Змі

Вступ
1 Огляд MVC5
1.1 Загальний огляд5
1.2 Історія 6
1.3 Призначення
1.4 Концепція
1.5 Найчастіші помилки9
1.6 Реалізація10
1.6.1 Java
2 Проектування програмного забезпечення
2.1 Прецеденти
2.1.1 Авторизація у системі
2.1.2 Створення нового акаунту
2.1.3 Видалення акаунту12
2.1.4 Редагування акаунту12
2.1.5 Перегляд розкладу групи
2.1.6 Перегляд історії активності користувачів
2.1.7 Редагування розкладу групи
2.1.8 Вихід із системи
2.1.9 Перегляд посібника користувача
2.2 Сценарії використання програми
2.2.1 Отримання інформації про профілі
2.2.2 Створення нової категорії у БД
2.2.3 Налаштування програми
2.2.4 Отримання загальної інформації про програму
2.3 Діаграма граничних класів
2.4 Проектування графічного інтерфейсу користувача
2.5 Таблиця відповідності елементів інтерфейсу
3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ21
3.1 Структура програмного додатку

3.2 Специфікація класів програмного додатка	21
3.2.1 Package com.Oyster.app.model	21
3.2.2 Package com.Oyster.config	37
3.2.3 Package com.Oyster.core.controller	43
3.2.4 Package com.Oyster.core.controller.command	47
3.2.5 Package com.Oyster.dao	51
3.2.6 com.Oyster.ui	56
3.3 Інструкція для користувача	63
4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ	65
ВИСНОВКИ	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТОК А. ДІАГРАМИ КЛАСІВ ДОДАТКУ	72
ДОДАТОК Б. ВИХІДНИЙ КОД ДОДАТКУ	76

ВСТУП

Курсова робота спрямована на закріпення знань та отримання навичок із розробки програмних додатків на Java із використанням бібліотеки Swing. У курсовій роботі я розроблю програмний додаток згідно вимог MVC для управління навчальним процесом в університеті (реалізувати функціонал для адміністратора).

У роботі розгляну основні принципи MVC, принципи побудови графічного інтерфейсу користувача, основні елементи інтерфейсу, організацію обробки подій, систему відлову помилок і організацію багатопоточності.

Робота міститиме повну документацію, а також програмний код проекту та UML діаграму класів.

1 ОГЛЯД МУС

1.1 Загальний огляд

Модель-вид-контролер (або **Модель-представлення-контролер** англ. *Model-view-controller*, **MVC**) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

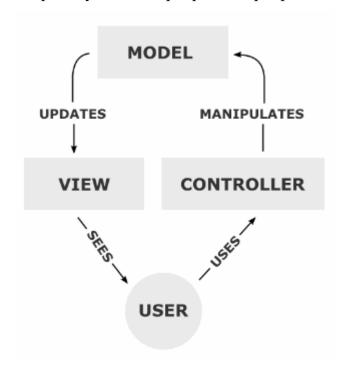


Рис 1.1 Концепція Model-View-Controller

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача(вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Дана схема проектування часто використовується для побудови архітектурного каркасу, коли переходять від теорії до реалізації в конкретній предметній області.

1.2 Історія

Концепція MVC була описана у 1979 році Трюгве Реенскауг (англ. Trygve Reenskaug), який тоді працював над мовою програмування Smalltalk в Xerox PARC. Оригінальна реалізація описана в статті «Applications Programming in Smalltalk-80: How to use Model-View-Controller». Потім Джим Алтоф з командою розробників реалізували версію MVC для бібліотеки класів Smalltalk-80.

В оригінальній концепції була описана сама ідея і роль кожного із елементів: моделі, представлення і контролера. Та зв'язки між ними були описані без конкретизації. Крім того, розрізняли дві основні модифікації:

- 1. Пасивна модель модель не має ніяких способів впливу на представлення чи контролер і використовується в якості джерела даних для відображення. Всі зміни моделі відстежуються контролером і він же відповідає за перемальовування представлення, якщо це необхідно. Така модель частіше всього використовується в структурному програмуванні, оскільки у цьому випадку модель являє собою просто структуру даних, без методів їх обробки.
- 2. Активна модель модель сповіщає представлення про те, що в ній виникли зміни, а представлення, які зацікавленні в сповіщенні, підписуються на ці сповіщення. Це дозволяє зберегти незалежність моделі як від контролера, так і від представлення.

Класичною реалізацією концепції MVC вважають версію з активною моделлю.

У даній роботі використаємо класичну модель (Активну), оскільки вона дозволяє значно краще зберегти незалежність моделі від контролера та представлення, а також спростить реалізацію та забезпечить гнучкість системи.

Із розвитком Об'єктно-орієнтованого програмування і поняття про шаблони проєктування був створений ряд модифікацій концепції МVС, які при реалізації у різних авторів можуть відрізнятися від оригінальної. Так, наприклад, Еріан Вермі в 2004 році писав приклад узагальненого МVС.

1.3 Призначення

Основна мета використання цієї концепції у розділенні бізнес-логіки (моделі) від її візуалізації (представлення, вигляду). За рахунок такого розмежування підвищується можливість повторного використання. Найбільш корисне застосування даної теорії у тих випадках, коли користувач повинен бачити ті ж самі дані одночасно у різних контекстах та/або з різних точок зору. Виконуються наступні задачі:

- 1. До однієї моделі можна приєднати декілька видів, при цьому на чіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми.
- 2. Не чіпаючи реалізацію вигляду, можна змінити реакції на дії користувача(натиснення мишкою на копку, введення даних), для цього достатньо використовувати інший контролер.
- 3. Ряд розробників спеціалізуються лише в одній із областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо досягнути того, що програмісти, які займаються розробкою бізнес-логіки(моделі), взагалі не будуть обізнані в тому, яке представлення використовуватиметься.

1.4 Концепція

Концепція MVC дозволяє розділити дані, представлення і обробку дій користувача на три окремих компоненти:

- 1. Модель(англ. Model) Модель представляє інформацію: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, про те, як ці знання можна візуалізувати.
- 2. Представлення, вигляд (англ. View). Відповідає за відображення інформації (візуалізацію). Часто в якості представлення виступає форма (вікно) з графічними елементами.
- 3. Контролер (англ. Controller). Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції, керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель.

Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

Важливо зазначити, що як представлення, так і контролер залежать від моделі. Однак модель не залежить ні від представлення, ні від контролера. Тим самим досягається призначення такого розмежування: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати декілька різних представлень для однієї моделі

Для реалізації схеми Model-View-Controller використовується достатньо велике число шаблонів проектування(в залежності від складності архітектурного рішення), основні з яких стратегія, композит, спостерігач.

Найбільш типова реалізація відділяє вигляд від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій (підписка/сповіщення). При кожній зміні внутрішніх даних в моделі вона сповіщає всі залежні від неї представлення і представлення обновлюється. Для цього використовують шаблон спостерігач. При обробці реакції користувача вигляд обирає, в залежності від потрібної реакції, потрібний контролер, який забезпечує той чи інший зв'язок з моделлю. Для цього використовується шаблон стратегія, або замість цього може бути модифікація із використанням шаблону команда. А для можливості однотипної роботи з підоб'єктами

складно-скомпонованого ієрархічного виду може використовуватись шаблон композит. Крім того, можуть використовуватись і інші шаблони проектування, наприклад, фабричний метод, який дозволить задати за замовчуванням тип контролера для відповідного виду.

1.5 Найчастіші помилки

Програмісти-початківці (особливо в веб-програмуванні, де абревіатура МVС стала популярною) дуже часто трактують архітектурну модель МVС як пасивну модель МVС. У цьому випадку модель виступає виключно сукупністю функцій для доступу до даних, а контролер містить бізнес логіку. В результаті код моделі по факту являється засобом для отримання даних із СУБД, а контролер являє собою типовий модуль, наповнений бізнес-логікою, або скрипт у термінології веб-програміста. В результаті такого розуміння МVС розробники стали писати код, який Ра́draic Brady, відомий у колах спільноти Zend Framework, охарактеризував як ТТПК – «Товсті тупі потворні контролери» (Fat Stupid Ugly Controllers)

Середньостатистичний ТТПК отримував дані із БД (використоруючи рівень абстракції бази даних, враховуючи, що ще модель) або маніпулював, перевіряв, записував, а також передавав дані у вигляд. Такий підхід став дуже популярним тому, що використання таких контролерів схоже на класичну практику використання окремого рһр-файлу для кожної сторінки додатку.

Але в об'єктно-орієнтованому програмуванні використовується активна модель МVС, де модель — це не тільки сукупність коду доступу до даних і СУБД, але і вся бізнес логіка. Варто зазначити про можливості моделі інкапсулювати в собі інші моделі. В свою чергу контролери являють собою лише елементи системи, в безпосередні обов'язки яких входить прийом даних із запиту і передача їх іншим елементам системи. Тільки у цьому випадку контролер стає «тонким» і виконує виключно функції зв'язкової ланки(glue layer) між певними компонентами системи.

1.6 Реалізація

Концепція MVC вперше застосувалася при проектуванні мови програмування Smalltalk як модель для інтерфейсу користувача . Також в область застосування концепції входить реалізація каркаса Документ-Вид (Document-View) в рамках бібліотеки MFC для мови Visual C++ . У сучасних технологіях концепція MVC представлена схемою JSP Model 1/2 для динамічної обробки Web-змісту на основі Java Server Pages (JSP).

1.6.1 Java

У мові програмування Java концепція MVC підтримується на рівні стандартних класів-бібліотек. В результаті використання парадигми MVC програміст отримує в своє розпорядження могутню структуру об'єктів-компонентів, функції яких чітко розмежовані, що гарантує надійність і розширюваність системи, що розробляється.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Прецеденти

2.1.1 Авторизація у системі

Адміністратор може увійти до системи, заповнивши поля "Логін" та "Пароль" та натиснувши кнопку "Вхід" (Рис. 2.1).

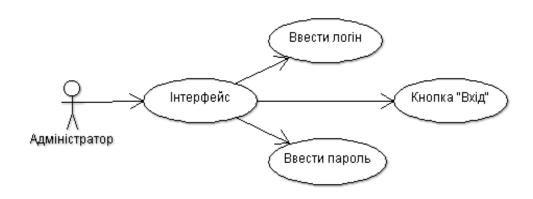


Рис. 2.1 Діаграма прецеденту операції входу до системи

2.1.2 Створення нового акаунту

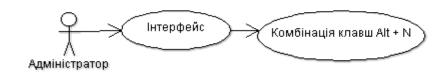
Адміністратор може додати нови акаунт, натиснувши кнопку "Додати акаунт у вкладці "Профілі" (Рис. 2.2.а) або у меню "Файл", натиснувши підпункт "Додати акаунт" (Рис. 2.2.б), або із допомогою комбінації клавіш Alt + N (Рис. 2.2.в)



а) створення акаунту через вкладку "Профілі"



б) створення акаунту чарез головне меню



в) створення акаунту із допомогою комбінації клавіш Alt + N

Рис. 2.2 Діаграма прецедентів операції створення акаунту

2.1.3 Видалення акаунту

Адміністратор може видалити акаунт, вибравши його та натиснувши кнопку "Видалити" (Рис. 2.3.a)

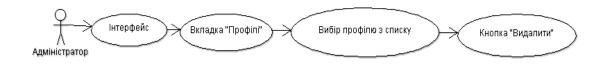


Рис. 2.3 Діаграма прецеденту операції видалення акаунту

2.1.4 Редагування акаунту

Адміністратор може редагувати акаунт, вибравши його, змінивши поля та натиснувши кнопку "Зберегти" (Рис. 2.4)

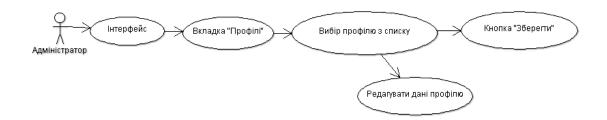


Рис. 2.4 Діаграма прецеденту операції редагування акаунту

2.1.5 Перегляд розкладу групи

Адміністратор може переглянути розклад групи, перейшовши у вкладку "Розклад", вибравши групу та предмет із списку (Рис. 2.5).

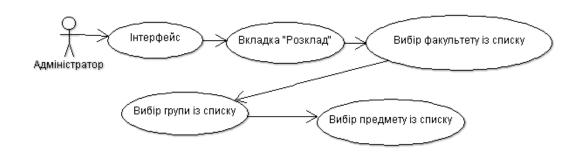


Рис. 2.5 Діаграма прецеденту перегляду розкладу групи

2.1.6 Перегляд історії активності користувачів

Адміністратор може переглянути історію активності всіх користувачі, перейшовши у вкладку "Історія" вибравши пункт "Повна" (Рис. 2.6)



Рис. 2.6 Діаграма прецеденту операції перегляду історії

2.1.7 Редагування розкладу групи

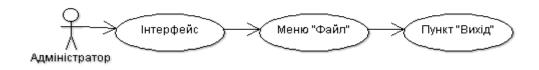
Адміністратор може редагувати розклад групи, здійснивши прецедент перегляду розкладу групи (Рис 2.5), редагувавши поля розкладу та натиснувиши кнопку "Зберегти" (Рис. 2.7).



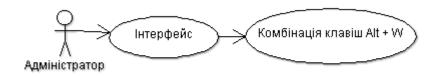
Рис. 2.7 Діаграма прецеденту операції редагування розкладу

2.1.8 Вихід із системи

Адміністратор може вийти із системи, натиснувши пункт "Вихід" у меню "Файл" (Рис. 2.8.a, B) або із допомогою комбінації клавіш Alt + W (Рис. 2.8.6).



а) вихід із системи через головне меню



б) вихід із системи або із допомогою комбінації клавіш Alt + W



в) вихід із системи із допомогою комбінації клавіш Alt + F та пункту меню

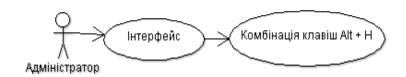
Рис. 2.8 Діаграма прецедентів операції виходу із системи

2.1.9 Перегляд посібника користувача

Адміністратор може переглянути посібник користувача, натиснувши пункт "Посібник користувача" у меню "Допомога" (Рис. 2.9.а) або із допомогою комбінації клавіш Alt + H (Рис. 2.9.б).



а) перегляд посібника користувача із допомогою головного меню



б) перегляд посібника користувача із допомогою комбінації клавіш Alt + H Рис. 2.9 Діаграма прецедентів операції перегляду посібника користувача

2.2 Сценарії використання програми

2.2.1 Отримання інформації про профілі

Передумова: програма запущена, адміністратор пройшов аутентифікацію.

- 1. Користувач переходить на вкладку "Профілі".
- 2. Програма надає список профілів відсортованих по категоріям.
- 3. Користувач виділяє певний профіль.
- 4. Програма відображає інформацію про нього.

2.2.2 Створення нової категорії у БД

Передумова: програма запущена, адміністратор пройшов аутентифікацію.

- 1. Користувач натискає на кнопку "Додати акаунт".
- 2. Програма відображає типи акаунтів.
- 3. Користувач вибирає один із типів.
- 4. Програма відображає діалогове вікно.
- 5. Користувач заповнює всі необхідні поля, натискає на кнопку "Створити".

2.2.3 Налаштування програми

Передумова: програма запущена, користувач пройшов аутентифікацію.

- 1. Користувач натискає на меню "Налаштування".
- 2. Програма зображує пункти меню "Налаштування".
- 3. З'являється вікно з налаштуваннями.
- 4. Користувач редагує налаштування, натискає на кнопку "Зберегти".
 - 2.2.4 Отримання загальної інформації про програму.

Передумова: програма запущена, користувач пройшов аутентифікацію.

- 1. Користувач натискає на меню "Допомога".
- 2. Програма відображає пункти меню "Допомога".
- 3. Користувач натискає на пункт "Про програму".
- 4. Програма відображає загальну інформацію про додаток.

2.3 Діаграма граничних класів

На рис. 2.10 зображена діаграма граничних класів програмного додатку "KPI City".

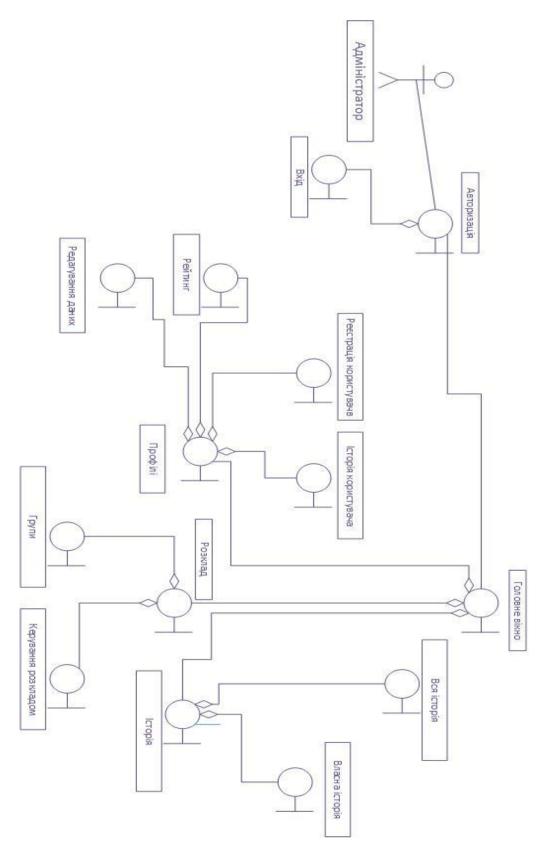


Рис. 2.10 Діаграма граничних класів

2.4 Проектування графічного інтерфейсу користувача

На рисунках 2.3.1 зображено ескізи моделей графічного інтерфейсу програмного додатку "KPI City". На ньому виділено області для розташування

всіх елементів. Графічний інтерфейс розроблявся таким чином, щоб адміністратор мав можливість швидко і легко знайти потрібну йому інформацію чи вибрати та здійснити операцію. Також при розробці інтерфейсу брались до уваги праці [8, 9], де автори описують тонкі аспекти побудови моделі вікна програмного додатку, які суттєво впливають на його привабливість і легкість сприйняття.

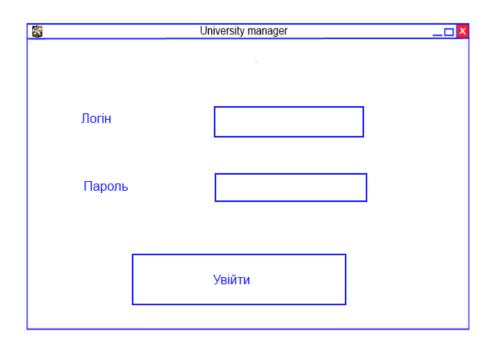


Рис 2.11 Ескіз вікна авторизації програми

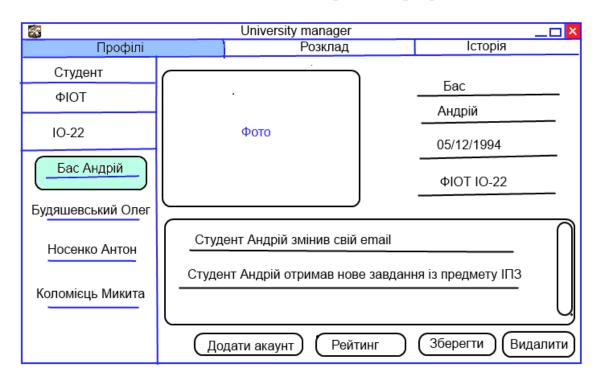


Рис 2.12 Ескіз вкладки "Профілі" головного вікна програми

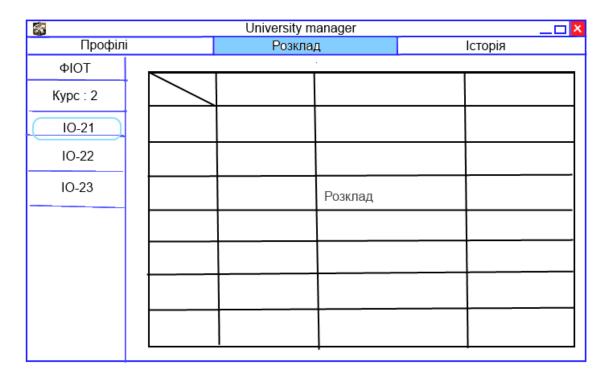


Рис 2.13 Ескіз вкладки "Розклад" головного вікна програми

g c		University manager		X
	Профілі	Розклад	Історія	
	Повна історія	•		\sim
_	Історія 1			_
_	Історія 2			
_	Історія 3			-
_	Історія 4			_
_	Історія 5			_
_	Історія 6			-

Рис 2.14 Ескіз вкладки "Історія" головного вікна програми

2.5 Таблиця відповідності елементів інтерфейсу

У таблиці 2.1 показані відповідності елементів ескізу графічного інтерфейсу рисунків 2.11 - 2.14 та класів пакету Swing, що їх реалізують.

