми прецедентів, створеній в попередній лабораторній роботі. Для кожного класу необхідно задати атрибути і операції. Кожен клас повинен бути детально задокументований - необхідно задати текстовий опис цього класу, опис його атрибутів і операцій
* Створити пакети для групування класів, створених в пункті 1
* Згрупувати класи з пункту 1 пакети
* Для кожного пакета побудувати свою діаграму класів
* Розробити головну діаграму класів

# Результати та пояснення

**Прецедент:** пошук музикального трека.

**Сценарій:** користувач обирає функцію «Пошук музикальних треків». Перед користувачем постає тип пошуку «Пошук по метеданим», «Пошук по файлу», «Пошук рекомендованих». В залежності від обраного типу пошуку відбувається реалізація пошуку в базі даних.

**Діаграма класів:**



**Кількісний аналіз діаграми класів:**

Маємо 5 класів, 1 інтерфейс, 1 пакет, 1 відношень типу «Ассоціація», 1 відношення типу «Залежність», 3 відношення типу «Реалізація» та 1 відношення типу «Узагальнення».

S*obj* = 5 \* 5 + 1 \* 4 + 1 \* 5 = 34

S*lnk* = 1 \* 1 + 1 \* 2 + 3 \* 2 + 1 \* 3= 12

T*obj* = 3

T*lnk* = 4

O*obj* = 7

S = (34 + 12)/(1 + 7 + (4 + 7)^0.5) = 46/11.32 = 4

**Скорочений опис класів та відношень:**

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | | | |
| Атрибути | - Customer\_id | Unsigned long | Ідентифікаційний номер користувача |
| Функції | + List <Track> Interact\_with\_search\_engine () | | Асинхронний процес взаємодії з пошуковим двигуном |

|  |  |  |  |
| --- | --- | --- | --- |
| *Search* | | | |
| Атрибути | - Search\_id | Unsigned long | Ідентифікаційний номер пошуку |
| Функції | + Void get\_params() | | Отримання класом параметрів необхідних для пошуку |
| + List <Track> Search() | | Пошук музикальних треків за заданими параметрами |

|  |  |  |  |
| --- | --- | --- | --- |
| SearchByMetaData | | | |
| Атрибути | - Search\_id | Unsigned long | Ідентифікаційний номер пошуку |
| Функції | + Void get\_params() | | Отримання класом параметрів необхідних для пошуку |
| + List <Track> Search() | | Пошук музикальних треків за заданими параметрами |
| # Void get\_metada() | | Отримання класом метеданих для пошуку |

|  |  |  |  |
| --- | --- | --- | --- |
| SearchBySearchLine | | | |
| Атрибути | - Search\_id | Unsigned long | Ідентифікаційний номер пошуку |
| Функції | + Void get\_params() | | Отримання класом параметрів необхідних для пошуку |
| + List <Track> Search() | | Пошук музикальних треків за заданими параметрами |
| # Void get\_metada(MetaData) | | Отримання класом метаданих для пошуку |
| # Void get\_serch\_query(String) | | Отримання класом метаданих шляхом вводу в запиту в пошуковій строчці |

