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 Олександр Антаолійович |

Київ – 2017

**ЗМІСТ:**

1. Мета РОБОТИ 3
2. Постановка задачі 4
3. Результати та пояснення 5
4. ЗАПИТАННЯ ТА ВІДПОВІДІ 10
5. Висновок 12

# Мета РОБОТИ

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

# Постановка задачі

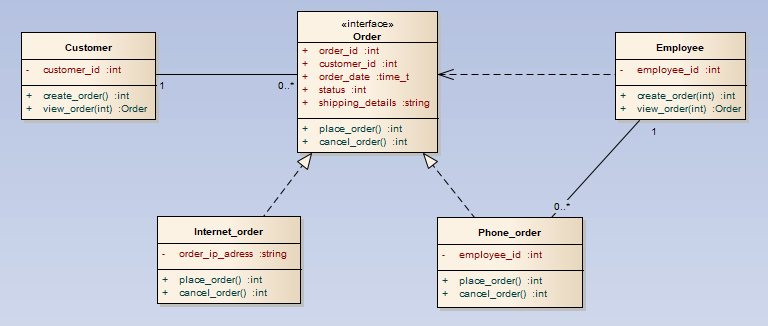
* Побудувати діаграму класів для одного із сценаріїв діаграми прецедентів, створеній в попередній лабораторній роботі. Для кожного класу необхідно задати атрибути і операції. Кожен клас повинен бути детально задокументований - необхідно задати текстовий опис цього класу, опис його атрибутів і операцій
* 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:** замовлення в інтернет-магазині.

**Сценарій:** користувач обирає функцію «Замовлення товару». Перед користувачем постає вибір типу замовлення «Замовлення по телефону», «Замовлення через сайт». В залежності від обраного типу замовлення інформація додається відповідним чином.

**Діаграма класів:**



**Кількісний аналіз діаграми класів:**

Маємо 4 класів, 1 інтерфейс, 1 пакети, 2 відношень типу «Ассоціація», 1 відношення типу «Залежність» та 2 відношення типу «Реалізація».

S*obj* = 4 \* 5 + 1 \* 4 + 1 \* 5 = 29

S*lnk* = 2 \* 1 + 1 \* 2 + 2 \* 2 = 8

T*obj* = 3

T*lnk* = 3

O*obj* = 5

S = (29 + 8)/(1 + 5 + (3 + 5)^0.5) = 4.19

**Скорочений опис класів та відношень:**

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | | | |
| Атрибути | - Customer\_id | int | Ідентифікаційний номер користувача |
| Функції | + int create\_order () | | Процес створення замовлення |
| + int view­\_order (int order\_id) | | Отримує данні про замовлення |

|  |  |  |  |
| --- | --- | --- | --- |
| Employee | | | |
| Атрибути | - Employee\_id | int | Ідентифікаційний номер робітника |
| Функції | + int create\_order (int customer\_id) | | Створює замовлення клієнту |
| + int view­\_order (int order\_id) | | Отримує данні про замовлення |

|  |  |  |  |
| --- | --- | --- | --- |
| Order | | | |
| Атрибути | + order\_id | int | Ідентифікаційний номер замовлення |
| + customer\_id | int | Ідентифікатор користувача |
| + order\_date | time\_t | Дата замовлення |
| + status | int | Код статусу замовлення |
| + shipping\_details | string | Данні доставки |
| Функції | + int place\_order() | | Вказує що оформлення замовлення завершено |
| + int cancel\_order() | | Відміняє замовленння |

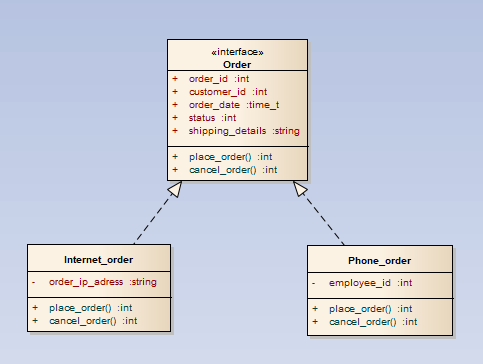
|  |  |  |  |
| --- | --- | --- | --- |
| Internet\_order | | | |
| Атрибути | - order\_ip\_adress | string | IP-адресса з якої оформлено замовлення |
| Функції | + int place\_order() | | Вказує що оформлення замовлення завершено |
| + int cancel\_order() | | Відміняє замовленння |

|  |  |  |  |
| --- | --- | --- | --- |
| Phone\_order | | | |
| Атрибути | - employee\_id | int | Ідентифікаційний номер робітника створившого замовлення |
| Функції | + int place\_order() | | Вказує що оформлення замовлення завершено |
| + int cancel\_order() | | Відміняє замовленння |