Таблиця 2.1. Відповідність елементів інтерфейсу до класів

Елемент інтерфейсу	Класи пакету java.swing
Головне вікно	java.swing .JFrame
Панель «Меню»	java.swing .JMenuBar
Панель «Вкладки»	java.swing .JTabbedPane
Список вибору факультету	java.swing .JComboBox
Список вибору групи	java.swing .JComboBox
Список вибору профілю	java.swing .JList
Область історії	java.swing .JList
Кнопка «Файл»	java.swing .JMenu
Кнопка «Допомога»	java.swing .JMenu
Кнопка «Про програму»	java.swing .JMenuItem
Кнопка «Посібник користувача»	java.swing .JMenuItem
Кнопка «Додати акаунт	java.swing .JMenuItem
Кнопка «Вихід»	java.swing .JMenuItem
Бічна панель кнопок	java.swing .JPanel
Кнопка «Додати акаунт»	java.swing .JButton
Кнопка «Рейтинг»	java.swing .JButton
Кнопка «Зберегти»	java.swing .JButton
Кнопка «Видалити»	java.swing .JButton
Поля профілів	java.swing .JTextEdit
Назви полів профілів	java.swing .JLable
Область розкладу	java.swing .JTableLayout
Область рейтингу	java.swing .JTableLayout

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Структура програмного додатку

На діаграмі класів додатку, що розробляється, зображується структура додатку (див додаток A). Програма складається із класів пакетів ці, model, core а також config, dao. При розробці класів я використовував знання шаблонів проектування набуті в праці [1]. Також я використовував інформацію про мову програмування Java, принципи ООП та принципи побудови графічних інтерфейсів набуті з праць [2-10].

3.2 Специфікація класів програмного додатка

3.2.1 Package com.Oyster.app.model

Перелік Інтерфейсів		
Інтерфейс	Пояснення	
IProfile	Інтерфейс для узагальнення сутностей, що містять граничний клас Profile серед полів Created by oleg on 5/11/14.	

Перелік класів		
Class	Пояснення	
Absence	Граничний клас, що представляє таблицю ABSENCE_TBL у базі даних Проміжна сутність для отримання розкладу	
Classes	Граничний клас, що представляє таблицю CLASSES_TBL у базі даних	
Faculty	Граничний клас, що представляє таблицю FACULTY_TBL у базі даних	
Group	Граничний клас, що представляє таблицю GROUP_TBL у базі даних	

Mark	Граничний клас, що представляє таблицю MARK_TBL у базі даних Created by oleg on 5/22/14.
Message	Граничний клас, що представляє таблицю MESSAGE_TBL у базі даних Created by oleg on 5/11/14.
Profile	Граничний клас, що представляє таблицю PROFILE_TBL у базі даних
Sending	Граничний клас, що представляє таблицю SENDING_TBL у базі даних Created by oleg on 5/11/14.
Student	Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних
Subject	Граничний клас, що представляє таблицю TEACHER_TBL у базі даних
Task	Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних Created by oleg on 5/22/14.
Teacher	Граничний клас, що представляє таблицю TEACHER_TBL у базі даних

3.2.1.1 Interface IProfile

Method Summary

Методи	
Модифікатор і тип	Метод і Пояснення
Profile	getProfile() повертає профіль класу

getProfile

Profile getProfile()

повертає профіль класу

Returns:

граничний клас Profile

3.2.1.2 Class Absence

public class **Absence** extends java.lang.Object

Граничний клас, що представляє таблицю ABSENCE_TBL у базі даних Проміжна сутність для отримання розкладу

Поля	
Модифікатор і тип	Поле та пояснення
private java.util.UUID	classId
private java.util.UUID	groupId
private java.util.UUID	id

Constructors

Constructor and Description

Absence()

конструктор зазамовчуванням

Absence(java.util.UUID id, java.util.UUID groupId, java.util.UUID classId)

Конструктор класу Absence

Методи

Модифікатор і тип	Метод та Пояснення
java.util.UUID	getClassId()
	метод, що повертає id заняття
java.util.UUID	getGroupId()
	метод, що повертає id групи
java.util.UUID	getId()
	метод, що повертає id присутності
void	setClassId(java.util.UUID classId)
	метод, що встановлю ϵ id заняття
void	setGroupId(java.util.UUID groupId)
	метод, що встановлює id групи
void	setId(java.util.UUID id)
	метод, що встановлює id присутності

Absence

public Absence()

Absence

public Absence(java.util.UUID id, java.util.UUID groupId, java.util.UUID classId)

Конструктор класу Absence

Parameters:

id - ідентифікатор об'єкту groupId - Ідетифікатор групи classId - ідентифікатор заняття

3.2.1.3 Class Classes

Граничний клас, що представляє таблицю CLASSES TBL у базі даних

Поля		
Модифікатор і тип	Поле та пояснення	
private java.lang.String	audience	
private java.lang.String	building	
private int	date	
private java.util.UUID	id	
private java.util.UUID	subjectId	
private java.util.UUID	teacherId	

Методи

Модифікатор і тип	Метод та Пояснення
java.lang.String	getAudience()
	метод, що повертає номер аудиторії
java.lang.String	getBuilding()
	метод, що повертає номер корпусу
int	getDate()
	метод, що повертає час заняття
java.util.UUID	getId()
	поветрає ідентифікатор заняття
java.util.UUID	getSubjectId()
	повертає іd предмету
java.util.UUID	getTeacherId()
	повертає id викладача
void	setAudience(java.lang.String audience)
	метод, що встановлює номер аудиторії
void	setBuilding(java.lang.String building)
	метод, що встановлює номер корпусу
void	setDate(int date)
	метод, що встановлює час заняття
void	setId(java.util.UUID id)
	встановлює ідентифікатор заняття
void	setSubjectId(java.util.UUID subjectId)

	встановює іd предмету
void	setTeacherId(java.util.UUID teacherId)
	встановює id викладача

Classes

public Classes()

Classes

```
public Classes(java.util.UUID id,
java.util.UUID subjectId,
java.util.UUID teacherId,
java.lang.String building,
java.lang.String audience,
int date)
```

Parameters:

id - ідентифікатор поля заняття в таблиці бази даних subjectId - ідентифікатор поля предмету в таблиці бази даних teacherId - ідентифікатор поля викладач в таблиці бази даних building - ідентифікатор поля заняття в таблиці бази даних audience - номер аудиторії date - час проведення заняття

3.2.1.4 Class Faculty

Граничний клас, що представляє таблицю FACULTY_TBL у базі даних

Поля	
Модифікатор і тип	Поле та пояснення
private java.util.UUID	id

private java.lang.String	name	
Constructors		
Constructor and Description		
Faculty()		
Faculty(java.util.UUID id, java.	lang.String name)	
Конструктор класу Faculty		

Faculty

public Faculty(java.util.UUID id, java.lang.String name)

Конструктор класу Faculty

Parameters:

id - ідентифікатор факультету name - назва факультету

3.2.1.5 Class Group

Граничний клас, що представляє таблицю GROUP_TBL у базі даних

Поля	
Модифікатор і тип	Поле та пояснення
private Faculty	faculty
private java.util.UUID	facultyId
private java.util.UUID	id

name

Group

public Group(java.util.UUID id, java.util.UUID facultyId, java.lang.String name)

Parameters:

id - ідентифікатор поля групи в таблиці бази даних facultyId - ідентифікатор поля факультету в таблиці бази даних name - назва групи

3.2.1.6 Class Mark

Граничний клас, що представляє таблицю MARK_TBL у базі даних Created by oleg on 5/22/14.

Поля

Модифікатор і тип	Поле та пояснення
private java.util.UUID	id
private java.lang.String	mark
private java.util.UUID	studentId
private java.util.UUID	taskId

Mark

public Mark()

Mark

public Mark(java.util.UUID id, java.util.UUID taskId, java.util.UUID studentId, java.lang.String mark)

Parameters:

id - ідентифікатор поля оцінки в таблиці бази даних taskId - ідентифікатор поля завдання в таблиці бази даних studentId - ідентифікатор поля студента в таблиці бази даних mark - значення оцінки

3.2.1.7 Class Message

Граничний клас, що представляє таблицю MESSAGE_TBL у базі даних Created by oleg on 5/11/14.

Поля

Модифікатор і тип	Поле та пояснення
private java.util.UUID	id
private java.lang.String	message
private Profile	recipient
private Profile	sender
private java.lang.String	senderStr
private java.lang.String	topic

Constructor Detail

Message

public Message()

Message

public Message(java.util.UUID id, java.lang.String topic, java.lang.String message)

Parameters:

id - ідентифікатор поля повідомлення в таблиці бази даних topic - тема повідомлення message - повідомлення

3.2.1.8 Class Profile

Граничний клас, що представляє таблицю PROFILE_TBL у базі даних

Поля

Модифікатор і тип	Поле та пояснення
private java.lang.String	address
private java.lang.String	email
private java.lang.String	firstName
private java.util.UUID	id
private java.lang.String	password

private java.lang.String	phoneNom
private java.lang.String	secondName

Profile

public Profile()

Profile

```
public Profile(java.util.UUID id,
java.lang.String firstName,
java.lang.String secondName,
java.lang.String email,
java.lang.String password,
java.lang.String phoneNom,
java.lang.String address)
```

конструктор класу Profile

Parameters:

```
id - ідентифікатор поля профіль в таблиці бази даних firstName - ім'я куристувача secondName - прізвище користувача email - електронна адреса користувача password - пароль користувача phoneNom - номер телефону користувача address - адреса користувача
```

3.2.1.9 Class Sending

Граничний клас, що представляє таблицю SENDING_TBL у базі даних Created by oleg on 5/11/14.

Field Summary

Поля

Модифікатор і тип	Поле та пояснення
private java.util.UUID	id
private Message	message
private java.util.UUID	messageId
private IProfile	recipient
private java.util.UUID	recipientId
private IProfile	sender
private java.util.UUID	senderId

Sending

public Sending()

Sending

```
public Sending(java.util.UUID id,
java.util.UUID senderId,
java.util.UUID recipientId,
java.util.UUID messageId)
```

Parameters:

id - ідентифікатор поля розсилки в таблиці бази даних senderId - ідентифікатор відправника recipientId - ідентифікатор поля отримувача messageId - ідентифікатор поля повідомлення в таблиці бази даних

3.2.1.10 Class Student

Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних

Поля	
Модифікатор і тип	Поле та пояснення
private int	bookNum
private int	course
private Group	group
private java.util.UUID	groupId
private java.util.UUID	id
private Profile	profile
private java.util.UUID	profileId

Student

public Student()

Student

public Student(java.util.UUID id, java.util.UUID profileId, java.util.UUID groupId, int course, int bookNum)

констуктор класу Student

Parameters:

id - ідентифікатор поля студента в таблиці бази даних profileId - ідентифікатор профіля студента groupId - ідентифікатор групи студента course - курс студента

3.2.1.11 Class Subject

Граничний клас, що представляє таблицю TEACHER TBL у базі даних

Поля

Модифікатор і тип	Поле та пояснення
private java.util.UUID	id
private java.lang.String	name

Subject

public Subject()

Subject

public Subject(java.util.UUID id, java.lang.String name)

Parameters:

id - ідентифікатор поля предмет в таблиці бази даних name - ім'я предмету

3.2.1.12 Class Task

Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних Created by oleg on 5/22/14.

Модифікатор і тип Поле та пояснення private java.util.UUID groupId private java.util.UUID id

private java.lang.String	name
private Subject	subject
private java.util.UUID	subjectId

Task

public Task()

Task

public Task(java.util.UUID id, java.util.UUID groupId, java.util.UUID subjectId, java.lang.String name)

конструктор класу Task

Parameters:

id - ідентифікатор поля завдання в таблиці бази даних groupId - ідетифікатор групи, до якої відноситься завдання subjectId - ідетифікатор предмету, до якого відноситься завдання name - назва завдання

3.2.1.13 Class Teacher

Граничний клас, що представляє таблицю TEACHER_TBL у базі даних

Since:

4/15/14 11:03 PM

Поля	
Модифікатор і тип	Поле та пояснення
private java.util.UUID	id
private Profile	profile

private java.util.UUID	profileId	
private java.util.UUID	workerInfoId	

Constructor Detail

Teacher

public Teacher()

Teacher

public Teacher(java.util.UUID id, java.util.UUID profileId, java.util.UUID workerInfoId)

Конструктор класу Teacher

Parameters:

id - ідентифікатор викладача profileId - ідентифікатор профіля workerInfoId - ідентифікатор інформації про робітника

3.2.2 Package com.Oyster.config

Перелік Інтерфейсів

Інтерфейс	Пояснення
IConfig	Інтерфейс для всіх об'єктів конфігурації декларує методи для отримання та запису полів конфігурації

Перелік класів

Class	Пояснення
AppConfig	Клас конфігурації
ConfigReader	Абстрактний клас для уніфікованого доступу до об'єкту конфігурації

ConfigWriter	Абстрактний клас для уніфікованого доступу до об'єкту	
	конфігурації	

3.2.2.1 Interface IConfig

Інтерфейс для всіх об'єктів конфігурації декларує методи для отримання та запису полів конфігурації

Методи	
Модифікатор і тип	Метод та Пояснення
java.util.Set <java.lang.string></java.lang.string>	getAllKeys() повертає усі ключі із карти конфігурації прешого рівня
java.lang.Object	getValue(java.lang.String key) дістає значення із ключем key
void	setValue(java.lang.String key, java.lang.Object value) встановлює значення для поля із ключем key

setValue

void setValue(java.lang.String key, java.lang.Object value)

встановлює значення для поля із ключем кеу

Parameters:

key - ключ поля

value - значення поля

getValue

java.lang.Object getValue(java.lang.String key)

дістає значення із ключем кеу

Parameters:

key - ключ поля

Returns:

значення поля

getAllKeys

java.util.Set<java.lang.String> getAllKeys()

повертає усі ключі із карти конфігурації прешого рівня

Returns:

усі ключі першого рівня

3.2.2.2 Class AppConfig

Клас конфігурації

Method Summary

Метод та Пояснення
getAllKeys() Повертає множину всіх ключів
getInstance()
Метод для отримання об'єкту класу конфігурацій, якщо такого немає, створює його, використовує метод подвійної
перевірки, тому може безпечно використовуватись у багатопоточних застосунках

java.lang.Object	getValue(java.lang.String key) дістає значення із ключем key
java.lang.String	out() Виводить усі значення конфігурацій
void	setValue(java.lang.String key, java.lang.Object value) встановлює значення для поля із ключем key

3.2.2.3 Class ConfigReader

Абстрактний клас для уніфікованого доступу до об'єкту конфігурації

Методи	
Модифікатор і тип	Метод та Пояснення
abstract void	loadFromFile(java.lang.String path) завантажує зовнішній файл конфігурації
void	setConfig(IConfig config) встановлює об'єкт конфігурації

loadFromFile

public abstract void loadFromFile(java.lang.String path) завантажує зовнішній файл конфігурації

Parameters:

path - шлях до файлу

setConfig

public void setConfig(IConfig config) встановлює об'єкт конфігурації

Parameters:

config - об'єкт конфігурації

3.2.2.4 Class ConfigWriter

Абстрактний клас для уніфікованого доступу до об'єкту конфігурації

Методи	
Модифікатор і тип	Метод та Пояснення
abstract void	save(java.lang.String path) записує конфігураіцю у JSON - файл
void	setConfig(IConfig config) встановлює об'єкт конфігурації

save

public abstract void save(java.lang.String path)

записує конфігураіцю у JSON - файл

Parameters:

path - шлях до файлу

setConfig

public void setConfig(IConfig config)

встановлює об'єкт конфігурації

Parameters:

config - об'єкт конфігурації

3.2.2.5 JSONConfigReader

Клас зчитування конфігурації із JSON – файлу

Методи	
Модифікатор і тип	Метод та Пояснення
void	loadFromFile(java.lang.String path) завантажує зовнішній файл конфігурації

JSONConfigReader

public JSONConfigReader(IConfig config)

Конструктор класу JSONConfigReader

Parameters:

config - об'єкт, у якому зберігатиметься конфігурація

loadFromFile

public void loadFromFile(java.lang.String path)

завантажує зовнішній файл конфігурації

Specified by:

loadFromFile in class ConfigReader

Parameters:

path - шлях до файлу

3.2.2.6 Class JSONConfigWriter

Клас запису конфігурації в JSON – файлу

JSONConfigWriter

 $public\ JSONConfigWriter(IConfig\ config)$

Конструктор класу JSONConfigWriter

Parameters:

config - об'єкт, у якому зберігатиметься конфігурація

save

public void save(java.lang.String path)

записує конфігураіцю у JSON - файл

Specified by:

save in class ConfigWriter

Parameters:

path - шлях до файлу

3.2.3 Package com.Oyster.core.controller

Клас відповідає за реєстрацію, валідацію та виконання команд, реалізує паттерн Singleton

Constructors

Modifier	Constructor and Description
private	CommandExecutor()
	створює екземпляр класу CommandExecutor

Методи	
Модифікатор і тип	Метод та Пояснення
void	addCommand(java.lang.Class command)
	реєструє нову команду в CommandExecutor
protected java.lang.Object	clone()

	викидає помилку при спробі клонувати об'єкт, оскільки CommandExecutor повинен бути одиночкою
void	execute(java.lang.String action, Context params, java.lang.Runnable onPostExecute) виконує команду у фоновому режимі
private java.lang.Class	findCommandByAction (java.lang.String action) шукає команду за ключем
static CommandExecutor	getInstance() повертає змінну класу CommandExecutor

CommandExecutor

private CommandExecutor()

створює екземпляр класу CommandExecutor

getInstance

public static CommandExecutor getInstance()

повертає змінну класу CommandExecutor

Returns:

екземпляр класу CommandExecutor

execute

public void execute(java.lang.String action,
Context params,
java.lang.Runnable onPostExecute)
throws CommandNotFoundException,
java.lang.InstantiationException,
java.lang.IllegalAccessException,
InvalidCommandParameterException

виконує команду у фоновому режимі

Parameters:

action - ключ команди

params - контекст команди

Throws:

CommandNotFoundException

java.lang.InstantiationException

java.lang.IllegalAccessException

Invalid Command Parameter Exception

findCommandByAction

private java.lang.Class findCommandByAction(java.lang.String action)

шукає команду за ключем

Parameters:

action - ключ команди

Returns:

команду, якщо знаходить, інакше null

addCommand

public void addCommand(java.lang.Class command)

реєструє нову команду в CommandExecutor

Parameters:

command - class of the new command

clone

protected java.lang.Object clone()

throws java.lang.CloneNotSupportedException

викидає помилку при спробі клонувати об'єкт, оскільки CommandExecutor повинен бути одиночкою

Overrides:

clone in class java.lang.Object

Returns:

нічого

Throws:

java.lang.CloneNotSupportedException

3.2.3.1 Class Validator

Клас, що відповідає за валідацію команди

Методи	
Модифікатор	Метод та Пояснення
static boolean	validate(java.lang.Class command, Context context)
	виконує валідацію команди

validate

виконує валідацію команди

Parameters:

command - команда для валідації

context - контекст команди

Returns:

true - якщо команда пройшла валідацію, якщо ні - викидає помилку

Throws:

3.2.4 Package com.Oyster.core.controller.command

Перелік класів	
Class	Пояснення
AbstractCommand	Абстпрактний клас, що реалізує базову команду, містить контекст, та екземпляр інтерфейсу Runnable, та методи його отримання
Context	Клас відображає контекст команди
DeleteIProfileCommand	реалізує команду видалення профілю Created by oleg on 5/12/14.
FillRecipientsCommand	команда, для заповнення списку отримувачів повідомлення
LoadClassesCommand	команда, що реалізує завантаження занять з бази даних
LoadMaksCommand	команда для завантаження оцінок з бази даних
LoadMessageCommand	
LogInCommand	Реалізує команду авторизації в системі
RegisterStudentCommand	Реалізує команду для реєстрації користуваача
SendMessageCommand	команда для відправлення повідомлень

3.2.4.1 Class AbstractCommand

Абстпрактний клас, що реалізує базову команду, містить контекст, та екземпляр інтерфейсу Runnable, та методи його отримання

Поля	
Модифікатор і тип	Поле та пояснення

protected Context	context
	контекст, який передається команді

Методи	
Модифікатор і тип	Метод та Пояснення
java.lang.Object	addParameter(java.lang.String name, java.lang.Object value)
	додає параметер до контексту
Context	getContext()
	повертає контекст команди
java.lang.Runnable	getOnPostExecute()
	повертає екземпляр інтерфейсу Runnable, метод якого run() буде виконаний у UI потоці
java.lang.Object	removeParametr(java.lang.String name)
	видаляє параметер з контексту
void	setContext(Context context)
	встановлює контекст команди
void	setOnPostExecute(java.lang.Runnable runnable)
	встановлює екземпляр інтерфейсу Runnable, метод якого run() буде виконаний у UI потоці

3.2.4.2 Class DeleteIProfileCommand

реалізує команду видалення профілю Created by oleg on 5/12/14.