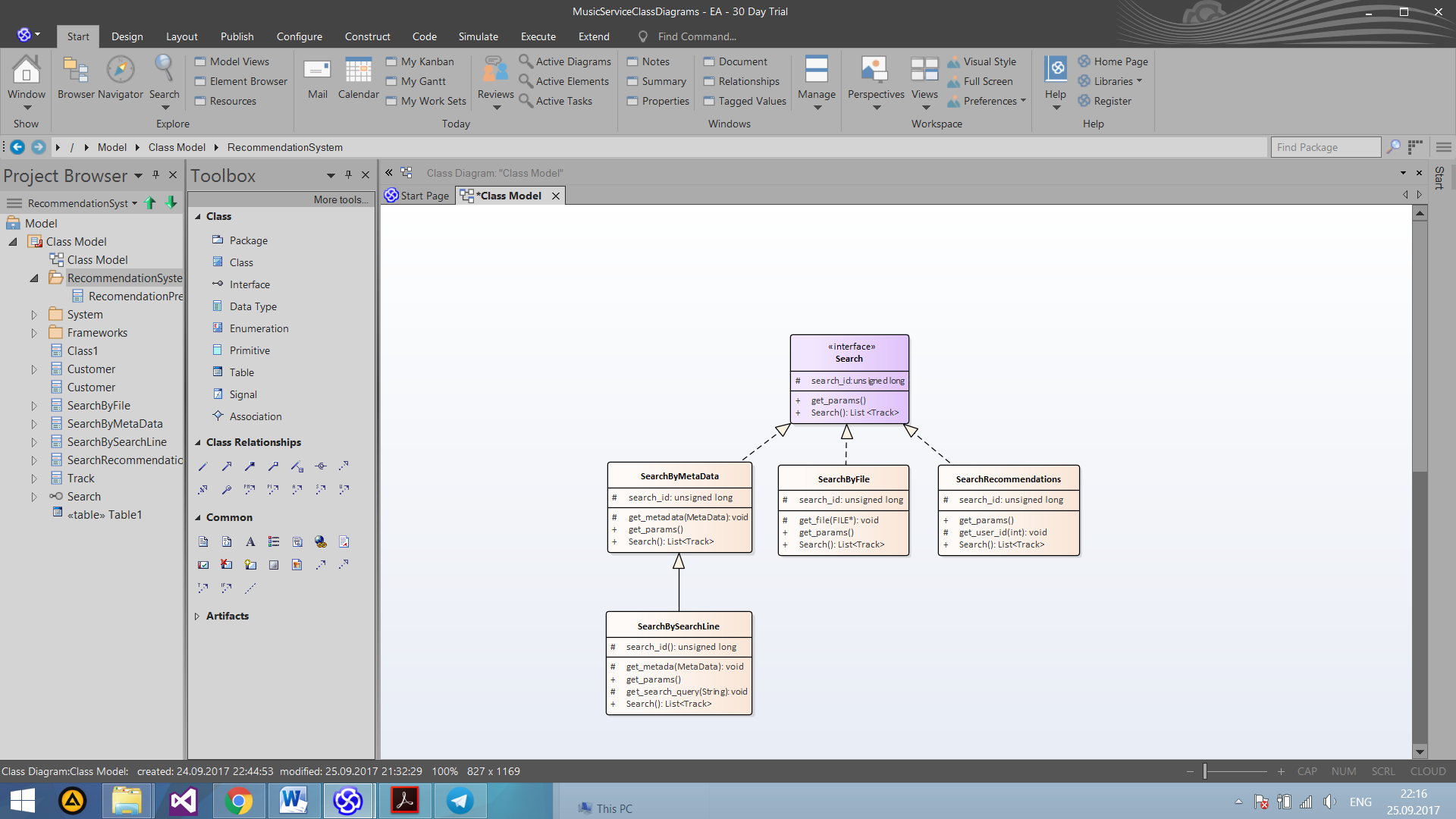
|  |  |  |  |
| --- | --- | --- | --- |
| SearchByFile | | | |
| Атрибути | - Search\_id | Unsigned long | Ідентифікаційний номер пошуку |
| Функції | + Void get\_params() | | Отримання класом параметрів необхідних для пошуку |
| + List <Track> Search() | | Пошук музикальних треків за заданими параметрами |
| # Void get\_user\_id(FILE\*) | | Отримання класом файлу для пошуку |

|  |  |  |  |
| --- | --- | --- | --- |
| SearchRecommendations | | | |
| Атрибути | - Search\_id | Unsigned long | Ідентифікаційний номер пошуку |
| Функції | + Void get\_params() | | Отримання класом параметрів необхідних для пошуку |
| + List <Track> Search() | | Пошук музикальних треків за заданими параметрами |
| # Void get\_user\_id(unsigned long) | | Отримання класом ідентифікатора користувача для пошуку |