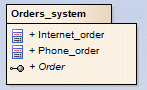
1. Класи *Internet\_order, Phone\_order* та інтерфейс *Order* — відношення резалізації, оскільки описані класи реалізують функціональність інтерфейсу.
2. Класи *Employee* та *Order* — відношення залежності , оскільки *Employee* має метод що залежить від Order.
3. Клас *Customer* та інтерфейс *Order* — відношення асоціації, оскільки обидва знають про існування одне одного. Будь-який користувач може мати довільну кількість замовлень, але кожне замовлення належить певному одному користувачеві. Тому кратність зв’язку з боку класу *Customer* — 1, а з боку інтерфейса *Order* — 0..\*.
4. Класи *Employee* та *PhoneOrder* — відношення асоціації, оскільки обидва знають про існування одне одного. Будь-який робітник може мати довільну кількість замовлень, але кожне замовлення належить певному одному робітнику. Тому кратність зв’язку з боку класу *Employee* — 1, а з боку класу *PhoneOrder* — 0..\*.

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

**Діаграма пакету** *Orders\_system***:**



**Головна діаграма пакетів:**



# ЗАПИТАННЯ ТА ВІДПОВІДІ

1. **Що такое відношення** **“Один до багатьох”**

Відношення “Один до багатьох” – це таке відношення між классами коли одному з класів може відповідати біль ніж один клас, але іншому класу завжди відповідає єдиний. При такому відношенні на діаграммі на кінцях відношення біля того класу якому може відповідати багато класів ставиться число один, а біля іншого класу ставиться 0..\* або 1..\* в залежності від того чи може перший клас не відноситись ні до одного другого.

1. **Чому UML діаграми розбивають на пакети?**

Пакет - це спосіб організації елементів моделі в блоки, якими можна керувати як єдиним цілим. Також можна приховати ті елементи пакету що не повинні використовуватись в інших пакетах. Таким чином розбиття діаграми на пакети спрощує діаграму та гарантує що приховані елементи не використовуються там де вони не повинні, тобто задовольняє один з основних механізмів об’єктно орієнтовного програмування - Інкапсуляцію

1. **Що таке реалізація (realization)?**

Реалізація – це відношення при якому клас повністю реалізує методи інтерфейсу, тобто являється самостійним класом що може виконувати в тому числі функції інтерфейсу який він реалізує, а також може мати інші функції, наприклад реалізацію іншого інтерфейму. Ключова відмінність реалізації від генералізації (generalization) в тому що при генералізації один з классів розширює можливості уже реалізованого классу а не реалізує інтерфейс.

1. **Які є типи відношень?**

Існують наступні види відношень:

* Залежність (dependency)
* Асоціація (association)
* Агрегація (aggregation)
* Композиція (composition)
* Генералізація (generalization)
* Реалізація (realization)

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

Головна діаграма пакетів показує усі пакети усіх класів. Таким чином головна діаграма пакетів демонструє усі інші діаграми. По ній ви можете зрозуміти яким чином відносяться пакети, та загальну структуру проекту. Якщо б її не було то для цих цілей потрібно було б продивитись усі пакети, навіть якщо вас не цікавить їх внутрішня будова.

1. **Що таке пакет?**

Пакет – це механізм організації різних елементів моделі в єдину сутність за принципом декомпозиції складної схеми. Пакет включає в себе різні елементи та може включати інші пакети. В пакеті можна керувати видимістю різних елементів, щоб заборонити їх використання ззовні пакету, таким чином реалізується інкапсуляція. Крім того короткий опис пакету дає уявлення про загальне призначення усіх його елементів.

# Висновок

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