DeleteIProfileCommand

public DeleteIProfileCommand()

DeleteIProfileCommand

public DeleteIProfileCommand(Context context1)

Parameters:

context1 - параметри для видалення профілю

3.2.4.3 Class FillRecipientsCommand

Команда, для заповнення списку отримувачів повідомлення

FillRecipientsCommand

public FillRecipientsCommand()

FillRecipientsCommand

public FillRecipientsCommand(Context context1)

конструктор класу FillRecipientsCommand

Parameters:

context1 - параметри для заповнення списку отримувачів

3.2.4.4 Class LoadClassesCommand

Команда, що реалізує завантаження занять з бази даних

LoadClassesCommand

public LoadClassesCommand()

LoadClassesCommand

public LoadClassesCommand(Context context1)

конструктор класу LoadClassesCommand

Parameters:

context1 - параметри, необхідні для завантаження занять

3.2.4.5 Class LoadMaksCommand

Команда для завантаження оцінок з бази даних

Constructor Detail

LoadMaksCommand

public LoadMaksCommand()

LoadMaksCommand

 $public\ LoadMaksCommand(Context\ context1)$

конструктор класу LoadMaksCommand

Parameters:

context1 - параметри, необхідня для завантаження оцінок

3.2.4.6 Class LoadMessageCommand

Constructor Detail

LoadMessageCommand

public LoadMessageCommand()

LoadMessageCommand

public LoadMessageCommand(Context context1)

конструктор класу LoadMessageCommand

Parameters:

context1 - параметри, необхідня для завантаження повідомлень

3.2.4.7 Class LogInCommand

Методи	
Модифікатор і тип	Метод та Пояснення
static void	centreWindow(javax.swing.JFrame frame) встановлює вікно в центер екрану
void	run()

LogInCommand

public LogInCommand()

LogInCommand

public LogInCommand(Context context1)

Конструктор для класу LogInCommand

Parameters:

context1 - параметри для реєстрації

3.2.5 Package com.Oyster.dao

3.2.5.1 Interface CRUDInterface

визначає CRUD-інтерфейс для базових операцій із постійним місцем збереження

Методи	
Модифікатор і тип	Метод та Пояснення
<t> void</t>	delete(T instance) виконує базову операцію видалення для даної сутності
<t> T</t>	insert(T instance) виконує базову операцію вставлення для даної сутності
<t> T</t>	read(java.lang.Class entityClass, java.util.UUID id) виконує базову операцію зчитування для даної сутності
<t> java.util.List<t></t></t>	select(java.lang.Class entityClass, DAOFilter filter) виконує пошук сутностей у постійному хранилищі
<t> java.util.List<t></t></t>	select(java.lang.Class entityClass, java.lang.String SQLString) виконує пошук сутностей у базі даних

<T> void **update**(T instance)

виконує базову операцію оновлення для даної сутності

Insert

<T> T insert(T instance) throws DAOException

виконує базову операцію вставлення для даної сутності

Type Parameters:

Т - тип сутності

Parameters:

instance - сутність

Returns:

саму сутність

Throws:

DAOException

read

<T> T read(java.lang.Class entityClass, java.util.UUID id) throws DAOException

виконує базову операцію зчитування для даної сутності

Type Parameters:

Т - тип сутності

Parameters:

entityClass - клас сутності

id - ключ сутності

Returns: саму сутність **Throws:** DAOException update <T> void update(T instance) throws DAOException виконує базову операцію оновлення для даної сутності **Type Parameters:** Т - тип сутності **Parameters:** instance - сутність **Throws: DAOException** delete <T> void delete(T instance) throws DAOException виконує базову операцію видалення для даної сутності **Type Parameters:** Т - тип сутності **Parameters:** instance - сутність **Throws:** DAOException

select

<T> java.util.List<T> select(java.lang.Class entityClass, DAOFilter filter) throws DAOException

виконує пошук сутностей у постійному хранилищі

Type Parameters:

Т - тип сутності

Parameters:

entityClass - клас сутності

filter - фільтр

Returns:

список знайдених сутностей, що задовільняють даному фільтру

Throws:

DAOException

select

<T> java.util.List<T> select(java.lang.Class entityClass, java.lang.String SQLString) throws DAOException

виконує пошук сутностей у базі даних

Type Parameters:

Т - тип сутності

Parameters:

entityClass - клас сутності

SQLString - SQL-запит до бази даних

Returns:

список знайдених сутностей

Throws:

3.2.5.2 Interface DAOFilter

public interface **DAOFilter**

допоміжний інтерфейс для відбирання сутностей за певними параметрами

Методи	
Модифікатор і тип	Метод та Пояснення
<t> boolean</t>	accept(T entity) перевіряє, чи сутність задовільняє дані умови

accept

<T> boolean accept(T entity)

перевіряє, чи сутність задовільняє дані умови

Type Parameters:

Т - тип сутності

Parameters:

entity - сутність для перевірки

Returns:

true - якщо сутність задовільняє умови, false - якщо ні

3.2.5.3 Class DAOCRUDJdbc

Методи	
Модифікатор і тип	Метод та Пояснення
<t> void</t>	delete(T instance) видаляє сутність із сховища даних

static DAOCRUDJ dbc	getInstance (org.springframework.context.ApplicationConte xt context) отримує екземпляр JDBC dao
<t> T</t>	insert(T instance) додає екземпляр нової сутності у сховище даних
<t> T</t>	read(java.lang.Class entityClass, java.util.UUID id) запит сутності у сховищі даних по його UUID
<t> java.util.List<t></t></t>	select(java.lang.Class entityClass, DAOFilter filter) запит для доступу до сховища даних
private <t> java.util.List<t></t></t>	select(java.lang.Class entityClass, DAOFilter filter, java.lang.String sql) допоміжна функція для пертворення різних типів запитів
<t> java.util.List<t ></t </t>	select(java.lang.Class entityClass, java.lang.String SQLString) запит для доступу до сховища даних
<t> void</t>	update(T instance) оновлює сутність у сховищі даних

3.2.6 com.Oyster.ui

3.2.6.1 Class LoginFrame

Клас,що відображає вікно авторизації користувача у програмі

Constructors
Constructor and Description
LoginFrame()
Конструктор класу LoginFrame створює вікно, встановлює його розміри

Методи	
Модифікатор і тип	Метод та Пояснення
void	actionPerformed(java.awt.event.ActionEvent e)
static void	centreWindow(javax.swing.JFrame frame) встановлює вікно в центрі екрану
private void	placeComponents(javax.swing.JPanel panel) ініціалізує компоненти
private void	tryToLogIn() метод викликає команду авторизації користувача

LoginFrame

public LoginFrame()

Конструктор класу LoginFrame створює вікно, встановлює його розміри

placeComponents

private void placeComponents(javax.swing.JPanel panel)

ініціалізує компоненти

Parameters:

panel - панель, на якій будуть розміщені компоненти

actionPerformed

public void actionPerformed(java.awt.event.ActionEvent e)

Specified by:

 $action Performed\ in\ interface\ java.awt.event. Action Listener$

tryToLogIn

private void tryToLogIn()

метод викликає команду авторизації користувача

centreWindow

public static void centreWindow(javax.swing.JFrame frame)

встановлює вікно в центрі екрану

Parameters:

frame - вікно, яке потрібно встановити по центру

3.2.6.2 Class MainForm

Клас відображає головне вікно програми

Constructors

Constructor and Description

MainForm()

створює головне вікно програми

Методи	
Модифікатор і тип	Метод та Пояснення
void	actionPerformed(java.awt.event.ActionEvent e)
private void	fillInfoFields(Profile profile)
	Відображає поля, які берігаються в проіфлі
private void	fillInfoFields(Student student)
	Відображає поля, які берігаються в об'єкті
	студент
void	fillRecipients(java.lang.Class c)

	заповнення списку отримувачів повідомлення
private java.util.ArrayList< Group >	getGroups()
private void	getSchedule()
	Завантаження розкладу
private java.util.ArrayList< Subject >	getSubjects()
private void	loadHelp()
private void	loadMarks()
	Завантаження таблиці успішності
private void	loadMessage()
	завантажує повідомлення, які надіслані поточному користувачу
private void	logOut()
	Завершення поточної сесії користувача
private void	<pre>placeComponents(javax.swing.JComponent compo nent)</pre>
	ініціалізує компоненти
private void	removeProfile()
	видалення профілю
private void	saveButtonClick()
	збереження змін до облікового запису
private void	saveUpdate(Profile p)
	збереження змін у профілі
private void	saveUpdate(Student s)

	збереження змін у обєкті студент
private void	sendMessage()
	Відправлення повідомлення
private void	set Received Messages ()
private void	updateUI()
	оновлення графічного інтерфейсу

3.2.6.3 Class MarksTable

Клас, що відповідає за створення і наповнення таблиці з оцінками

Constructors	
Constructor and Description	
MarksTable(javax.swing.JFrame frame, javax.swing.JTable table, Group group,Subject subject)	
конструкотр створює об'єкт та ініціалізує поля	

Методи	
Модифікатор і тип	Метод та Пояснення
private void	createScheduleTable() метод ініціалізує структуру даних у якій зберігатимуться дані
private void	fillTable() заповнення таблиці даними про успішність

private int	findColumnByTaskId(java.util.UUID taskid)
	знаходження стовпця завдання за його id
private int	findRowByStudentId(java.util.UUID studentid)
	знаходження рядка студента за його id
private void	initialization()
	ініціалізуються списк студентів та завдань
private void	loadTable()
	метод встановлює модель для таблиці та викликає методи для ініціалізації та заповнення даних

MarksTable

public MarksTable(javax.swing.JFrame frame, javax.swing.JTable table,Group group,Subject subject)

конструкотр створює об'єкт та ініціалізує поля

Parameters:

frame - вікно у якому відображатиметься таблиця table - таблиця, у якій відображатимуться оцінки group - група для якої потрібно відобразити успішність subject - предмет, оцінки з якого потрібно відобразити

3.2.6.4 Class Registration

Клас що відображає вікно реєстрації нового користувача Created by oleg on 4/13/14.

Constructor and Description

Registration()

Створює вікно реєстрації

Методи		
Модифікатор і тип	Метод та Пояснення	
void	actionPerformed(java.awt.event.ActionEvent e)	
static void	centreWindow(javax.swing.JFrame frame) встановлює вікно в центрі екрану	
private void	placeComponents(javax.swing.JComponent component) Ініціалізує компоненти	
void	tryToRegister() перевіряє чи всі необхідні поля заповнені та викликає команду реєстрації	

3.2.6.5 Class ScheduleTbl

Клас, що відповідає за створення і наповнення таблиці з розкладом

Constructor and Description ScheduleTbl(javax.swing.JFrame frame, Group group, javax.swing.JTable table) конструкотр створює об'єкт та ініціалізує поля

Методи	
Модифікатор і тип	Метод та Пояснення
private void	createScheduleTable()
	ініціалізує модель даних, у якій зберігатимуеться розклад
private void	fillTable()
	заповнення таблиці даними про успішність
private Subject	getSubjectById(java.util.UUID id)
	повертає предмет за його id
private Teacher	getTeacherById(java.util.UUID id)
	повертає викладача за його id
private void	initialization()
	ініціалізуються списк студентів та викладачів
private void	loadTable()
	метод встановлює модель для таблиці та викликає методи для ініціалізації та заповнення даних

3.3 Інструкція для користувача

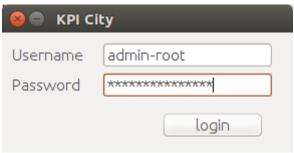
Для отримання інформації про профіль необхідно знайти його у вкладці "Профілі". Інформація з'явиться праворуч. Для того щоб преглянути розклад, необхідно перейти у вкладку 'Розклад" та вибрати групу та факультет. Там ж і можна змінити розклад. Для того щоб змінити налаштування програми необхідно натиснуты на пункт меню "Інструменти" -> "Налаштування". Потім заповнити необхідні поля та натиснути на кнопку "Зберегти". Для того щоб

створити новий акаунт необхідно натиснути на кнопку "Додати акаунт" та заповнити необхідні поля. Потім треба натиснути кнопку "Створити". Для того, щоб переглянути історію, необхідно перейти на вкладку "Історія"

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

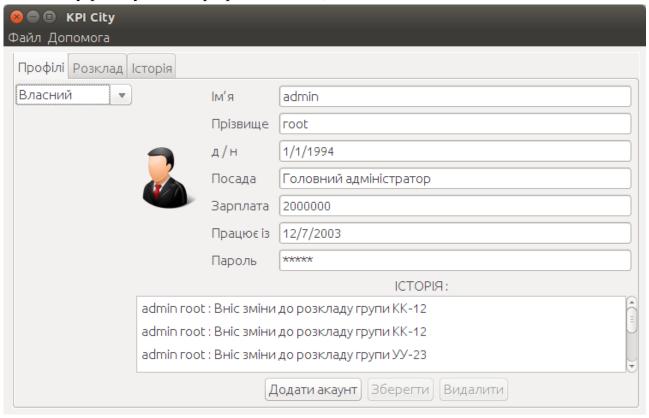
В якості тестування перевіримо усі преценденти.

На рисунку 4.1 представлено вікно авторизації (перевірка преценденту авторизації 2.1.1).



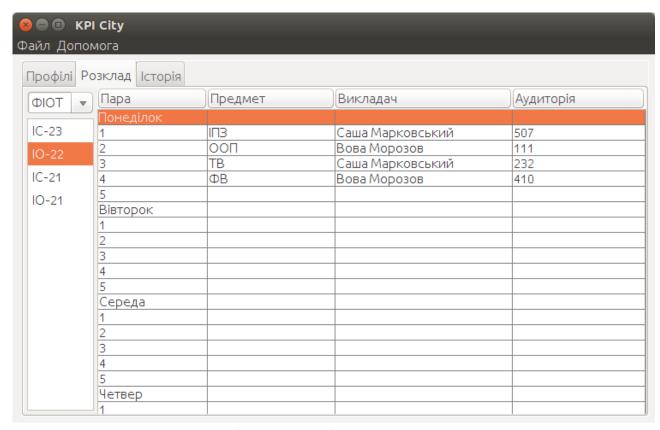
4.1 Вікно авторизації

На рисунку 4.2 представлено вікно управління профілями (перевірка преценденту редагування профілю п. 2.2.4).



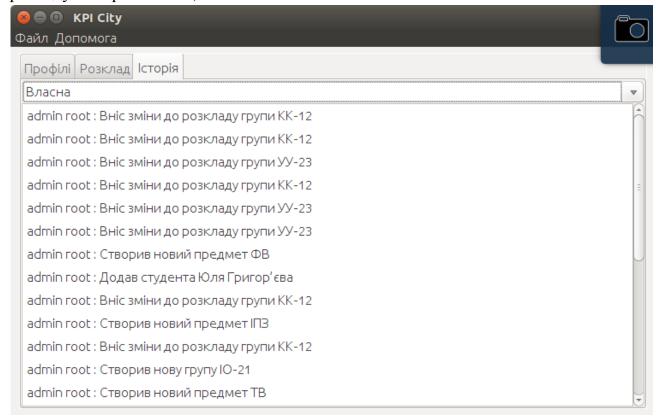
4.2 Вікно управління профілями

На рисунку 4.3 представлено вікно управління розкладом (перевірака преценденту перегляду та редагування розкладу п. 2.2.5, п. 2.2.6).



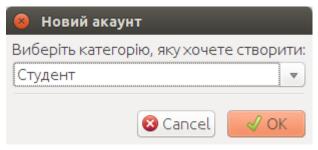
4.3 Вікно управління розкладом

На рисунку 4.3 представлено вікно перегляду історії (перевірка преценденту перегляду історії п. 2.2.7).



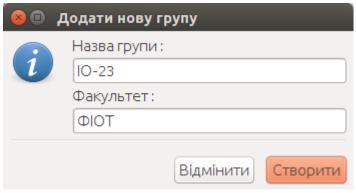
4.4 Вікно перегляду історії

На рисунку 4.5 представлено вікно вибору типу нового акаунту.



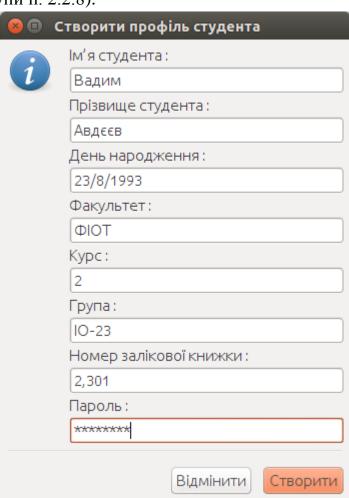
4.5 Вікно вибору типу нового акаунту

На рисунку 4.6 представлено вікно створення групи (перевірка преценденту створення нової групи п. 2.2.2).



4.6 Вікно створення групи

На рисунку 4.7 представлено вікно створення студента (перевірка преценденту створення нової групи п. 2.2.8).



4.7 Вікно створення студента

Під час проведення тестувваня усі преценденти пройшли перевіру, отже тестування проведено успішно.

висновки

Під час виконання роботи було закріплено теоретичні знання і практичні навички з проєктування, моделювання, розробки та тестування програмного забезпечення набуті у курсі інженерії програмного забезпечення, вивчив шаблон проєктування МVC.

Всі вимоги, які були зазначені в технічному завданні, в програмному додатку було повністю виконано. Інтерфейс адміністратора розроблено так, щоб можна було швидко і зручно оперувати профілями та розкладом.

В розробленому додатку реалізовано наступний функціонал: можливість авторизації, створення нового викладача, адміністратора, студента, факультету, групи, предмету, а також їх редагування. Реалізовано можливість редагування розкладу групи, а також перегляду історії.

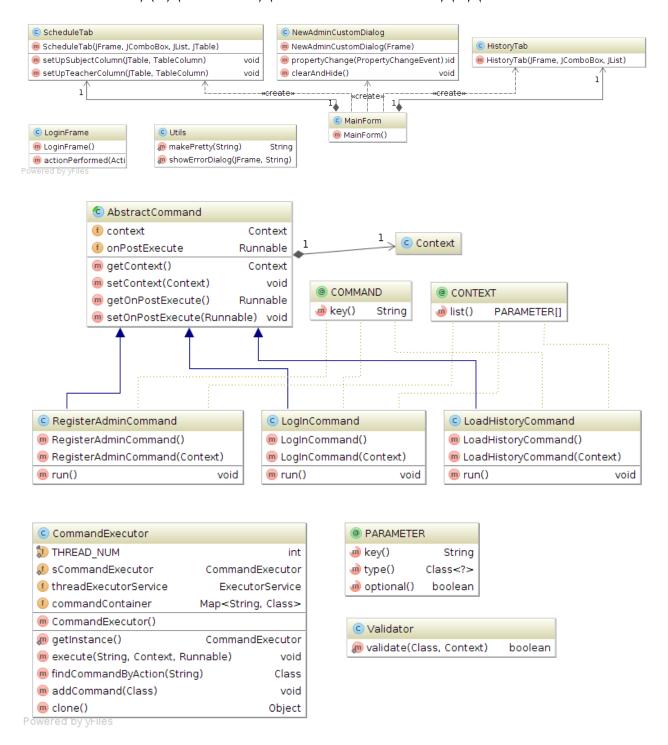
В процесі розробки програмного додатку, я ознайомився з основними класами пакету Swing, за допомогою яких будуються графічні додатки, а також закріпив ці знання на практиці.