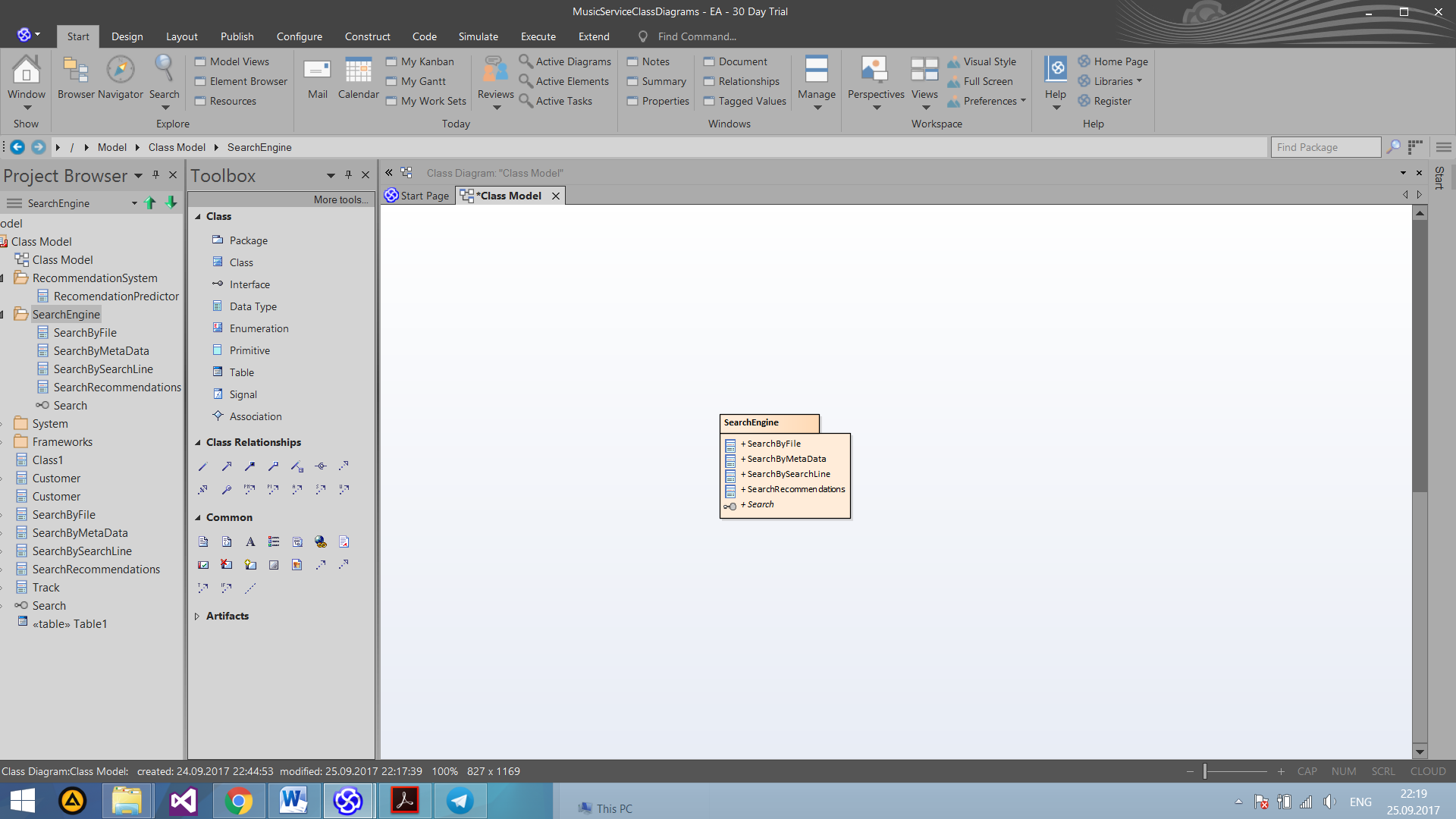
1. Класи *SerchByMetaData, SearchByFile, SearchRecommendations* та інтерфейс *Search* — відношення резалізації, оскільки описані класи реалізують весь запрошений функціонал інтерфейсом.
2. Класи *SerchByMetaData* та *SearchBySearchLine* — відношення узагльнення, оскільки *SearchBySearchLine* являється спеціалізованим узагальнення класу *SerchByMetaData*.
3. Клас *Customer* та інтерфейс *Search* — відношення асоціації, оскільки обидва знають про існування одне одного. Будь-який користувач може мати довільну кількість запитів, але кожен запит належить певному одному користувачеві. Тому кратність зв’язку з боку класа *Customer* — 1, з боку інтерфейса *Search* — 0..\*.
4. Клас *SearchRecommendations* та пакет *RecommendationSystem* — залежність, оскільки частина функціоналу класу *SearchRecommendations* від інтерфейсу та реалізації класів в пакеті *RecommendationSystem*.