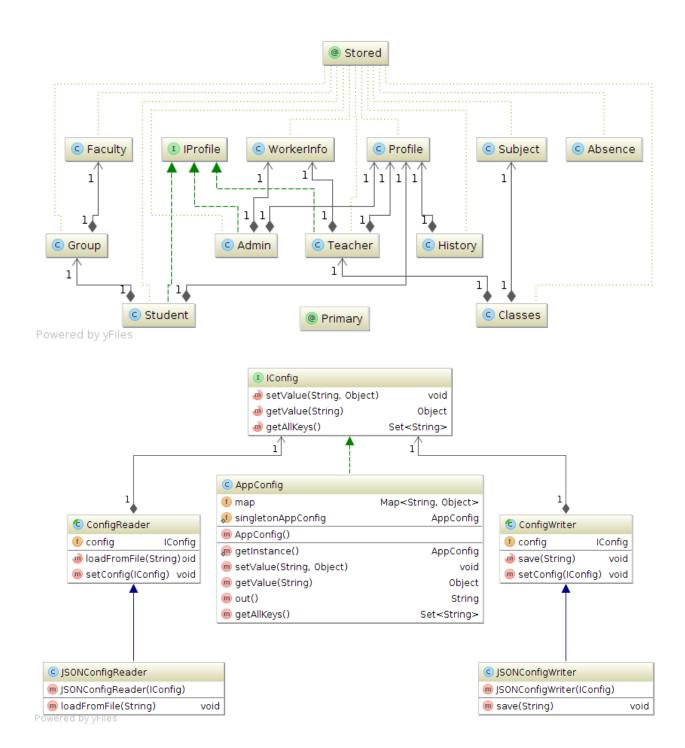
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

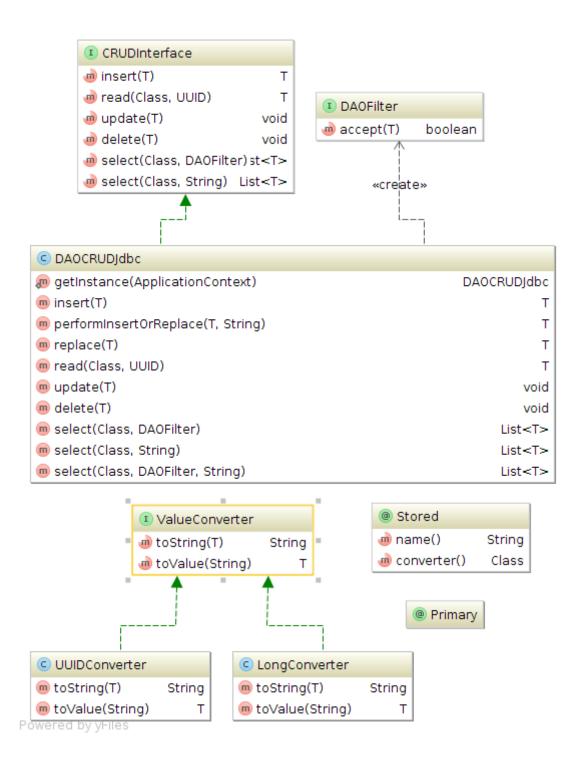
- 1. Приемы объектно-ориентированого проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. СПб.: Питер, 2011 368 с.: ил. –ISBN 978-5-469-01136-1.
- 2. Герберт Шилдт Java. Полное руководство, 8-е изд. : Пер. с англ. М. : ООО "И.Д. Вильямс", 20012. 1104 с. ISBN 978-5-8459-1759-1 (рус.)
- 3. Эккель Б. Философия Java / Эккель Брюс; Пер.с англ. Е.Матвеев. 4-е изд. СПб.: Питер, 2010. 640с.: ил. (Библиотека программиста). Алф.указ.:с.631. ISBN 978-5-388-00003-3.
- 4. Хорстманн Кей С. Java 2. Том 1. Основы / Кей Хорстманн, Гари Корнелл; Пер с англ. Изд. 8-е. М.: ООО "И.ДВильямс", 2011. 816 с.: ил. Парал. тит. англ. (Библиотека профессионала). –ISBN 978-5-8459-1378-4 (рус.).
- 5. Хорстманн Кей С. Java 2. Том 2. Тонкости программирования / Кей Хорстманн, Гари Корнелл; Пер с англ. Изд. 8-е. М.: ООО "И.ДВильямс", 2011. 992 с.: ил. Парал. тит. англ. (Библиотека профессионала). –ISBN 978-5-8459-1482-8 (рус.).
- 6. Стелтинг Стивен Применение шаблонов Java /Стелтинг Стивен, Маасен Олав; Пер. с англ. –М.: Издательский дом "Вильямс", 2002. 576 с.: ил. Парал. тит. англ. (Библиотека профессионала). ISBN 5-8459-0339-4 (рус.).
- 7. Дженифер Тидвелл Разработка пользовательских интрефейсов; Пер. с англ. –Е. Шикарева: Издательский дом "Питер", 2008. 416 с. ISBN 978-5-91180-073-4 (рус.).
- 8. А. К. Гультяев Проектирование и дизайн пользовательского интерфейса / В. А. Машин; Пер. с англ. : Издательский дом " Корона-Принт", 2010. 350 с. ISBN 978-5-7931-0814-0 (рус.).
- 9. Герберт Шилдт. Библиотека SWING для Java: руководство для начинающих Вильямс , 2007 С. 704 ISBN 978-5-8459-1162-9, 0-07-2263148.
- 10. Марк Гранд Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML = Patterns

in Java, Volume 1. A Catalog of Reusable Design Patterns Illustrated with UML. — М.: «Новое знание», 2004. — С. 560. — ISBN 5-94735-047-5

ДОДАТОК А. ДІАГРАМИ КЛАСІВ ДОДАТКУ







ДОДАТОК Б. ВИХІДНИЙ КОД ДОДАТКУ

```
* Повертає адміністратора, що останній раз
package com.oyster.app.model;
                                                                                                                                            return faculty;
import com.oyster.dao.annotation.Primary;
                                                                    авторизувався
import com.oyster.dao.annotation.Stored;
                                                                                                                                         public void setFaculty(Faculty faculty) {
                                                                      * @return адміністратора, що залогінився, або
                                                                                                                                            this.faculty = faculty;
com.oyster.dao.annotation.utils.converter.UUIDConvert
                                                                   null. якшо такого немає
                                                                                                                                          @Override
import java.util.UUID;
                                                                      public static Admin getCurrentAdmin() {
                                                                                                                                         public String toString() {
                                                                        return currentAdmin;
                                                                                                                                           return getName();
* Граничний клас, що представляє таблицю
FACULTY_TBL у базі даних
                                                                                                                                       package com.oyster.app.model;
* @author bamboo
                                                                      * Встановлює адміністратора, що розпочав сесію
                                                                                                                                       import com.oyster.dao.annotation.Primary;
@Stored(name = "FACULTY_TBL")
                                                                      * @param currentAdmin адміністратор, що
                                                                                                                                       import com.oyster.dao.annotation.Stored;
public class Faculty {
                                                                    здійснив авторизацію
  @Primary
                                                                                                                                       com.oyster.dao.annotation.utils.converter.UUIDConvert
  @Stored(name = "faculty_id", converter =
                                                                      public static void setCurrentAdmin(Admin
UUIDConverter.class)
                                                                   currentAdmin) {
  private UUID id;
                                                                        AppConst.currentAdmin = currentAdmin;\\
                                                                                                                                       import java.util.UUID;
  @Stored(name = "faculty_name")
                                                                                                                                        * Граничний клас, що представляє таблицю
  private String name;
                                                                   package com.oyster.app.model;
                                                                                                                                       HISTORY_TBL у базі даних
  public Faculty() {
                                                                   import com.oyster.dao.annotation.Primary;
                                                                   import com.oyster.dao.annotation.Stored;
                                                                                                                                        * @author bamboo
  public Faculty(UUID id, String name) {
                                                                   com. oyster. dao. annotation. utils. converter. UUID Convert\\
                                                                                                                                       @Stored(name = "HISTORY TBL")
    this.id = id:
    this.name = name:
                                                                                                                                       public class History {
                                                                   import java.util.UUID;
  public UUID getId() {
                                                                                                                                          @Stored(name = "history_id", converter =
    return id;
                                                                    * Граничний клас, що представляє таблицю
                                                                                                                                       UUIDConverter.class)
                                                                                                                                         private UUID id:
                                                                   GROUP_TBL у базі даних
  public void setId(UUID id) {
                                                                    * @author bamboo
                                                                                                                                          @Stored(name = "author_id", converter =
    this.id=id;\\
                                                                                                                                       UUIDConverter.class)
                                                                                                                                         private UUID authorId;
                                                                    @Stored(name = "GROUP TBL")
  public String getName() {
                                                                   public class Group {
                                                                                                                                          @Stored(name = "action")
                                                                                                                                         private String action:
    return name:
                                                                      @Primary
                                                                      @Stored(name = "group_id", converter =
                                                                                                                                         private Profile author:
  public void setName(String name) {
                                                                   UUIDConverter.class)
                                                                      private UUID id;
    this.name = name;
                                                                                                                                          public History() {
                                                                      @Stored(name = "faculty_id", converter =
  @Override
                                                                    UUIDConverter.class)
                                                                                                                                         public History(UUID id, UUID authorId, String
  public String toString() {
                                                                      private UUID facultyId;
                                                                                                                                       action) {
    return getName();
                                                                                                                                            this.id = id:
                                                                      @Stored(name = "group\_name")\\
                                                                                                                                            this.authorId = authorId;
                                                                      private String name;
                                                                                                                                            this.action = action;
package com.oyster.app;
                                                                      private Faculty faculty;
                                                                                                                                         public UUID getAuthorId() {
import com.oyster.app.model.Admin;
import com.oyster.config.AppConfig;
                                                                      public Group() {
                                                                                                                                            return authorId;
import\ com. oyster. dao. impl. DAOCRUDJ dbc;
org.spring framework.context. Application Context;\\
                                                                      public Group(UUID id, UUID facultyId, String name)
                                                                                                                                         public void setAuthorId(UUID authorId) {
                                                                                                                                           this.authorId = authorId;
import java.text.SimpleDateFormat;
                                                                        this.facultyId = facultyId;
                                                                                                                                         public UUID getId() {
                                                                        this.name = name;
* Клас містить константні поля(переважно
                                                                                                                                           return id;
статичні), що є спільними для
 * всього застосунку
                                                                      public String getName() {
                                                                                                                                         public void setId(UUID id) {
                                                                        return name;
public class AppConst {
                                                                                                                                            this.id = id;
  private static Admin currentAdmin;
                                                                      public void setName(String name) {
                                                                                                                                         public String getAction() {
  public static ApplicationContext CONTEXT;
  public static SimpleDateFormat DATE_FORMAT;
                                                                      public UUID getId() {
                                                                                                                                          public void setAction(String action) {
  public static int SESSION TIME;
                                                                        return id;
                                                                                                                                            this.action = action;
  public static DAOCRUDJdbc DAO;
                                                                      public void setId(UUID id) {
                                                                                                                                          @Override
  public static AppConfig APP_CONFIG;
                                                                        this.id = id;
                                                                                                                                         public String toString() {
                                                                                                                                            return String.format("%s %s: %s",
                                                                                                                                       getAuthor().getFirstName().
    DATE_FORMAT = new
                                                                      public UUID getFacultyId() {
                                                                                                                                       getAuthor().getSecondName(), getAction());
SimpleDateFormat("d/M/y");
                                                                        return facultyId;
     SESSION\_TIME = 0;
                                                                                                                                         public Profile getAuthor() {
                                                                      public void setFacultyId(UUID facultyId) {
                                                                                                                                           return author;
                                                                        this.facultyId = facultyId;
                                                                      public Faculty getFaculty() {
                                                                                                                                         public void setAuthor(Profile author) {
```

this.author = author;	}	public void setId(UUID id) {
}	public void setSecondName(String secondName) {	this.id = id;
package com.oyster.app.model;	this.secondName = secondName;	1
	}	<pre>public UUID getProfileId() {</pre>
/**	U. I	return profileId;
* Інтерфейс для узагальнення сутностей, що містять граничний клас Profile серед полів	<pre>public long getBirthday() { return birthday;</pre>	}
*	}	<pre>public void setProfileId(UUID profileId) {</pre>
* @author bamboo		this.profileId = profileId;
*/	public void setBirthday(long birthday) {	}
public interface IProfile {	this.birthday = birthday;	
F (j	public UUID getGroupId() {
/**	<pre>public String getPassword() {</pre>	return groupId;
* повертає профіль класу *	return password;	}
* @return граничний клас Profile	I	public void setGroupId(UUID groupId) {
*/	<pre>public void setPassword(String password) {</pre>	this.groupId = groupId;
<pre>public Profile getProfile();</pre>	this.password = password;	}
} package com.oyster.app.model;	}	<pre>public int getCourse() {</pre>
package conf.oyster.app.moder,	@Override	return course;
import com.oyster.dao.annotation.Primary;	public String toString() {	}
import com.oyster.dao.annotation.Stored;	return String.format("%s %s", getFirstName(),	
import	getSecondName());	<pre>public void setCourse(int course) { this.course = course;</pre>
com.oyster.dao.annotation.utils.converter.LongConverte r;	}	uns.course = course;
import	package com.oyster.app.model;	ı
com.oyster.dao.annotation.utils.converter.UUIDConvert		
er;	import com.oyster.dao.annotation.Primary;	public int getBookNum() {
import java.util.UUID;	import com.oyster.dao.annotation.Stored; import	return bookNum;
import java.um.0010,	com.oyster.dao.annotation.utils.converter.IntConverter;	1
/**	import	<pre>public void setBookNum(int bookNum) {</pre>
* Граничний клас, що представляє таблицю	com.oyster.dao.annotation.utils.converter.UUIDConvert	this.bookNum = bookNum;
PROFILE_TBL у базі даних	er;	}
* @author bamboo	import java.util.UUID;	<pre>public Profile getProfile() {</pre>
*/		return profile;
	/**	}
Off. I/ IIDDOFFIE TDVII	* Граничний клас, що представляє таблицю	II. II. D. CL. D. CL. CL.) (
<pre>@Stored(name = "PROFILE_TBL") public class Profile {</pre>	STUDENT_TBL у базі даних *	<pre>public void setProfile(Profile profile) { this.profile = profile;</pre>
public class Frome (* @author bamboo	lins.prome = prome,
@Primary	*/	,
@Stored(name = "profile_id", converter =		
UUIDConverter.class)	@Stored(name = "STUDENT_TBL")	public Group getGroup() {
private UUID id;	public class Student implements IProfile {	return group;
@Stored(name = "first_name")	@Primary	,
<pre>private String firstName = "";</pre>	@Stored(name = "student_id", converter =	<pre>public void setGroup(Group group) {</pre>
	UUIDConverter.class)	this.group = group;
0.0	t very very tra	
@Stored(name = "second_name")	private UUID id;	}
@Stored(name = "second_name") private String secondName = "";		} @Override
	<pre>private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class)</pre>	} @Override public String toString() {
private String secondName = "";	<pre>@Stored(name = "profile_id", converter =</pre>	<pre>public String toString() { return getProfile().toString();</pre>
private String secondName = ""; @Stored(name = "password") private String password;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class)</pre>	<pre>public String toString() { return getProfile().toString(); }</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter =	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId;</pre>	<pre>public String toString() { return getProfile().toString(); }</pre>
private String secondName = ""; @Stored(name = "password") private String password;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class)</pre>	<pre>public String toString() { return getProfile().toString(); }</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter =</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId;</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter =</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId;</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course;</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter =</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UIID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class)</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter =</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class)</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID;</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password;	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum;</pre>	<pre>public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /**</pre>
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UIID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; }	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile;</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних *
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum;</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * * @author bamboo
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UIID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; }	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile;</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних *
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UIID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; }	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile;</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL")
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { }</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищю SUBJECT_TBL у базі даних * @author bamboo */
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UIID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; }	<pre>@Stored(name = "profile_id", converter = UUIDCOnverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDCOnverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject {
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) {	<pre>@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { }</pre>	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) {	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId;	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject {
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() { return firstName;	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId; this.groupId = groupId;	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter =
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.password = password; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() {	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId; this.groupId = groupId; this.course = course;	public String toString() { return getProfile().toString(); } } package com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * @ author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter = UUIDConverter.class) private UUID id;
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.spassword = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() { return firstName; }	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId; this.groupId = groupId;	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "subject_name")
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() { return firstName;	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId; this.groupId = groupId; this.course = course;	public String toString() { return getProfile().toString(); } } package com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * @ author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter = UUIDConverter.class) private UUID id;
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.secondName = firstName; this.secondName = secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() { return firstName; } public void setFirstName(String firstName) {	@Stored(name = "profile_id", converter = UUIDCOnverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDCOnverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profiled = profileId; this.groupId = groupId; this.course = course; this.bookNum = bookNum; } public UUID getId() {	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "subject_name")
private String secondName = ""; @Stored(name = "password") private String password; @Stored(name = "birthday", converter = LongConverter.class) private long birthday; public Profile() { } public Profile(UUID id, String firstName, String secondName, String password, long birthday) { this.id = id; this.firstName = firstName; this.secondName; secondName; this.password = password; this.birthday = birthday; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public String getFirstName() { return firstName; } public void setFirstName(String firstName) { this.firstName = firstName;	@Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "course", converter = IntConverter.class) private int course; @Stored(name = "book_num", converter = IntConverter.class) private int bookNum; private Profile profile; private Group group; public Student() { } public Student(UUID id, UUID profileId, UUID groupId, int course, int bookNum) { this.id = id; this.profileId = profileId; this.groupId = groupId; this.course = course; this.bookNum = bookNum; }	public String toString() { return getProfile().toString(); } } package com.oyster.app.model; import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored; import com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю SUBJECT_TBL у базі даних * @author bamboo */ @Stored(name = "SUBJECT_TBL") public class Subject { @Primary @Stored(name = "subject_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "subject_name") private String name;

this.id = id;	III II TIGATAN IN C	this.salary = salary;
this.name = name;	public void setId(UUID id) {	this.dateHired = dateHired;
}	this.id = id;	}
	}	
public UUID getId() {		public UUID getId() {
return id;	<pre>public UUID getProfileId() {</pre>	return id;
}	return profileId;	}
	}	
public void setId(UUID id) {		<pre>public void setId(UUID id) {</pre>
this.id = id;	<pre>public void setProfileId(UUID profileId) {</pre>	this.id = id;
}	this.profileId = profileId;	}
	}	
public String getName() {		public String getPosition() {
return name;	public Profile getProfile() {	return position;
}	return profile;	}
	}	
<pre>public void setName(String name) {</pre>		public void setPosition(String position) {
this.name = name;	<pre>public void setProfile(Profile profile) {</pre>	this.position = position;
}	this.profile = profile;	}
	}	
@Override		<pre>public int getSalary() {</pre>
public String toString() {	<pre>public WorkerInfo getWorkerInfo() {</pre>	return salary;
return getName();	return workerInfo;	}
}	}	public void setSalary(int salary) {
		this.salary = salary;
@Override	<pre>public void setWorkerInfo(WorkerInfo workerInfo) {</pre>	}
public boolean equals(Object obj) {	this.workerInfo = workerInfo;	<pre>public long getDateHired() {</pre>
if (obj instanceof Subject) {	}	return dateHired;
return ((Subject)		}
obj).toString().equals(this.toString());	@Override	public void setDateHired(long dateHired) {
} else if (obj instanceof String) {	<pre>public String toString() {</pre>	this.dateHired = dateHired;
return ((String) obj).equals(this.toString());	return getProfile().toString();	}
}	}	}
return this.equals(obj);	,	package com.oyster.app.model;
}	@Override	1 , , , ,
}	public boolean equals(Object obj) {	import com.oyster.dao.annotation.Primary;
package com.oyster.app.model;	if (obj instanceof Teacher) {	import com.oyster.dao.annotation.Stored;
paemage commojsternappimoder,	return ((Teacher)	import
import com.oyster.dao.annotation.Primary;	obj).toString().equals(this.toString());	com.oyster.dao.annotation.utils.converter.UUIDConver
import com.oyster.dao.annotation.Stored;	} else if (obj instanceof String) {	er;
import connoyster.dao.annotation.stored,	return ((String) obj).equals(this.toString());	cı,
com.oyster.dao.annotation.utils.converter.UUIDConvert	}	import java.util.UUID;
er;	return this.equals(obj);	import java.uui.com,
ci,	l	/**
import java.util.UUID;	}	* Граничний клас, що представляє таблицю
import java.utii.eeib,	package com.oyster.app.model;	ABSENCE_TBL у базі даних
/**	package connoyster.app.moder,	*
/	import com aveter dec appotation Primery	* @author bamboo
* Граничний клас, що представляє таблицю	import com.oyster.dao.annotation.Primary;	*/
TEACHER_TBL у базі даних	import com.oyster.dao.annotation.Stored;	•
* A	import	@Stored(name = "ABSENCE_TBL")
* @author bamboo	com.oyster.dao.annotation.utils.converter.IntConverter;	public class Absence {
*/	import	@Primary
	com.oyster.dao.annotation.utils.converter.LongConverte	@Stored(name = "absence_id", converter =
O.C. IV HERD LOTTED MENT III	r;	UUIDConverter.class)
@Stored(name = "TEACHER_TBL")		
public class Teacher implements IProfile {	import	private UUID id;
public class Teacher implements IProfile { @Primary	com.oyster.dao.annotation.utils.converter.UUIDConvert	
<pre>public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter =</pre>	•	<pre>@Stored(name = "group_id", converter =</pre>
<pre>public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class)</pre>	com.oyster.dao.annotation.utils.converter.UUIDConvert er;	<pre>@Stored(name = "group_id", converter = UUIDConverter.class)</pre>
<pre>public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter =</pre>	com.oyster.dao.annotation.utils.converter.UUIDConvert	<pre>@Stored(name = "group_id", converter =</pre>
public class Teacher implements IProfile {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID;	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId;</pre>
<pre>public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class)</pre>	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /**	<pre>@Stored(name = "group_id", converter = UUIDConverter.class)</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class)	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID;	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId;</pre>
public class Teacher implements IProfile {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /**	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter =</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class)	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class)</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class)	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class)</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних *	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId;</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter =	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId;</pre>
public class Teacher implements IProfile {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId;</pre>
public class Teacher implements IProfile {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { }</pre>
public class Teacher implements IProfile { @ Primary @ Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @ Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @ Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL")	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID</pre>
public class Teacher implements IProfile { @ Primary @ Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @ Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @ Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL")	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) {
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo {	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id;</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId;
public class Teacher implements IProfile { @ Primary @ Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @ Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @ Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter =	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId;
public class Teacher implements IProfile { @ Primary @ Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @ Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @ Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class)	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo; public Teacher() { }	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId;
public class Teacher implements IProfile { @ Primary @ Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @ Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @ Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position")	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() {</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupld; this.classId = classId; } public UUID getId() { return id;
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfoId; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfoId) { this.id = id;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position")	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupId; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id;
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfoId; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfoId) { this.id = id; this.profileId = profileId;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class)	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) {</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfoId; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfoId) { this.id = id; this.profileId = profileId;	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter =	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) {</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfoId; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfoId) { this.id = id; this.profileId = profileId; this.workerInfoId = workerInfoId; }	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileld, UUID workerInfold) { this.id = id; this.profileId = profileId; this.workerInfold = workerInfold; } public UUID getWorkerInfold() {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter =	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() {
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfoId; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfoId) { this.id = id; this.profileId = profileId; this.workerInfoId = workerInfoId; }	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class)	<pre>@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; return groupId; }</pre>
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileld, UUID workerInfold) { this.id = id; this.profileId = profileId; this.workerInfold = workerInfold; } public UUID getWorkerInfold() {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter =	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() {
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) { this.rofileId = profileId; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; }	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupld; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) { this.jprofileId = profileId; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; } public void setWorkerInfold(UUID workerInfold) {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired; public WorkerInfo() {	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileld, UUID workerInfold) { this.id = id; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; } public void setWorkerInfold(UUID workerInfold) { this.workerInfold = workerInfold() { return workerInfold = workerInfold() { this.workerInfold = workerInfold() {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired;	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; } public VUID getGroupId() { return groupId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileId; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) { this.jprofileId = profileId; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; } public void setWorkerInfold(UUID workerInfold) {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired; public WorkerInfo() { }	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileId, UUID workerInfold) { this.id = id; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; } public void setWorkerInfold(UUID workerInfold) { this.workerInfold = workerInfold; }	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблищо WORKER_INFO_TBL у базі даних * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired; public WorkerInfo() { } public WorkerInfo(UUID id, String position, int	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupld, UUID classId) { this.id = id; this.groupld = groupld; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupld; } public UUID getGroupId() { return groupId; }
public class Teacher implements IProfile { @Primary @Stored(name = "teacher_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "profile_id", converter = UUIDConverter.class) private UUID profileld; @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID workerInfold; private Profile profile; private Profile profile; private WorkerInfo workerInfo; public Teacher() { } public Teacher(UUID id, UUID profileld, UUID workerInfold) { this.id = id; this.workerInfold = workerInfold; } public UUID getWorkerInfold() { return workerInfold; } public void setWorkerInfold(UUID workerInfold) { this.workerInfold = workerInfold() { return workerInfold = workerInfold() { this.workerInfold = workerInfold() {	com.oyster.dao.annotation.utils.converter.UUIDConvert er; import java.util.UUID; /** * Граничний клас, що представляє таблицю WORKER_INFO_TBL у базі даних * * @author bamboo */ @Stored(name = "WORKER_INFO_TBL") public class WorkerInfo { @Primary @Stored(name = "worker_info_id", converter = UUIDConverter.class) private UUID id; @Stored(name = "position") private String position; @Stored(name = "salary", converter = IntConverter.class) private int salary; @Stored(name = "date_hired", converter = LongConverter.class) private long dateHired; public WorkerInfo() { }	@Stored(name = "group_id", converter = UUIDConverter.class) private UUID groupld; @Stored(name = "class_id", converter = UUIDConverter.class) private UUID classId; public Absence() { } public Absence(UUID id, UUID groupId, UUID classId) { this.id = id; this.groupId = groupId; this.classId = classId; } public UUID getId() { return id; } public void setId(UUID id) { this.id = id; } public UUID getGroupId() { return groupId; } public VUID getGroupId() { return groupId; }

	return getProfile().toString();	this.subjectId = subjectId;
public void setClassId(UUID classId) {	}	}
this.classId = classId;	} package com.oyster.app.model;	public UUID getId() {
}	package comoyster.app.moder,	return id;
package com.oyster.app.model;	import com.oyster.dao.annotation.Primary; import com.oyster.dao.annotation.Stored;	}
import com.oyster.dao.annotation.Primary;	import	<pre>public void setId(UUID id) {</pre>
import com.oyster.dao.annotation.Stored;	com.oyster.dao.annotation.utils.converter.IntConverter;	this.id = id;
import	import	}
com.oyster.dao.annotation.utils.converter.UUIDConvert er;	com.oyster.dao.annotation.utils.converter.UUIDConvert er;	<pre>public Subject getSubject() {</pre>
import java.util.UUID;	import java.util.UUID;	return subject; }
/** * Граничний клас, що представляє таблицю	/** * Граничний клас, що представляє таблицю	<pre>public void setSubject(Subject subject) { this.subject = subject;</pre>
АDMIN_TBL у базі даних	CLASSES_TBL у базі даних	lins.subject = subject,
*	*	,
* @author bamboo */	* @author bamboo */	<pre>public Teacher getTeacher() { return teacher; }</pre>
@Stored(name = "ADMIN_TBL")	@Stored(name = "CLASSES_TBL")	1
public class Admin implements IProfile {	public class Classes {	<pre>public void setTeacher(Teacher teacher) {</pre>
@Primary	OD:	this.teacher = teacher;
@Stored(name = "admin_id", converter = UUIDConverter.class)	<pre>@Primary @Stored(name = "classes_id", converter =</pre>	}
private UUID id;	UUIDConverter.class)	package com.oyster.app.model;
	private UUID id;	
@Stored(name = "profile_id", converter =	00. 1/ 11	import com.oyster.dao.annotation.Primary;
UUIDConverter.class) private UUID profileId;	<pre>@Stored(name = "subject_id", converter = UUIDConverter.class)</pre>	import com.oyster.dao.annotation.Stored; import
private CCD prometa,	private UUID subjectId;	com.oyster.dao.annotation.utils.converter.UUIDConver
@Stored(name = "worker_info_id", converter =	r	er;
UUIDConverter.class)	<pre>@Stored(name = "teacher_id", converter =</pre>	
private UUID workerInfoId;	UUIDConverter.class)	import java.util.UUID;
private Profile profile;	private UUID teacherId;	/**
private Frome prome,	@Stored(name = "audience", converter =	* Граничний клас, що представляє таблицю
private WorkerInfo workerInfo;	IntConverter.class)	FACULTY_TBL у базі даних
IF A1 : 0 (private int audience;	*
public Admin() {	@Stored(name = "class_date", converter =	* @author bamboo */
,	IntConverter.class)	@Stored(name = "FACULTY_TBL")
public Admin(UUID id, UUID profileId, UUID workerInfoId) {	private int time;	public class Faculty {
this.id = id; this.profileId = profileId;	private Subject subject;	@Primary @Stored(name = "faculty_id", converter =
this.workerInfoId = workerInfoId;	private Teacher teacher;	UUIDConverter.class) private UUID id;
,	public Classes() {	private corb id,
public UUID getId() {	}	<pre>@Stored(name = "faculty_name")</pre>
return id;	and it classed the partial than the state of the partial than	private String name;
}	public Classes(UUID id, UUID subjectId, UUID teacherId, int audience, int time) {	public Faculty() {
public void setId(UUID id) {	this.id = id;	}
this.id = id;	this.subjectId = subjectId;	
}	this.teacherId = teacherId;	public Faculty(UUID id, String name) {
public UUID getWorkerInfoId() {	this.audience = audience;	this.id = id; this.name = name;
return workerInfold;	this.time = time;	lins.name = name,
}	}	,
		public UUID getId() {
<pre>public void setWorkerInfoId(UUID workerInfoId) { this.workerInfoId = workerInfoId;</pre>	<pre>public int getTime() { return time;</pre>	return id;
workermioid – workermioid,	}	ı
		<pre>public void setId(UUID id) {</pre>
public UUID getProfileId() {	public void setTime(int time) {	this.id = id;
return profileId;	this.time = time;	}
j	J	<pre>public String getName() {</pre>
<pre>public void setProfileId(UUID profileId) {</pre>	<pre>public int getAudience() {</pre>	return name;
this.profileId = profileId;	return audience;	}
}	}	public void setName(String name) {
<pre>public Profile getProfile() {</pre>	<pre>public void setAudience(int audience) {</pre>	this.name = name;
return profile;	this.audience = audience;	}
}	}	00 11
<pre>public void setProfile(Profile profile) {</pre>		@Override public String toString() {
this.profile = profile;	public UUID getTeacherId() {	return getName();
}	return teacherId;	}
nublic WorkerInfo getWestesInfo() (}) poskogo com oveter config impli
<pre>public WorkerInfo getWorkerInfo() { return workerInfo;</pre>	<pre>public void setTeacherId(UUID teacherId) {</pre>	package com.oyster.config.impl;
}	this.teacherId = teacherId;	
	}	$import\ org. code haus. jackson. map. Object Mapper;$
public void setWorkerInfo(WorkerInfo workerInfo) {	muhlia IIIID aarQukis stalo (import org.codehaus.jackson.type.TypeReference;
this.workerInfo = workerInfo;	<pre>public UUID getSubjectId() { return subjectId;</pre>	import com.oyster.config.ConfigReader; import com.oyster.config.IConfig;
,	}	p
@Override		import java.io.File;
public String toString() {	<pre>public void setSubjectId(UUID subjectId) {</pre>	import java.util.Map;

```
import java.util.Set:
                                                                        } catch (Exception e) {
                                                                                                                                              ObjectMapper mapper = new ObjectMapper();
                                                                           e.printStackTrace();
                                                                                                                                               // read JSON from a file
                                                                                                                                               Map<String, Object> map = mapper.readValue(
* клас реалізує зчитування файлу конфігурації із
                                                                                                                                                   new File(path),
JSON - файлу
                                                                   package com.oyster.config.impl;
                                                                                                                                                   new TypeReference<Map<String,
                                                                                                                                        Object>>() {
public class JSONConfigReader extends ConfigReader {
                                                                    import org.codehaus.jackson.map.ObjectMapper;
                                                                   import org.codehaus.jackson.type.TypeReference;
  * Конструктор
                                                                   import\ com. oyster. config. ConfigReader;
                                                                                                                                              Set<String> set = map.keySet();
   * @param config об'єкт конфігурації
                                                                   import com.oyster.config.IConfig;
                                                                                                                                              for (String key : set) {
                                                                                                                                                 config.setValue(key, map.get(key));
  public JSONConfigReader(IConfig config) {
                                                                    import java.io.File;
     this.config = config;
                                                                    import java.util.Map;
                                                                                                                                            } catch (Exception e) {
                                                                    import java.util.Set;
                                                                                                                                              e.printStackTrace();
   * завантажує зовнішній файл конфігурації
   * @param path шлях до файлу
                                                                    * клас реалізує зчитування файлу конфігурації із
                                                                   JSON - файлу
                                                                                                                                       package com.oyster.config.impl;
  @Override
  public void loadFromFile(String path) {
                                                                    public class JSONConfigReader extends ConfigReader {
                                                                                                                                        import org.codehaus.jackson.map.ObjectMapper;
                                                                                                                                        import org.codehaus.jackson.type.TypeReference;
       ObjectMapper mapper = new ObjectMapper();
                                                                      * Конструктор
                                                                                                                                        import com.oyster.config.ConfigReader;
       // read JSON from a file
                                                                       * @param config об'єкт конфігурації
                                                                                                                                        import com.oyster.config.IConfig;
       Map<String, Object> map = mapper.readValue(
           new File(path),
                                                                      public JSONConfigReader(IConfig config) {
                                                                                                                                        import java.io.File;
           new TypeReference<Map<String,
                                                                        this.config = config;
                                                                                                                                        import java.util.Map;
                                                                                                                                        import java.util.Set;
Object>>() {
       );
                                                                       * завантажує зовнішній файл конфігурації
                                                                                                                                        * клас реалізує зчитування файлу конфігурації із
       Set<String> set = map.keySet();
                                                                       * @param path шлях до файлу
                                                                                                                                        JSON - файлу
       for (String key : set) {
                                                                      @Override
         config.setValue(key, map.get(key));
                                                                      public void loadFromFile(String path) {
                                                                                                                                        public class JSONConfigReader extends ConfigReader {
     } catch (Exception e) {
                                                                                                                                           * Конструктор
       e.printStackTrace();
                                                                           ObjectMapper\ mapper = new\ ObjectMapper();
                                                                           // read JSON from a file
                                                                                                                                           * @param config об'єкт конфігурації
                                                                           Map<String, Object> map = mapper.readValue(
                                                                                                                                          public JSONConfigReader(IConfig config) {
                                                                                new File(path),
package com.oyster.config.impl;
                                                                                new TypeReference<Map<String,
                                                                   Object>>() {
import\ org.code haus. jacks on. map. Object Mapper;
                                                                           );
import org.codehaus.jackson.type.TypeReference;
                                                                                                                                           * завантажує зовнішній файл конфігурації
import com.oyster.config.ConfigReader;
                                                                           Set<String> set = map.keySet();
                                                                                                                                           * @param path шлях до файлу
import com.oyster.config.IConfig;
                                                                           for (String key : set) {
                                                                             config.setValue(key, \, map.get(key));\\
                                                                                                                                          @Override
                                                                                                                                          public void loadFromFile(String path) {
import java.io.File;
import java.util.Map;
import java.util.Set;
                                                                        } catch (Exception e) {
                                                                           e.printStackTrace();
                                                                                                                                              ObjectMapper mapper = new ObjectMapper();
                                                                                                                                               // read JSON from a file
                                                                                                                                              Map<String, Object> map = mapper.readValue(
* клас реалізує зчитування файлу конфігурації із
                                                                                                                                                   new File(path),
JSON - файлу
                                                                   package com.oyster.config.impl;
                                                                                                                                                   new TypeReference<Map<String,
                                                                                                                                        Object>>() {
public class JSONConfigReader extends ConfigReader {
                                                                    import\ org.code haus.jackson.map. Object Mapper;
                                                                                                                                              );
                                                                   import\ org.code haus. jacks on. type. Type Reference;
  * Конструктор
                                                                   import com.oyster.config.ConfigReader;
                                                                                                                                              Set < String > set = map.keySet();
   * @param config об'єкт конфігурації
                                                                    import com.oyster.config.IConfig;
                                                                                                                                              for (String key : set) {
                                                                                                                                                 config.setValue(key, map.get(key));
  public JSONConfigReader(IConfig config) {
                                                                   import java.io.File;
    this.config = config;
                                                                   import java.util.Map;
                                                                   import java.util.Set;
                                                                                                                                            } catch (Exception e) {
                                                                                                                                              e.printStackTrace();
   * завантажує зовнішній файл конфігурації
   * @param path шлях до файлу
                                                                    * клас реалізує зчитування файлу конфігурації із
                                                                   JSON - файлу
  @Override
                                                                                                                                       package com.oyster.config.impl;
  public void loadFromFile(String path) {
                                                                    public class JSONConfigReader extends ConfigReader {
                                                                                                                                        import\ org.code haus. jacks on. map. Object Mapper;
                                                                      * Конструктор
       ObjectMapper mapper = new ObjectMapper();
                                                                                                                                       import\ org.code haus. jacks on. type. Type Reference;
                                                                       * @param config об'єкт конфігурації
       // read JSON from a file
                                                                                                                                        import com.oyster.config.ConfigReader;
       Map<String, Object> map = mapper.readValue(
                                                                                                                                        import com.oyster.config.IConfig;
           new File(path),
                                                                      public JSONConfigReader(IConfig config) {
            new TypeReference<Map<String,
                                                                                                                                        import java.io.File;
                                                                        this.config = config;
                                                                                                                                        import java.util.Map;
Object>>() {
                                                                                                                                        import java.util.Set;
      );
                                                                       * завантажує зовнішній файл конфігурації
       Set<String> set = map.keySet();
                                                                       * @param path шлях до файлу
       for (String key : set) {
                                                                                                                                        * клас реалізує зчитування файлу конфігурації із
         config.setValue(key, map.get(key));\\
                                                                                                                                        JSON - файлу
                                                                      public void loadFromFile(String path) {
                                                                                                                                        public class JSONConfigReader extends ConfigReader {
```

	}	/**
/**		* повертає список усіх параметрів команди
* Конструктор	return true;	*
* @param config об'єкт конфігурації	}	* @рагат с Клас команди
*/	1	* @return список усіх параметрів команди
•	J	
<pre>public JSONConfigReader(IConfig config) {</pre>	package com.oyster.core.controller.utils;	*/
this.config = config;		public static PARAMETER[] getParameterList(Class
}	import	c) {
	com.oyster.core.controller.annotation.COMMAND;	CONTEXT t = (CONTEXT)
/**	import	<pre>c.getAnnotation(CONTEXT.class);</pre>
* завантажує зовнішній файл конфігурації	com.oyster.core.controller.annotation.CONTEXT;	if $(t == null)$ return null;
* @param path шлях до файлу	import	return t.list();
*/	com.oyster.core.controller.annotation.PARAMETER;	1
@Override	comoyster.core.controller.annotation.r/M/M/M/E/ER,	J
public void loadFromFile(String path) {	/**	/**
public void loadi folili fie(String patii) (,	,
	 Допоміжний клас для роботи із аннотаціями 	* визначає, чи є даний параметр обов'язковим
try {	команди	*
ObjectMapper mapper = new ObjectMapper();	*/	 * @param paramKey ключ параметра
// read JSON from a file	public class ControllerAnnotationUtils {	 * @param command команда, у якій перевіряємо
Map <string, object=""> map = mapper.readValue(</string,>		 * @return true - якщо необов'язковий, інакше -
new File(path),	/**	false
new TypeReference <map<string,< td=""><td>* повертає ключ команди</td><td>*/</td></map<string,<>	* повертає ключ команди	*/
Object>>() {	*	public static boolean paramIsOptional(String
1	* @рагат с Клас команди	paramKey, Class command) {
);	* @return ключ команди	paramites, crass command) (
),	*/	DADAMETEREI al = corporameter List (common d).
S. 4 - St		PARAMETER[] pl = getParameterList(command);
Set <string> set = map.keySet();</string>	public static String getCommandKey(Class c) {	for (PARAMETER p : pl) {
for (String key : set) {	COMMAND t = (COMMAND)	if (p.key().equals(paramKey)) return
config.setValue(key, map.get(key));	<pre>c.getAnnotation(COMMAND.class);</pre>	p.optional();
}	return (t != null) ? t.key() : null;	}
	}	
} catch (Exception e) {		return true;
e.printStackTrace();	/**	}
}	* повертає список усіх параметрів команди	١,
ı	*	J
	***	package com.oyster.core.controller.command;
}	* @рагат с Клас команди	
}	* @return список усіх параметрів команди	import com.oyster.app.AppConst;
package com.oyster.core.controller.utils;	*/	import com.oyster.app.model.Admin;
	<pre>public static PARAMETER[] getParameterList(Class</pre>	import com.oyster.app.model.Profile;
import	c) {	import com.oyster.app.model.WorkerInfo;
com.oyster.core.controller.annotation.COMMAND;	CONTEXT t = (CONTEXT)	import
import	c.getAnnotation(CONTEXT.class);	com.oyster.core.controller.annotation.COMMAND;
com.oyster.core.controller.annotation.CONTEXT;	if (t == null) return null;	import
import	return t.list();	com.oyster.core.controller.annotation.CONTEXT;
com.oyster.core.controller.annotation.PARAMETER;	1	import
como jotorico controloriamio antomi i ira anta i ante	j	com.oyster.core.controller.annotation.PARAMETER;
/**	/**	import com.oyster.dao.DAOFilter;
'	•	•
* Допоміжний клас для роботи із аннотаціями	* визначає, чи є даний параметр обов'язковим	import com.oyster.dao.exception.DAOException;
команди	* 0	import com.oyster.ui.MainForm;
*/	* @рагат рагатКеу ключ параметра	import com.oyster.ui.Utils;
public class ControllerAnnotationUtils {	* @param command команда, у якій перевіряємо	
	 * @return true - якщо необов'язковий, інакше - 	import javax.swing.*;
/**	false	import java.util.List;
* повертає ключ команди	*/	
*	public static boolean paramIsOptional(String	/**
* @param с Клас команди	paramKey, Class command) {	 команда виконує авторизацію користувача із
* @return ключ команди		логіном та паролем, що ії передаються
*/	PARAMETER[] pl = getParameterList(command);	*/
public static String getCommandKey(Class c) {	for (PARAMETER p : pl) {	@COMMAND(key = "logIn")
COMMAND t = (COMMAND)	if (p.key().equals(paramKey)) return	@CONTEXT(list = {
	* * * * * * * * * * * * * * * * * * * *	· · · ·
c.getAnnotation(COMMAND.class);	p.optional();	@PARAMETER(key = "username", type =
return (t != null) ? t.key() : null;	}	String.class),
}		
		@PARAMETER(key = "password", type =
	return true;	<pre>@PARAMETER(key = "password", type = String.class)</pre>
/**	return true; }	String.class) })
/** * повертає список усіх параметрів команди	return true; } }	String.class)
'	return true; } } package com.oyster.core.controller.utils;	String.class) })
* повертає список усіх параметрів команди *	}	String.class) })
* повертає список усіх параметрів команди * * @рагат с Клас команди	} } package com.oyster.core.controller.utils;	String.class))) public class LogInCommand extends AbstractCommand {
* повертає список усіх параметрів команди * * @рагат с Клас команди * @return список усіх параметрів команди	} package com.oyster.core.controller.utils; import	String.class) })
* повертає список усіх параметрів команди * * @рагат с Клас команди * @return список усіх параметрів команди */	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND;	String.class))) public class LogInCommand extends AbstractCommand {
* повертає список усіх параметрів команди * * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { }
* повертає список усіх параметрів команди *	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT;	String.class))) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /**
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) { CONTEXT t = (CONTEXT)	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { }
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT;	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор *
* повертає список усіх параметрів команди *	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER;	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * * @param context1 контекст команди
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /**	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */
* повертає список усіх параметрів команди *	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER;	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) {
* повертає список усіх параметрів команди * @ратат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) {
* повертає список усіх параметрів команди * @ратат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) {
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) {
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /**
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /**	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); }
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils {	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди */	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Kонструктор * * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * * @рагат с Клас команди	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * * @рагат с Клас команди * @return ключ команди * @return ключ команди	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override public void run() {
* повертає список усіх параметрів команди * @рагат є Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) { CONTEXT t = (CONTEXT) c.getAnnotation(CONTEXT.class); if (t == null) return null; return t.list(); } /** * визначає, чи є даний параметр обов'язковим * * @рагат рагатКеу ключ параметра * @рагат соттальной команда, у якій перевіряємо * @return true - якщо необов'язковий, інакше - false */ public static boolean paramIsOptional(String	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * @param с Клас команди * @peturn ключ команди */ @return ключ команди */	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override public void run() { final String userName = (String)
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * * @рагат с Клас команди * public static String getCommandKey(Class c) {	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override public void run() { final String userName = (String) context.get("username");
* повертає список усіх параметрів команди * @рагат с Клас команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * @рагат с Клас команди * @patum ключ команди */ public static String getCommandKey(Class c) { COMMAND t = (COMMAND)	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * выконує роботу команди */ @Override public void run() { final String userName = (String) context.get("username"); final String userPassword = (String)
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * * @рагат с Клас команди * @return ключ команди */ public static String getCommandKey(Class c) {	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override public void run() { final String userName = (String) context.get("username");
* повертає список усіх параметрів команди *	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * @рагат с Клас команди * @patum ключ команди */ public static String getCommandKey(Class c) { COMMAND t = (COMMAND)	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * виконує роботу команди */ @Override public void run() { final String userName = (String) context.get("username"); final String userPassword = (String) context.get("password");
* повертає список усіх параметрів команди * @рагат с Клає команди * @return список усіх параметрів команди */ public static PARAMETER[] getParameterList(Class c) {	} package com.oyster.core.controller.utils; import com.oyster.core.controller.annotation.COMMAND; import com.oyster.core.controller.annotation.CONTEXT; import com.oyster.core.controller.annotation.PARAMETER; /** * Допоміжний клас для роботи із аннотаціями команди */ public class ControllerAnnotationUtils { /** * повертає ключ команди * * @рагат с Клас команди * @return ключ команди */ public static String getCommandKey(Class c) {	String.class) }) public class LogInCommand extends AbstractCommand { public LogInCommand() { } /** * Конструктор * * @param context1 контекст команди */ public LogInCommand(Context context1) { setContext(context1); } /** * выконує роботу команди */ @Override public void run() { final String userName = (String) context.get("username"); final String userPassword = (String)