**Пакети:** усі створені класи є сутностями, тому створимо для них один пакет *SearchEngine*.

**Діаграма пакету** *SearchEngine***:**



**Головна діаграма пакетів:**



# ЗАПИТАННЯ ТА ВІДПОВІДІ

1. **Що таке пряме та зворотнє проектування?**

Пряме проектування (forward engineering) - це процес трансформації моделі в код з відображенням на мову реалізації. В результаті прямого проектування відбувається втрата інформація, оскільки моделі, описані на UML, семантично багатшими, ніж будь-який сучасний об'ектно-орієнтованний підхід в програмуванні.

Зворотне проектування (reverse engineering) - це процес трансформації коду в модель. Зворотне проектування породжує надлишок інформації, частина якої представлена на більш низькому рівні деталізації, ніж потрібно для побудови зручною моделі.

1. **Що таке пакет та його основне застосування в стандарті UML?**

Пакет - це спосіб організації елементів моделі в блоки, якими можна керувати як єдиним цілим. Можна керувати видимістю елементів пакету, так що деякі будуть видні користувачеві, а інші - приховані. Крім того, за допомогою пакетів зображуються різні уявлення архітектури системи.

1. **Що таке узагальнення (генералізація)?**

Узагальнення - це відношення між більш загальної сутністю, званої суперкласом, і її конкретним втіленням, званим підкласом. Іноді узагальнення називають відносинами типу "є", маючи на увазі, що одні сутності (наприклад, коло, квадрат, трикутник) є втіленням більш загальної сутності (наприклад, класу "геометрична фігура"). При цьому всі атрибути і операції суперкласу незалежно від модифікаторів видимості входять до складу підкласу.

1. **З яких етапів складається створення діаграми класів?**

Створення діаграми класів має наступні етапи:

• вибір прецеденту

• виділення класів

• виділення атрибутів та функцій класів

• встановлення відношень між класами

• групування класів за пакетами

1. **Чому схожі за призначенням класи краще групувати в пакет?**

Пакети призначені для групування елементів в групи за визначеними критеріями. В найпростішому випадку класи можна групувати за їх стереотипами. Якщо певні класи в моделі є просто сутностями, а інші відповідають за управління між ними, доцільно згрупувати їх у два пакети: *Entities*(класи-сутності) *і Control* (управляючі класи).

1. **Для чого використовується головна діаграма пакетів?**

Головна діаграма пакетів наочно відображає групування всіх класів моделі за пакетами, відображаючи вміст кожного пакета на одній діаграмі. Це дозволяє отримати уявлення про основні сутності системи, що проектується, їх взаємовідносини, а також відносини окремих груп між собою, без необхідності занурюватись у деталі внутрішньої будови кожної сутності.

# Висновок

Отже, я отримав навички створення класів засобами UML-діаграм, навички проектування класів, їх детальної документації. Також я ознайомився з принципами групування класів за пакетами та створенням головної діаграми пакетів.