profiles = AppConst.DAO.select(Profile.class,	}	IF I WE G
new DAOFilter() {	}	public LoadHistoryCommand() {
@Override	}	}
public <t> boolean accept(T entity) {</t>	package com.oyster.core.controller.command;	At at
Profile p = (Profile) entity;		/**
return userName.equals(p.getFirstName() +	Mark.	* Конструктор
"-" + p.getSecondName())	/**	* 0
&&	* Абстпрактний клас, що реалізує базову команду,	* @param context1 контекст команди
userPassword.equals(p.getPassword());	* містить контекст, та екземпляр інтерфейсу	*/
}	Runnable,	<pre>public LoadHistoryCommand(Context context1) {</pre>
}) ;	* та методи його отримання	setContext(context1);
} catch (final DAOException e) {	*	}
e.printStackTrace();	* @author bamboo	
SwingUtilities.invokeLater(new Runnable() {	*/	/**
@Override		* виконує роботу команди
public void run() {	public abstract class AbstractCommand implements	*/
Utils.showErrorDialog(null,	Runnable {	@Override
Utils.makePretty(e.getMessage()));		public void run() {
}	protected Context context;	F ()
));	protected Context Context; protected Runnable onPostExecute;	String sqlQuery = (String) context.get("sqlQuery");
return;	protected Rumnable om ostExecute,	Sumg sqiQuery = (Sumg) context.get(sqiQuery),
icturii,	/**	List <history> histories = (List<history>)</history></history>
ı	,	
'C/ CI ' O O) (* повертає контекст команди	context.get("list");
if (profiles.size() == 0) {	*	
SwingUtilities.invokeLater(new Runnable() {	* @return контекст команди	try {
@Override	*/	List <history> list =</history>
public void run() {	<pre>public Context getContext() {</pre>	AppConst.DAO.select(History.class, sqlQuery);
Utils.showErrorDialog(null, "Помилка	return context;	for (History h : list) {
авторизації : неправильні логін чи пароль !");	}	histories.add(h);
}		}
});	/**	•
1//	* встановлює контекст команди	} catch (DAOException e) {
return;	*	e.printStackTrace();
l	* @param context контекст команди	e.prinistaekTrace(),
ı	*/	I
a in an an an	•	
final Profile profile = profiles.get(0);	<pre>public void setContext(Context context) {</pre>	
WorkerInfo wi = null;	this.context = context;	if (getOnPostExecute() != null) {
List <admin> admins = null;</admin>	}	
		SwingUtilities.invokeLater(getOnPostExecute());
try {	/**	}
admins = AppConst.DAO.select(Admin.class,	* повертає екземпляр інтерфейсу Runnable, метод	}
new DAOFilter() {	якого run()	}
@Override	* буде виконаний у UI потоці	package com.oyster.core.controller.command;
<pre>public <t> boolean accept(T entity) {</t></pre>	*	1
Admin a = (Admin) entity;	* @return екземпляр інтерфейсу Runnable	import com.oyster.app.AppConst;
return	*/	import com.oyster.app.model.History;
	•	
a.getProfileId().equals(profile.getId());	public Runnable getOnPostExecute() {	import com.oyster.app.model.IProfile;
,	return onPostExecute;	import
<pre>});</pre>	}	com.oyster.core.controller.annotation.COMMAND;
wi = AppConst.DAO.read(WorkerInfo.class,		import
admins.get(0).getWorkerInfoId());	/**	com.oyster.core.controller.annotation.CONTEXT;
} catch (final Exception e) {	* встановлює екземпляр інтерфейсу Runnable,	import
e.printStackTrace();	метод якого run()	com.oyster.core.controller.annotation.PARAMETER;
SwingUtilities.invokeLater(new Runnable() {	* буде виконаний у UI потоці	import com.oyster.dao.exception.DAOException;
@Override	*	
public void run() {	* @param runnable екземпляр інтерфейсу	import javax.swing.*;
Utils.showErrorDialog(null,	Runnable	import java.util.UUID;
Utils.makePretty(e.getMessage()));	*/	import juvaianie e 125,
ottis.maker retty(e.getiviessage())),	public void setOnPostExecute(Runnable runnable) {	/**
1).	onPostExecute = runnable;	'
});	onrostexecute = runnable;	* команда виконує видалення профілю, що
	}	передається її параметром
return;		*
}	}	*/
	package com.oyster.core.controller.command;	@COMMAND(key = "deleteIProfile")
if $(admins.size() == 0)$ {		@CONTEXT(list = {
SwingUtilities.invokeLater(new Runnable() {	import com.oyster.app.AppConst;	<pre>@PARAMETER(key = "profile", type =</pre>
@Override	import com.oyster.app.model.History;	IProfile.class)
public void run() {	import	})
Utils.showErrorDialog(null, "Помилка	com.oyster.core.controller.annotation.COMMAND;	public class DeleteIProfileCommand extends
авторизації : дані логін та пароль не відповідають	import	AbstractCommand {
адміністратору !");	com.oyster.core.controller.annotation.CONTEXT;	·
}	import	<pre>public DeleteIProfileCommand() {</pre>
});	com.oyster.core.controller.annotation.PARAMETER;)
17,	import com.oyster.dao.exception.DAOException;	J
ratura	import com.oyster.uao.exeeption.orroz.xeeption,	/**
return;	import invov owir ~ *-	•
}	import javax.swing.*;	* Конструктор
m 1 4 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2	import java.util.ArrayList;	* @param context1 контекст команди
final Admin admin = admins.get(0);	import java.util.List;	*/
admin.setProfile(profile);		<pre>public DeleteIProfileCommand(Context context1) {</pre>
admin.setWorkerInfo(wi);	/**	setContext(context1);
	 команда виконує завантаження історії, 	}
AppConst.setCurrentAdmin(admin);	виконуюючи SQL-запит, що їй передається	
•• "	*/	/**
SwingUtilities.invokeLater(new Runnable() {		* виконує роботу команди
@Override	@COMMAND(key = "loadHistory")	*/
public void rup() [· · · · · · · · · · · · · · · · · · ·	@Override
public void run() { MainForm form = now MainForm();	@CONTEXT(list = {	@Override
<pre>public void run() { MainForm form = new MainForm();</pre>	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type =</pre>	@Override public void run() {
MainForm form = new MainForm(); }	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type = String.class),</pre>	public void run() {
*	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type = String.class), @PARAMETER(key = "list", type =</pre>	<pre>public void run() { IProfile currentPerson = (IProfile)</pre>
MainForm form = new MainForm(); } });	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type = String.class), @PARAMETER(key = "list", type = ArrayList.class)</pre>	public void run() {
MainForm form = new MainForm(); }	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type = String.class), @PARAMETER(key = "list", type = ArrayList.class) })</pre>	<pre>public void run() { IProfile currentPerson = (IProfile)</pre>
MainForm form = new MainForm(); } });	<pre>@CONTEXT(list = { @PARAMETER(key = "sqlQuery", type = String.class), @PARAMETER(key = "list", type = ArrayList.class)</pre>	<pre>public void run() { IProfile currentPerson = (IProfile)</pre>

```
AppConst.DAO.delete(currentPerson.getProfile());
                                                                                                                                            final Profile profile = profiles.get(0);
       AppConst.DAO.delete(currentPerson);
                                                                   import com.oyster.app.AppConst;
                                                                                                                                            WorkerInfo wi = null;
                                                                   import com.oyster.app.model.Admin;
                                                                                                                                            List<Admin> admins = null;
       History h = new History(
                                                                   import com.oyster.app.model.Profile;
           UUID.randomUUID().
                                                                   import\ com. oyster. app. model. Worker Info;
                                                                                                                                              admins = AppConst.DAO.select(Admin.class,
                                                                   import
AppConst.getCurrentAdmin().getProfileId(),
                                                                   com.oyster.core.controller.annotation.COMMAND;
                                                                                                                                       new DAOFilter() {
            "Видалив користувача "
                                                                   import
                                                                                                                                                 @Override
                                                                   \stackrel{-}{\text{com.oyster.core.controller.annotation.CONTEXT}};
currentPerson.getProfile().toString()
                                                                                                                                                public <T> boolean accept(T entity) {
                                                                   import
                                                                                                                                                  Admin a = (Admin) entity;
       AppConst.DAO.insert(h);
                                                                   com.oyster.core.controller.annotation.PARAMETER;
                                                                                                                                                  return
                                                                                                                                       a.getProfileId().equals(profile.getId());
                                                                   import com.ovster.dao.DAOFilter:
    } catch (DAOException e) {
                                                                   import com.oyster.dao.exception.DAOException;
                                                                                                                                                }
      e.printStackTrace();
                                                                   import com.oyster.ui.MainForm;
                                                                                                                                              });
                                                                                                                                              wi = AppConst.DAO.read(WorkerInfo.class,
                                                                   import com.oyster.ui.Utils;
                                                                                                                                       admins.get(0).getWorkerInfoId());
    if (getOnPostExecute() != null) {
                                                                                                                                           } catch (final Exception e) {
                                                                   import javax.swing.*:
                                                                   import java.util.List;
                                                                                                                                              e.printStackTrace();
SwingUtilities.invokeLater(getOnPostExecute());
                                                                                                                                              SwingUtilities.invokeLater(new Runnable() {
                                                                                                                                                 @Override
    }
                                                                                                                                                public void run() {
                                                                    * команда виконує авторизацію користувача із
                                                                   логіном та паролем, що ії передаються
                                                                                                                                                  Utils.showErrorDialog(null,
                                                                                                                                       Utils.makePretty(e.getMessage()));
package com.oyster.core.controller.command;
                                                                    @COMMAND(key = "logIn")
                                                                                                                                                }
import com.oyster.app.AppConst;
                                                                    @CONTEXT(list = {
                                                                                                                                              });
import com.oyster.app.model.Classes;
                                                                        @PARAMETER(key = "username", type =
                                                                    String.class),
import
                                                                                                                                              return;
                                                                        @PARAMETER(key = "password", type =
com.oyster.core.controller.annotation.COMMAND;
import
                                                                    String.class)
com.oyster.core.controller.annotation.CONTEXT;
                                                                                                                                            if (admins.size() == 0) {
                                                                   public class LogInCommand extends AbstractCommand
                                                                                                                                              SwingUtilities.invokeLater(new Runnable() {
import
com.oyster.core.controller.annotation.PARAMETER;
                                                                                                                                                 @Override
import com.oyster.dao.exception.DAOException;
                                                                                                                                                public void run() {
                                                                                                                                                  Utils.showErrorDialog(null, "Помилка
                                                                      public LogInCommand() {
                                                                                                                                       авторизації : дані логін та пароль не відповідають
import javax.swing.*;
import java.util.ArrayList;
                                                                                                                                       адміністратору !");
import java.util.List;
                                                                                                                                                }
                                                                      * Конструктор
                                                                                                                                              });
* команда виконує завантаження розкладу,
                                                                       * @param context1 контекст команди
                                                                                                                                              return;
виконуюючи SOL-запит, що їй передається
                                                                                                                                            }
                                                                     public LogInCommand(Context context1) {
@COMMAND(key = "loadSchedule")
                                                                                                                                           final\ Admin\ admin=admins.get(0);
                                                                        setContext(context1);
@CONTEXT(list = {
                                                                                                                                            admin.setProfile(profile);
    @PARAMETER(key = "sqlQuery", type =
                                                                                                                                            admin.setWorkerInfo(wi):
String.class),
    @PARAMETER(key = "list", type =
                                                                      * виконує роботу команди
                                                                                                                                            AppConst.setCurrentAdmin(admin);
ArrayList.class)
                                                                      @Override
                                                                                                                                            SwingUtilities.invokeLater(new Runnable() {
public class LoadScheduleCommand extends
                                                                     public void run() {
                                                                                                                                              @Override
AbstractCommand {
                                                                                                                                              public void run() {
                                                                        final String userName = (String)
                                                                                                                                                MainForm form = new MainForm();
  public LoadScheduleCommand() {
                                                                   context.get("username");
                                                                        final String userPassword = (String)
                                                                                                                                            });
                                                                    context.get("password");
                                                                                                                                           if \ (getOnPostExecute() \ != null) \ \{\\
   * Конструктор
                                                                        List<Profile> profiles = null;
   * @param context1 контекст команди
                                                                                                                                       SwingUtilities.invokeLater(getOnPostExecute());
                                                                        try {
                                                                          profiles = AppConst.DAO.select(Profile.class,
                                                                   new DAOFilter() {
  public LoadScheduleCommand(Context context1) {
                                                                                                                                         }
    setContext(context1);
                                                                             @Override
                                                                             public <T> boolean accept(T entity) {
                                                                                                                                       package com.oyster.core.controller.command.register;
                                                                               Profile p = (Profile) entity;
                                                                               return userName.equals(p.getFirstName() +
                                                                                                                                       import com.oyster.app.AppConst;
   * виконує роботу команди
                                                                    "-" + p.getSecondName())
                                                                                                                                       import com.oyster.app.model.History;
                                                                                                                                       import com.oyster.app.model.Profile;
  @Override
                                                                   user Password.equals (p.get Password ());\\
                                                                                                                                       import com.oyster.app.model.Teacher;
  public void run() {
                                                                                                                                       import com.oyster.app.model.WorkerInfo;
                                                                            }
                                                                          });
                                                                                                                                       import
                                                                        } catch (final DAOException e) {
    String sqlOuery = (String) context.get("sqlOuery");
                                                                                                                                       com.ovster.core.controller.annotation.COMMAND:
                                                                          e.printStackTrace();
                                                                                                                                       import
    List<Classes> classesList = (List<Classes>)
                                                                                                                                       com.oyster.core.controller.annotation.CONTEXT;
                                                                          SwingUtilities.invokeLater(new Runnable() {
context.get("list");
                                                                             @Override
                                                                             public void run() {
                                                                                                                                       com.ovster.core.controller.annotation.PARAMETER:
                                                                               Utils.showErrorDialog(null,
    try {
                                                                                                                                       import
      List<Classes> list =
                                                                   Utils.makePretty(e.getMessage()));
                                                                                                                                       com.oyster.core.controller.command.AbstractCommand;
AppConst.DAO.select(Classes.class, sqlQuery);
                                                                                                                                       import com.oyster.core.controller.command.Context;
                                                                            }
       for (Classes c : list) {
                                                                                                                                       import com.oyster.ui.Utils;
         classesList.add(c);
                                                                          return;
                                                                                                                                       import javax.swing.*;
                                                                                                                                       import java.util.UUID;
    } catch (DAOException e) {
                                                                        if (profiles.size() == 0) {
      e.printStackTrace();
                                                                          SwingUtilities.invokeLater(new Runnable() {
                                                                             @Override
                                                                                                                                        * команда виконує реєстрацію викладача у системі,
                                                                             public void run() {
                                                                                                                                       інформацію про якого їй передається у контексті
                                                                               Utils.showErrorDialog(null, "Помилка
                                                                                                                                        * @author bamboo
    if (getOnPostExecute() != null) {
                                                                   авторизації : неправильні логін чи пароль !");
                                                                            }
                                                                                                                                        @COMMAND(key = "registerTeacher")
SwingUtilities.invokeLater(getOnPostExecute());
                                                                          });
                                                                                                                                       @CONTEXT(list = {
                                                                          return;
                                                                                                                                            @PARAMETER(key = "name", type =
                                                                        }
                                                                                                                                       String.class),
```

package com.ovster.core.controller.command:

```
@PARAMETER(key = "surname", type =
                                                                  import
                                                                                                                                     import javax.swing.*:
String.class),
                                                                  com.oyster.core.controller.annotation.COMMAND;
                                                                                                                                     import java.util.UUID;
    @PARAMETER(key = "birthday", type =
                                                                  import
Long.class, optional = true),
                                                                  com.oyster.core.controller.annotation.CONTEXT;
    @PARAMETER(key = "position", type =
                                                                                                                                      * команда виконує реєстрацію студента у системі,
                                                                  import
String.class).
                                                                  com.ovster.core.controller.annotation.PARAMETER:
                                                                                                                                     інформацію про якого їй передається у контексті
                                                                                                                                      * @author bamboo
                                                                  import
    @PARAMETER(key = "salary", type =
                                                                  com. oyster. core. controller. command. Abstract Command;\\
Integer.class),
                                                                  import com.oyster.core.controller.command.Context;
    @PARAMETER(key = "password", type =
                                                                  import com.oyster.ui.Utils;
                                                                                                                                     @COMMAND(key = "registerStudent")
String.class),
                                                                                                                                     @CONTEXT(list = {
    @PARAMETER(key = "dateHired", type =
                                                                  import javax.swing.*
                                                                                                                                          @PARAMETER(key = "name", type =
Long.class, optional = true)
                                                                  import java.util.UUID;
                                                                                                                                     String.class).
                                                                                                                                          @PARAMETER(key = "surname", type =
})
public class RegisterTeacherCommand extends
                                                                                                                                      String.class),
AbstractCommand {
                                                                                                                                          @PARAMETER(key = "birthday", type =
                                                                   * команда виконує реєстрацію факультету у системі,
                                                                  інформацію про якого їй передається у контексті
                                                                                                                                     Long.class, optional = true),
  public RegisterTeacherCommand() {
                                                                                                                                          @PARAMETER(key = "faculty", type =
                                                                   * @author bamboo
                                                                                                                                     String.class),
                                                                                                                                          @PARAMETER(key = "group", type =
                                                                                                                                      String.class),
  * Конструктор
                                                                   @COMMAND(key = "registerFaculty")
                                                                                                                                          @PARAMETER(key = "course", type =
  * @param context1 контекст команди
                                                                   @CONTEXT(list = {
                                                                                                                                     Integer.class),
                                                                       @PARAMETER(key = "name", \, type = \\
                                                                                                                                          @PARAMETER(key = "password", type =
  public RegisterTeacherCommand(Context context1) {
                                                                  String.class)
                                                                                                                                     String.class),
                                                                                                                                          @PARAMETER(key = "bookNum", type =
    setContext(context1);
                                                                  public class RegisterFacultyCommand extends
                                                                                                                                     Integer.class)
                                                                   AbstractCommand {
                                                                                                                                     public class RegisterStudentCommand extends
  * виконує роботу команди
                                                                     public RegisterFacultyCommand() {
                                                                                                                                     AbstractCommand {
  @Override
                                                                                                                                        public RegisterStudentCommand() {
  public void run() {
                                                                     * Конструктор
                                                                     * @param context1 контекст команди
                                                                                                                                        * Конструктор
    Profile pTeacher = new Profile(
                                                                    public RegisterFacultyCommand(Context context1) {
         UUID.randomUUID(),
                                                                                                                                        * @param context1 контекст команди
                                                                       setContext(context1);
         (String) context.get("name"),
         (String) context.get("surname"),
                                                                                                                                        public RegisterStudentCommand(Context context1) {
         (String) context.get("password"),
                                                                                                                                          setContext(context1);
         (Long)\ context.get("birthday")
                                                                     * виконує роботу команди
    ):
                                                                     @Override
    WorkerInfo wTeacher = new WorkerInfo(
                                                                    public void run() {
                                                                                                                                        * виконує роботу команди
         UUID.randomUUID(),
         (String) context.get("position"),
                                                                                                                                        @Override
                                                                       Faculty fac = new Faculty(UUID.randomUUID(),
         (Integer) context.get("salary"),
                                                                                                                                       public void run() {
         (Long) context.get("dateHired")
                                                                  ((String) context.get("name")).toUpperCase());
    );
                                                                       try {
                                                                          AppConst.DAO.insert(fac);
                                                                                                                                          Profile pStudent = new Profile(
    Teacher teacher = new Teacher(
                                                                         History h = new History(
                                                                                                                                              UUID.randomUUID(),
         UUID.randomUUID(),
                                                                             UUID.randomUUID(),
                                                                                                                                              (String) context.get("name"),
         pTeacher.getId(),
                                                                                                                                              (String) context.get("surname"),
         wTeacher.getId()
                                                                  AppConst.getCurrentAdmin().getProfileId(),
                                                                                                                                              (String) context.get("password"),
                                                                              "Створив новий факультет "
                                                                                                                                              (Long) context.get("birthday")
    );
                                                                   fac.getName()
      AppConst.DAO.insert(pTeacher);
                                                                          AppConst.DAO.insert(h);
                                                                                                                                          final String groupName = (String)
      AppConst.DAO.insert(wTeacher);
                                                                       } catch (final Exception e) {
                                                                                                                                     context.get("group");
      AppConst.DAO.insert(teacher);
                                                                         e.printStackTrace();
                                                                                                                                          java.util.List<Group> groups = null;
                                                                          SwingUtilities.invokeLater(new Runnable() {
      History h = new History(
                                                                                                                                            groups = AppConst.DAO.select(Group.class,
                                                                            @Override
                                                                            public void run() {
           UUID.randomUUID(),
                                                                                                                                     new DAOFilter() {
                                                                             Utils.showErrorDialog(null,
                                                                                                                                              @Override
                                                                  Utils.makePretty("Помилка створення факультету:
AppConst.getCurrentAdmin().getProfileId(),
                                                                                                                                              public < T > boolean \ accept(T \ entity) \ \{
           "Додав викладача " + pTeacher.toString()
                                                                  \n" + e.getMessage()));
                                                                                                                                                 Group g = (Group) entity;
                                                                                                                                                 return\ g.getName().equals(groupName);
      AppConst.DAO.insert(h);
                                                                         });
                                                                       }
                                                                                                                                            });
    } catch (final Exception e) {
                                                                                                                                            for (Group gg : groups) {
                                                                       if (getOnPostExecute() != null) {
      e.printStackTrace();
                                                                                                                                              gg.setFaculty((Faculty)
                                                                                                                                     AppConst.DAO.read(Faculty.class, gg.getFacultyId()));
      SwingUtilities.invokeLater(new Runnable() {
                                                                  SwingUtilities.invokeLater(getOnPostExecute());\\
                                                                                                                                          } catch (final DAOException e) {
         @Override
         public void run() {
                                                                    }
                                                                                                                                            e.printStackTrace();
           Utils.showErrorDialog(null,
                                                                  }
                                                                                                                                            SwingUtilities.invokeLater(new Runnable() {
               Utils.makePretty("Помилка
                                                                  package com.oyster.core.controller.command.register;
                                                                                                                                              @Override
створення викладача : \n" + e.getMessage()));
                                                                                                                                              public void run() {
                                                                                                                                                 Utils.showErrorDialog(null,
                                                                  import com.oyster.app.AppConst;
                                                                  import com.oyster.app.model.*;
                                                                                                                                     Utils.makePretty("Помилка зчитування групи : \n" +
      });
    }
                                                                  import
                                                                                                                                     e.getMessage()));
                                                                  com.oyster.core.controller.annotation.COMMAND;
                                                                                                                                              }
    if (getOnPostExecute() != null) {
                                                                                                                                            });
                                                                  import
                                                                  com.oyster.core.controller.annotation.CONTEXT;
SwingUtilities.invokeLater(getOnPostExecute());
                                                                  import
                                                                   com.oyster.core.controller.annotation.PARAMETER;
                                                                                                                                          if (groups.size() == 0) {
                                                                  import
                                                                                                                                            Utils.showErrorDialog(null, "Немає такої
  }
                                                                  com.ovster.core.controller.command.AbstractCommand:
                                                                                                                                     групи!");
package com.oyster.core.controller.command.register;
                                                                  import com.oyster.core.controller.command.Context;
                                                                                                                                            return;
                                                                  import com.oyster.dao.DAOFilter;
import com.oyster.app.AppConst;
                                                                   import com.oyster.dao.exception.DAOException;
import com.oyster.app.model.Faculty;
                                                                  import com.oyster.ui.Utils;
                                                                                                                                          final Group group = groups.get(0);
import com.oyster.app.model.History;
```

```
if \ (!group.getFaculty().getName().equals((String)
                                                                                                                                        public RegisterAdminCommand(Context context1) {
context.get("faculty"))) {

    виконує роботу команди

       SwingUtilities.invokeLater(new Runnable() {
                                                                                                                                          setContext(context1);
         @Override
                                                                     @Override
         public void run() {
                                                                     public void run() {
           Utils.showErrorDialog(null, "Немає такої
                                                                                                                                         * виконує роботу команди
групи на цьому факультеті!");
                                                                       Subject\ subject = new
         }
                                                                   Subject(UUID.randomUUID(), ((String)
                                                                                                                                        @Override
       return;
                                                                   context.get("name")).toUpperCase());
                                                                                                                                        public void run() {
                                                                          AppConst.DAO.insert(subject);
                                                                                                                                          Profile pAdmin = new Profile(
                                                                         History h = new History(
                                                                              UUID.randomUUID(),
    Student student = new Student(
                                                                                                                                               UUID.randomUUID(),
         UUID.randomUUID(),
                                                                                                                                               (String) context.get("name"),
                                                                   AppConst.getCurrentAdmin().getProfileId(),
         pStudent.getId(),
                                                                                                                                               (String) context.get("surname"),
          groups.get(0).getId(),
                                                                              "Створив новий предмет "
                                                                                                                                               (String) context.get("password"),
                                                                   subject.getName()
         (Integer) context.get("course"),
                                                                                                                                               (Long) context.get("birthday")
         (Integer) context.get("bookNum")
                                                                          AppConst.DAO.insert(h);
                                                                        } catch (final Exception e) {
                                                                                                                                          WorkerInfo wAdmin = new WorkerInfo(
                                                                          e.printStackTrace();
                                                                                                                                               UUID.randomUUID(),
       AppConst.DAO.insert(pStudent);
                                                                          SwingUtilities.invokeLater(new Runnable() {
                                                                                                                                               (String) context.get("position"),
       AppConst.DAO.insert(student);
                                                                            @Override
                                                                                                                                               (Integer) context.get("salary")
                                                                                                                                               (Long) context.get("dateHired")
                                                                            public void run() {
       History h = new History(
                                                                              Utils.showErrorDialog(null,
                                                                   Utils.makePretty("Помилка створення предмету : \n"
           UUID.randomUUID(),
                                                                   + e.getMessage()));
                                                                                                                                          Admin admin = new Admin(
AppConst.getCurrentAdmin().getProfileId(),
                                                                                                                                               UUID.randomUUID(),
            "Додав студента" + pStudent.toString()
                                                                         });
                                                                                                                                               pAdmin.getId(),
                                                                                                                                               wAdmin.getId()
       AppConst.DAO.insert(h);
     } catch (final DAOException e) {
                                                                       if (getOnPostExecute() != null) {
       e.printStackTrace();
                                                                                                                                            AppConst.DAO.insert(pAdmin);
                                                                   SwingUtilities.invokeLater(getOnPostExecute());
       SwingUtilities.invokeLater(new Runnable() {
                                                                                                                                             AppConst.DAO.insert(wAdmin);
         @Override
                                                                     }
                                                                                                                                            AppConst.DAO.insert(admin);
         public void run() {
           Utils.showErrorDialog(null,
                                                                  package com.oyster.core.controller.command.register;
                                                                                                                                            History h = new History(
Utils.makePretty("Помилка створення студента : \n" +
                                                                                                                                                 UUID.randomUUID(),
e.getMessage()));
                                                                  import\ com. oyster. app. App Const;
                                                                                                                                      AppConst.getCurrentAdmin().getProfileId().
                                                                  import com.oyster.app.model.Admin;
                                                                   import com.oyster.app.model.History;
                                                                                                                                                  "Додав адміністратора " +
       });
                                                                                                                                      pAdmin.toString()
                                                                   import com.oyster.app.model.Profile;
                                                                   import com.oyster.app.model.WorkerInfo;
    if (getOnPostExecute() != null) {
                                                                                                                                             AppConst.DAO.insert(h);
                                                                  import
SwingUtilities.invokeLater(getOnPostExecute());
                                                                  com.ovster.core.controller.annotation.COMMAND:
                                                                                                                                          } catch (final Exception e) {
    }
                                                                  import
                                                                   com.oyster.core.controller.annotation.CONTEXT;
                                                                                                                                            e.printStackTrace();
  }
                                                                                                                                             SwingUtilities.invokeLater(new Runnable() {
package com.oyster.core.controller.command.register;
                                                                   com. oyster. core. controller. annotation. PARAMETER;\\
                                                                                                                                               @Override
                                                                                                                                               public void run() {
                                                                                                                                                 Utils.showErrorDialog(null,
import com.oyster.app.AppConst;
                                                                  com.ovster.core.controller.command.AbstractCommand:
import com.oyster.app.model.History;
                                                                   import com.oyster.core.controller.command.Context;
                                                                                                                                                      Utils.makePretty("Помилка
import com.oyster.app.model.Subject;
                                                                   import com.oyster.ui.Utils;
                                                                                                                                      створення адміністратора : \n" + e.getMessage()));
com.oyster.core.controller.annotation.COMMAND;
                                                                   import javax.swing.*;
                                                                                                                                             });
                                                                  import java.util.UUID;
import
com.ovster.core.controller.annotation.CONTEXT:
import
com.oyster.core.controller.annotation.PARAMETER;
                                                                   * команда виконує реєстрацію адміністратора у
                                                                                                                                          if (getOnPostExecute() != null) {
                                                                   системі, інформацію про якого їй передається у
com.oyster.core.controller.command.AbstractCommand;
                                                                   контексті
import com.oyster.core.controller.command.Context;
                                                                    * @author bamboo
                                                                                                                                      SwingUtilities.invokeLater(getOnPostExecute());
import com.oyster.ui.Utils;
                                                                                                                                        }
import javax.swing.*;
                                                                   @COMMAND(key = "registerAdmin")
import java.util.UUID;
                                                                   @CONTEXT(list = {
                                                                                                                                     package com.oyster.core.controller.command.register;
                                                                       @PARAMETER(key = "name", type =
* команда виконує реєстрацію предмету у системі,
                                                                   String.class),
                                                                                                                                      import com.oyster.app.AppConst;
                                                                        @PARAMETER(key = "surname", type =
інформацію про якого їй передається у контексті
                                                                                                                                      import com.oyster.app.model.History;
  @author bamboo
                                                                                                                                      import com.oyster.app.model.Profile;
                                                                   String.class),
                                                                        @PARAMETER(key = "birthday", type =
                                                                                                                                      import com.oyster.app.model.Teacher;
                                                                   Long.class, optional = true),
                                                                                                                                      import com.oyster.app.model.WorkerInfo;
@COMMAND(key = "registerSubject")
                                                                       @PARAMETER(key = "position", type =
                                                                                                                                      import
@CONTEXT(list = {
                                                                                                                                      com.oyster.core.controller.annotation.COMMAND;
                                                                  String.class),
    @PARAMETER(key = "name", \, type =
                                                                                                                                      import
String.class)
                                                                        @PARAMETER(key = "salary", type =
                                                                                                                                      com.oyster.core.controller.annotation.CONTEXT;
                                                                   Integer.class),
public class RegisterSubjectCommand extends
                                                                       @PARAMETER(key = "password", type =
                                                                                                                                      com. oyster. core. controller. annotation. PARAMETER;\\
AbstractCommand {
                                                                   String.class),
                                                                        @PARAMETER(key = "dateHired", type =
                                                                                                                                      com.ovster.core.controller.command.AbstractCommand:
  public RegisterSubjectCommand() {
                                                                   Long.class, optional = true)
                                                                                                                                      import com.oyster.core.controller.command.Context;
                                                                                                                                      import com.oyster.ui.Utils;
                                                                  public class RegisterAdminCommand extends
                                                                   AbstractCommand {
                                                                                                                                      import javax.swing.*;
   * Конструктор
                                                                                                                                     import java.util.UUID;
   * @param context1 контекст команди
                                                                     public RegisterAdminCommand() {
  public RegisterSubjectCommand(Context context1) {
                                                                                                                                      * команда виконує реєстрацію викладача у системі,
                                                                                                                                      інформацію про якого їй передається у контексті
    setContext(context1);
                                                                      * Конструктор
                                                                                                                                      * @author bamboo
                                                                      * @param context1 контекст команди
```

```
@COMMAND(key = "registerTeacher")
                                                                                                                                                                                           * @param <T> тип сутності
@CONTEXT(list = {
                                                                                                                                                                                          * @throws DAOException
                                                                                           import com.oyster.dao.annotation.utils.ValueConverter;
      @PARAMETER(key = "name", type =
String.class),
                                                                                           import java.util.UUID;
                                                                                                                                                                                          public <T> void delete(T instance) throws
      @PARAMETER(key = "surname", type =
                                                                                                                                                                                      DAOException;
String.class),
      @PARAMETER(key = "birthday", type =
                                                                                            * конвертує сутність (String <==> UUID)
Long.class, optional = true),
                                                                                                                                                                                          * виконує пошук сутностей у постійному
      @PARAMETER(key = "position", type =
                                                                                           public class UUIDConverter implements
                                                                                                                                                                                      хранилищі
                                                                                            ValueConverter {
String.class),
                                                                                                                                                                                          * @param entityClass клас сутності
      @PARAMETER(key = "salary", type =
                                                                                                                                                                                          * @param filter фільтр
Integer.class),
                                                                                               * перетворює UUID в String
                                                                                                                                                                                          * @param <T>
                                                                                                                                                                                                                    тип сутності
       @PARAMETER(key = "password", type =
                                                                                                                                                                                          * @return список знайдених сутностей, що
                                                                                                                                                                                       задовільняють даному фільтру
String.class),
                                                                                               * @param value параметр для конвертації
      @PARAMETER(key = "dateHired", type =
                                                                                               * @param <T> тип параметру
                                                                                                                                                                                          * @throws DAOException
                                                                                               * @return параметр як стрічку
Long.class, optional = true)
                                                                                                                                                                                          public <T> List<T> select(Class entityClass,
public class RegisterTeacherCommand extends
                                                                                                                                                                                      DAOFilter filter) throws DAOException;
                                                                                              public <T> String toString(T value) {
AbstractCommand {
                                                                                                 if (value == null) return null;
                                                                                                 return "\"" + ((Object) value).toString() + "\"";
   public RegisterTeacherCommand() {
                                                                                                                                                                                          * виконує пошук сутностей у базі даних
                                                                                                                                                                                          * @param entityClass клас сутності
                                                                                                                                                                                          * @param SQLString SQL-запит до бази даних
    * Конструктор
                                                                                               * перетворює String в UUID
                                                                                                                                                                                          * @param <T>
                                                                                                                                                                                                                    тип сутності
    * @param context1 контекст команди
                                                                                                                                                                                          * @return список знайдених сутностей
                                                                                               * @param str String для конвертування
                                                                                                                                                                                          * @throws DAOException
                                                                                               * @рагат <T> тип параметру
   public RegisterTeacherCommand(Context context1) {
                                                                                               * @return стрічку як параметр
                                                                                                                                                                                         public <T> List<T> select(Class entityClass, String
      setContext(context1);
                                                                                                                                                                                       SQLString) throws DAOException;
                                                                                               @Override
                                                                                              public <T> T toValue(String str) {
                                                                                                                                                                                      package com.oyster.dao.impl;
    * виконує роботу команди
                                                                                                 if (str == null) return null;
                                                                                                 if (str.equals("null")) return null;
                                                                                                                                                                                      import com.oyster.dao.CRUDInterface;
   @Override
                                                                                                 str = str.substring(1, str.length() - 1);
                                                                                                                                                                                      import com.oyster.dao.DAOFilter;
                                                                                                 return (T) UUID.fromString(str);
   public void run() {
                                                                                                                                                                                      import com.oyster.dao.annotation.Stored;
                                                                                                                                                                                      import
                                                                                                                                                                                      com.oyster.dao.annotation.utils.DAOAnnotationUtils;
     Profile pTeacher = new Profile(
UUID.randomUUID(),
                                                                                           package com.oyster.dao;
                                                                                                                                                                                      import com.oyster.dao.exception.DAOException;
                                                                                                                                                                                      import
             (String) context.get("name"),
                                                                                           import com.oyster.dao.exception.DAOException;
                                                                                                                                                                                      org.springframework.context.ApplicationContext;
             (String) context.get("surname"),
                                                                                                                                                                                      import
             (String) context.get("password"),
                                                                                           import java.util.List;
                                                                                                                                                                                      org.spring framework.dao. Empty Result Data Access Excest Control of Contro
                                                                                                                                                                                      ption:
             (Long) context.get("birthday")
                                                                                           import java.util.UUID;
      );
                                                                                                                                                                                      import
                                                                                                                                                                                      org.spring framework.jdbc.core.support.Jdbc Dao Support\\
      WorkerInfo wTeacher = new WorkerInfo(
                                                                                            * визначає CRUD-інтерфейс для базових операцій із
             UUID.randomUUID(),
                                                                                            постійним місцем збереження
             (String) context.get("position"),
                                                                                                                                                                                      import java.lang.reflect.Field;
             (Integer) context.get("salary"),
                                                                                           public interface CRUDInterface {
                                                                                                                                                                                      import java.util.*;
            (Long) context.get("dateHired")
      );
                                                                                               * виконує базову операцію вставлення для даної
                                                                                                                                                                                       * Клас реалізує CRUDInterface для доступу до
      Teacher teacher = new Teacher(
                                                                                           сутності
                                                                                                                                                                                      локальної бази даних на комп'ютері
             UUID.randomUUID(),
                                                                                               * @param instance сутність
                                                                                                                                                                                       * @author bamboo
             pTeacher.getId(),
             wTeacher.getId()
                                                                                               * @param <T> тип сутності
                                                                                               * @return саму сутність
                                                                                                                                                                                      public class DAOCRUDJdbc extends JdbcDaoSupport
      );
                                                                                                                                                                                      implements CRUDInterface {
                                                                                               * @throws DAOException
         AppConst.DAO.insert(pTeacher);
                                                                                              public <T> T insert(T instance) throws
         AppConst.DAO.insert(wTeacher);
                                                                                                                                                                                          * отримує екземпляр JDBC dao
                                                                                           DAOException;
         AppConst.DAO.insert(teacher);
                                                                                                                                                                                          * @рагат context контекст додатку
         History h = new History(
                                                                                               * виконує базову операцію зчитування для даної
                                                                                                                                                                                          * @return екземпляр Jdbc dao
               UUID.randomUUID(),
                                                                                           сутності
                                                                                                                                                                                          public static DAOCRUDJdbc
AppConst.getCurrentAdmin().getProfileId(),
                                                                                               * @param entityClass клас сутності
                                                                                                                                                                                      getInstance(ApplicationContext\ context)\ \{
                                                                                                                                                                                            return (DAOCRUDJdbc)
                "Додав викладача " + pTeacher.toString()
                                                                                               * @param id
                                                                                                                       ключ сутності
                                                                                               * @param <T>
                                                                                                                                                                                      context.getBean("DAOJdbc");
                                                                                                                        тип сутності
         AppConst.DAO.insert(h);
                                                                                               * @return саму сутність
                                                                                               * @throws DAOException
      } catch (final Exception e) {
                                                                                              public <T> T read(Class entityClass, UUID id)
                                                                                                                                                                                          * додає екземпляр нової сутності у сховище даних
         e.printStackTrace();
                                                                                           throws DAOException;
         SwingUtilities.invokeLater(new Runnable() {
                                                                                                                                                                                          * @param instance сутність для вставки
                                                                                                                                                                                          * @рагат <T> тип сутності
                                                                                               * виконує базову операцію оновлення для даної
             public void run() {
                                                                                                                                                                                          * @return саму сутність
               Utils.showErrorDialog(null,
                                                                                           сутності
                                                                                                                                                                                          * @throws DAOException
                     Utils.makePretty("Помилка
створення викладача : \n" + e.getMessage()));
                                                                                               * @param instance сутність
                                                                                                                                                                                          @Override
                                                                                               * @рагат <T> тип сутності
                                                                                                                                                                                          public <T> T insert(T instance) throws
                                                                                               * @throws DAOException
         });
                                                                                                                                                                                      DAOException {
                                                                                                                                                                                            return\ performInsertOrReplace (instance,
      }
                                                                                              public <T> void update(T instance) throws
                                                                                                                                                                                       "INSERT"):
      if (getOnPostExecute() != null) {
                                                                                           DAOException;
SwingUtilities.invokeLater(getOnPostExecute());
                                                                                                                                                                                         private <T> T performInsertOrReplace(T instance,
                                                                                               * виконує базову операцію видалення для даної
                                                                                                                                                                                      String keyWord) {
                                                                                           сутності
```

package com.ovster.dao.annotation.utils.converter:

* @param instance сутність

```
HashMap<String, String> mapStrStr =
                                                                                                                                                if (sql == null || sql.trim(),length() == 0) {
                                                                                                                                                  sql = "SELECT * FROM " +
                                                                          Field primaryKeyField =
(HashMap<String, String>)
                                                                                                                                           DAOAnnotationUtils.getStorageName(entityClass) +
                                                                     DAOAnnotation Utils.get Primary Key (instance.get Class (\\
DAOAnnotation Utils.get Converted Stored Fields (instance) \\
                                                                     ));
                                                                          String sql = "UPDATE " +
                                                                                                                                               }
                                                                     DAOAnnotation Utils.getStorageName (instance.getClass\\
     StringBuilder sb = new StringBuilder();
                                                                                                                                                if (filter == null) {
                                                                     ())
     StringBuilder q = new StringBuilder();
                                                                                 " SET " + sb.substring(1) + " WHERE "
                                                                                                                                                  filter = new DAOFilter() {
     for (String key: mapStrStr.keySet()) {
    sb.append(", " + key);
    q.append(", " + mapStrStr.get(key));
                                                                                                                                                     @Override
                                                                     primary Key Field.get Annotation (Stored.class).name ()\\
                                                                                                                                                    public <T> boolean accept(T entity) {
                                                                                                                                                      return true;
                                                                     DAOAnnotation Utils.get String Value (instance,\\
                                                                     primaryKeyField) + ":":
                                                                                                                                                 }:
     String sql = keyWord + " INTO " +
DAOAnnotation Utils.getStorageName (instance.getClass)\\
                                                                          System.out.println(sql);
                                                                                                                                               List<Map<String, Object>> list =
         + " (" + sb.substring(1) + ")" + "VALUES" + "
                                                                          getJdbcTemplate().update(sql);
                                                                                                                                           getJdbcTemplate().queryForList(sql);
(" + q.substring(1) + ");";
     System.out.println(sql);
                                                                                                                                                for (int i = 0; i < list.size(); i++) {
     getJdbcTemplate().update(sql);
                                                                        * видаляє сутність із сховища даних
                                                                                                                                                  System.out.println(list.get(i));\\
     return instance:
                                                                        * @param instance сутність для видалення
                                                                         * @ param <T> тип сутності
                                                                                                                                                ArrayList<T> arrayList = new ArrayList<T>();
                                                                         * @throws DAOException
                                                                                                                                                for (int i = 0; i < list.size(); i++) {
                                                                        @Override
   * замінює екземпляр сутності у сховищі даних
                                                                        public <T> void delete(T instance) throws
                                                                                                                                           DAOAnnotationUtils.mapToEntity(entityClass,
                                                                                                                                           list.get(i));
   * @param instance сутність для вставки
                                                                      DAOException {
                                                                          Field primaryKeyField =
   * @param <T> тип сутності
                                                                                                                                                  if (filter.accept(instance)) {
    @return саму сутність
                                                                     DAOAnnotationUtils.getPrimaryKey(instance.getClass(
                                                                                                                                                    arrayList.add(instance);
   * @throws DAOException
                                                                                                                                                 }
                                                                     ));
                                                                          String sql = "DELETE FROM " +
  public <\!\!T\!\!> T\; replace(T\; instance)\; throws
                                                                     DAOAnnotation Utils.getStorageName (instance.getClass\\
                                                                                                                                                return arrayList;
DAOException {
                                                                     ())
                                                                                                                                             }
    return performInsertOrReplace(instance,
                                                                               + " WHERE "
"REPLACE");
                                                                                                                                          package com.oyster.dao.annotation;
  }
                                                                     primary Key Field.get Annotation (Stored.class).name ()\\
                                                                     DAOAnnotation Utils.get String Value (instance,\\
                                                                                                                                           com. oyster. dao. annotation. utils. converter. String Convert\\
   * запит сутності у сховищі даних по його UUID
                                                                     primaryKeyField) + ";";
   * @param entityClass клас-обгортка сутності
                                                                          getJdbcTemplate().update(sql);
                                                                                                                                           import java.lang.annotation.Retention;
   * @param id
                    унікальний ідентифікатор
                                                                                                                                           import\ java.lang. annotation. Retention Policy;
сутності
   * @param <T>
                      тип сутності
                                                                        * запит для доступу до сховища даних
   * @return сутність, якщо така існує, інакше null
                                                                                                                                           * анотація визначає ключ поля, за яким зберігається
   * @throws DAOException
                                                                         * @param entityClass клас-обгортка сутності
                                                                                                                                           поле, а також ковертер для нього,
  @Override
                                                                        * @param filter умови, яким повинна
  public <T> T read(Class entityClass, UUID id)
                                                                      відповідати сутність
                                                                                                                                           @Retention(RetentionPolicy.RUNTIME)
throws DAOException {
                                                                         * @param <T>
                                                                                            тип сутності
                                                                                                                                           public @interface Stored {
                                                                         * @return список сутностей, що відповідають
    Field primaryKeyField =
DAOAnnotationUtils.getPrimaryKey(entityClass);
                                                                                                                                              * @return name ключ поля, за яким параметр
                                                                     умовам фільтру
     String sql = "SELECT * FROM " +
                                                                        * @throws DAOException
                                                                                                                                           зберігається
DAOAnnotation Utils.getStorageName (entity Class) \\
         + " WHERE " +
                                                                        @Override
                                                                                                                                             public String name();
primaryKeyField.getAnnotation(Stored.class).name()
                                                                        public <T> List<T> select(Class entityClass,
                                                                     DAOFilter filter) throws DAOException {
          + "=\"" + id.toString() + "\";";
                                                                                                                                              * повертає ValueConverter для даної сутності
                                                                          return select(entityClass, filter, null);
     Map<String, Object> map = null;
                                                                                                                                              * @return ValueConverter для даної сутності
       map = getJdbcTemplate().queryForMap(sql); \\
                                                                        * запит для доступу до сховища даних
     } catch (EmptyResultDataAccessException e) {
                                                                                                                                             public Class converter() default StringConverter.class;
       return null;
                                                                         * @param entityClass клас-обгортка сутності
                                                                                                                                          package com.oyster.dao.annotation;
     System.out.println(sql);
                                                                        * @param SQLString sql select
                                                                         * @param <T>
                                                                                           тип сутності
                                                                                                                                           import java.lang.annotation.ElementType;
                                                                         * @return список сутностей, що відповідають the
                                                                                                                                           import java.lang.annotation.Retention;
                                                                      sql-clause
DAOAnnotationUtils.mapToEntity(entityClass, map);
                                                                                                                                           import java.lang.annotation.RetentionPolicy;
                                                                         * @throws DAOException
                                                                                                                                           import java.lang.annotation.Target;
                                                                        @Override
   * оновлює сутність у сховищі даних
                                                                        public <T> List<T> select(Class entityClass, String
                                                                                                                                           * Анотація, що означає, що поле позначене як
                                                                     SQLString) throws DAOException
                                                                                                                                           PRIMARY KEY у сховищі даних
   * @param instance сутність для оновлення
                                                                          return select(entityClass, null, SQLString);
   * @param <T> тип сутності
   * @throws DAOException
                                                                                                                                           @Retention(RetentionPolicy.RUNTIME)
                                                                                                                                           @Target(value = ElementType.FIELD)
  @Override
                                                                        * допоміжна функція для пертворення різних
                                                                                                                                           public @interface Primary {
  public <T> void update(T instance) throws
                                                                      типів запитів
DAOException {
                                                                                                                                           package com.oyster.dao.annotation.utils;
    HashMap<String, String> mapStrStr =
                                                                         * @param entityClass клас-обгортка сутності
                                                                         * @param filter умови, яким повинна
                                                                                                                                           import com.oyster.dao.annotation.Primary;
(HashMap<String, String>)
                                                                      відповідати сутність
                                                                                                                                           import com.oyster.dao.annotation.Stored;
                                                                                          sql select-where clause
DAOAnnotation Utils.get Converted Stored Fields (instance) \\
                                                                         * @param sql
                                                                         * @param <T>
                                                                                           тип сутності
                                                                                                                                           import java.beans.IntrospectionException;
                                                                         * @return список сутностей, що відповідають the
                                                                                                                                           import java.beans.PropertyDescriptor;
                                                                                                                                           import java.lang.reflect.Field;
     StringBuilder sb = new StringBuilder();
                                                                      sql-clause і умовам фільтру
     for (String key: mapStrStr.keySet()) {
    sb.append(", " + key + "=" +
                                                                                                                                           import\ java.lang.reflect. Invocation Target Exception;
                                                                                                                                           import java.util.HashMap;
                                                                        private <T> List<T> select(Class entityClass,
mapStrStr.get(key));
                                                                     DAOFilter filter, String sql) {
                                                                                                                                           import java.util.Map;
                                                                                                                                           import java.util.UUID;
```

}

```
mapStrStr.put (key,\ getStringValue (instance,
                                                                                                                                          mapStrField.get(key)));
public class DAOAnnotationUtils {
                                                                        * @param с клас, з якого отримується інформація
                                                                     про поля позначені Stored
                                                                                                                                               return mapStrStr;
                                                                        * @return HashMap <stored name - це поле> для
   * @рагат с клас чи поле для повернення {@link
                                                                     поточного класу
com.oyster.dao.annotation.Stored} анотації
                                                                       public static HashMap<String, Field>
                                                                                                                                             * перетворює екземпляр класу в карту значень

    « @return ім'я,якщо клас має анотацію Stored

                                                                     getStoredFields(Class c) {
  public static String getStorageName(Class c) {
                                                                         HashMap<String, Field> res = new
                                                                                                                                             * @param instance екземпляр для конвертування
    Stored\ t = (Stored)\ c.getAnnotation(Stored.class);
                                                                    HashMap<String, Field>();
                                                                                                                                             * @param <T> тип екземпляру
    return (t != null) ? t.name() : null;
                                                                         Field[\ ]\ fields = c.getDeclaredFields();
                                                                                                                                             * @return карту значень <ключ - поле>
                                                                         for (Field field : fields) {
                                                                            Stored p = field.getAnnotation(Stored.class);
                                                                                                                                            public static <T> Map entityToMap(T instance) {
                                                                            if (p != null) {
                                                                              res.put(p.name(), field);
                                                                                                                                              PropertyDescriptor p;
   * @рагат с клас, з якого потрібно повернути
                                                                           }
                                                                                                                                              Map res = new HashMap<String, Object>();
інформацію
                                                                                                                                              HashMap<String, Field> storedFields =
                                                                                                                                          getStoredFields(instance.getClass());

    @return primary key класу

                                                                         return res;
  public static Field getPrimaryKey(Class c) {
                                                                                                                                                 for (String storedName : storedFields.keySet()) {
    Field[] \ fields = c.getDeclaredFields();
                                                                                                                                                   Field\ f = storedFields.get(storedName);
    for (Field field : fields) {
      Primary \ p = field.getAnnotation(Primary.class); \\
                                                                        * повертає значення як стрічку
                                                                                                                                                   p = new\ PropertyDescriptor(f.getName(),
      if (p != null) {
                                                                                                                                          instance.getClass());
         return field;
                                                                        * @param instance екземпляр
                                                                                                                                                   Class returnType =
                                                                                                                                          p.getReadMethod().getReturnType();
                                                                        * @param f поле для перетворення
                                                                        * @param <T> тип екземпляру
                                                                                                                                                   Object value =
                                                                        * @return поле як стрічку
    return null;
                                                                                                                                          p.getReadMethod().invoke(instance, null);
  }
                                                                                                                                                   ValueConverter\ c = getValueConverter(f);
                                                                       public static <T> String getStringValue(T instance,
                                                                                                                                                   res.put(storedName, c.toString(value));
                                                                     Field f) {
  * повертає значення UUID з поля Primary Key,
                                                                         try {
вважається, що його тип UUID
                                                                           PropertyDescriptor \ p=new
                                                                                                                                              \}\ catch\ (IntrospectionException\ |
                                                                    Property Descriptor (f.getName(), instance.getClass());\\
                                                                                                                                          IllegalAccessException \mid IllegalArgumentException \mid
   * @param instance екземпляр класу з якого
                                                                                                                                          InvocationTargetException e) {
потрібно отримати інформацію
                                                                            ValueConverter valueConverter =
                                                                                                                                                e.printStackTrace();
   * @param <T> тип класу
                                                                    getValueConverter(f);
   * @return UUID значення primary key
                                                                           return
                                                                     valueConverter.toString(p.getReadMethod().invoke(inst
                                                                                                                                              return null;
 public static <T> UUID getPrimaryKeyValue(T
                                                                     ance. null)):
                                                                         } catch (IllegalArgumentException e) {
instance) {
    Field pk = getPrimaryKey(instance.getClass());
                                                                           e.printStackTrace();
    PropertyDescriptor p;
                                                                          } catch (IllegalAccessException e) {
                                                                                                                                             * перетворює карту значень у екземпляр класу Т
                                                                            e.printStackTrace();
                                                                                                                                             * @param instanceClass клас екземпляру
      p = new\ PropertyDescriptor(pk.getName(),
                                                                          } catch (InvocationTargetException e) {
instance.getClass());
                                                                            e.printStackTrace():
                                                                                                                                             * @param map
                                                                                                                                                                  карта значень
      return ((UUID)
                                                                          } catch (IntrospectionException e) {
                                                                                                                                             * @param <T>
                                                                                                                                                                  тип екземпляру
p.getReadMethod().invoke(instance, null));
                                                                           e.printStackTrace();
                                                                                                                                             * @return екземпляр із карти
    } catch (IntrospectionException e) {
      e.printStackTrace();
                                                                         return null;
                                                                                                                                            public static <T> T mapToEntity(Class instanceClass,
     } catch (IllegalAccessException e) {
                                                                                                                                          Map map) {
       e.printStackTrace();
    } catch (IllegalArgumentException e) {
                                                                                                                                              PropertyDescriptor p;
                                                                                                                                              HashMap<String, Field> storedFields =
      e.printStackTrace();
                                                                        * повертає конвертер даного поля
     } catch (InvocationTargetException e) {
                                                                                                                                          getStoredFields(instanceClass);
      e.printStackTrace();
                                                                        * @param field поле
                                                                        * @param <T> тип екземпляру
                                                                                                                                                 T instance = (T) instanceClass.newInstance();
                                                                        * @return конвертер поля
    return null:
                                                                                                                                                 for (String storedName : storedFields.keySet()) {
                                                                       public static <T extends ValueConverter> T
                                                                                                                                                   Object strValue = map.get(storedName);
                                                                                                                                                   Field f = storedFields.get(storedName);
                                                                     getValueConverter(Field field) {
   * встановлює значення UUID з поля Primary Key,
                                                                         Stored s = (Stored)
                                                                                                                                                   ValueConverter\ c = getValueConverter(f);
вважається, що його тип UUID
                                                                     field.getAnnotation(Stored.class):
                                                                                                                                                   p = new\ PropertyDescriptor(f.getName(),
                                                                         Class converterClass = s.converter();
                                                                                                                                          instanceClass);
   * @param instance екземпляр класу з якого
                                                                                                                                                   p.getWriteMethod().invoke(instance,
                                                                         try {
потрібно отримати інформацію
                                                                            ValueConverter res = (ValueConverter)
                                                                                                                                          c.toValue(c.toString(strValue)));
   * @param value нове значення для встановлення
                                                                     converterClass.newInstance();
primary key
                                                                           return (T) res;
                                                                                                                                                 return instance;
                                                                          } catch (InstantiationException e) {
   * @param <T> тип класу
                                                                                                                                               } catch (IntrospectionException e) {
                                                                           e.printStackTrace():
                                                                                                                                                 e.printStackTrace():
 public static <T> void setPrimaryKeyValue(T
                                                                          } catch (IllegalAccessException e) {
                                                                                                                                              } catch (InstantiationException e) {
instance, UUID value) {
                                                                           e.printStackTrace();
                                                                                                                                                 e.printStackTrace();
    Field pk = getPrimaryKey(instance.getClass());
                                                                                                                                               } catch (IllegalAccessException e) {
    PropertyDescriptor p;
                                                                         return null:
                                                                                                                                                 e.printStackTrace();
                                                                                                                                               } catch (IllegalArgumentException e) {
                                                                       }
    try {
      p = new PropertyDescriptor(pk.getName(),
                                                                                                                                                 e.printStackTrace();
instance.getClass());
                                                                                                                                               } catch (InvocationTargetException e) {
      p.getWriteMethod().invoke(instance, value);
                                                                        * мапа Stored.name() - значення поля
                                                                                                                                                 e.printStackTrace();
     } catch (IntrospectionException e) {
                                                                                                                                               } catch (NullPointerException e) {
       e.printStackTrace();
                                                                        * @param instance екземпляр для конвертації
                                                                                                                                                 e.printStackTrace();\\
                                                                        * @param <T> тип екземпляру
    } catch (IllegalAccessException e) {
      e.printStackTrace();
                                                                        * @return мапа, яка містить збережене ім'я поля -
                                                                                                                                              return null;
     } catch (IllegalArgumentException e) {
                                                                     його значення після застосування ValueConverter
       e.printStackTrace();
     } catch (InvocationTargetException e) {
                                                                       public static <T> Map<String, String>
                                                                     getConvertedStoredFields(T instance) {
      e.printStackTrace();
                                                                         HashMap<String, Field> mapStrField =
                                                                                                                                             * конструктор за замовчуванням
                                                                     getStoredFields(instance.getClass());
                                                                         HashMap<String, String> mapStrStr = new
                                                                                                                                            public DAOAnnotationUtils() {
                                                                     HashMap<String, String>();
   * повертає HashMap<String, Field> для поточного
                                                                         for (String key : mapStrField.keySet()) {
                                                                                                                                          package com.oyster.dao.annotation.utils:
класу с
```

* String - Stored.name() value

```
package com.oyster.ui;
* інтерфейс для класів, які пертворюють певні поля
типу Т в String і навпаки
                                                                        * перетворює String в Integer
                                                                                                                                          import com.oyster.app.AppConst;
                                                                        * @param str String для конвертування
                                                                                                                                          import com.oyster.app.model.*;
public interface ValueConverter {
                                                                                                                                          import com.oyster.core.controller.CommandExecutor:
                                                                        * @рагат <T> тип параметру
                                                                        * @return стрічку як параметр
                                                                                                                                          import com.oyster.core.controller.command.Context;
   * конвертує екземпляр типу Т в String
                                                                                                                                          import com.oyster.dao.DAOFilter;
                                                                                                                                          import com.oyster.dao.exception.DAOException;
                                                                        @Override
   * @param value параметр для конвертування
                                                                       public <T> T toValue(String str) {
                                                                                                                                          import com.oyster.ui.dialogs.*;
   * @param <T> тип параметру
                                                                          if \ (str.equals ("null")) \ return \ null; \\
   * @return параметр конвертований в String
                                                                          return (T) new Integer(Integer.parseInt(str));
                                                                                                                                          import javax.swing.*;
                                                                                                                                          import javax.swing.event.DocumentEvent;
  public <T> String toString(T value);
                                                                                                                                          import javax.swing.event.DocumentListener;
                                                                     package com.oyster.dao.annotation.utils.converter;
                                                                                                                                          import\ javax. swing. event. List Selection Event;
                                                                                                                                          import javax.swing.event.ListSelectionListener;
   * конвертує String до екземпляру типу Т
                                                                     import com.oyster.dao.annotation.utils.ValueConverter;
                                                                                                                                          import javax.swing.text.DateFormatter;
                                                                                                                                          import java.awt.*;
   * @param str String для конвертування
                                                                                                                                          import java.awt.event.ActionEvent;
   * @рагат <T> тип параметру
                                                                      * конвертує сутність (String <==> String)
                                                                                                                                          import java.awt.event.ActionListener;
   * @return об'єкт типу Т
                                                                                                                                          import java.awt.event.KeyEvent;
                                                                     public class StringConverter implements
                                                                                                                                          import java.text.ParseException;
  public <T> T toValue(String str);
                                                                     ValueConverter {
                                                                                                                                          import java.util.ArrayList;
                                                                                                                                          import java.util.Date:
package com.oyster.dao.annotation.utils.converter;
                                                                        * перетворює String в String
import com.oyster.dao.annotation.utils.ValueConverter;
                                                                        * @param value параметр для конвертації
                                                                                                                                           * Клас відповідає табові, що реалізує вікно
import java.math.BigInteger;
                                                                        * @param <T> тип параметру
                                                                                                                                          кправління профілями

    @return параметр як стрічку

                                                                                                                                           * @author bamboo
* конвертує сутність (String <==> Long)
                                                                       public <T> String toString(T value) {
                                                                                                                                          public class MainForm extends JFrame {
public class LongConverter implements ValueConverter
                                                                          if (value == null) return null:
                                                                          return "\"" + ((Object) value).toString() + "\"";
                                                                                                                                             private IPanel rootPanel:
                                                                                                                                             private JTabbedPane mTabbedPaneMain;
                                                                                                                                             private JComboBox mComboBoxProfileType;
   * перетворює Long в String
                                                                                                                                            private JComboBox
   * @param value параметр для конвертації
                                                                        * перетворює String в String
                                                                                                                                          mComboBoxProfileStudentTypeFaculty;
   * @param <T> тип параметру
                                                                        * @param str String для конвертування
                                                                                                                                             private JList mListPeople;
   * @return параметр як стрічку
                                                                        * @param <T> тип параметру
                                                                                                                                             private JComboBox
                                                                        * @return стрічку як параметр
                                                                                                                                          mComboBoxProfileStudentTypeGroup:
   @Override
                                                                                                                                            private JPanel mPanelProfile;
  public <T> String toString(T value) {
                                                                        @Override
                                                                                                                                             private JPanel mPanelLeftControl;
     if (value == null) return "null";
                                                                       public <T> T toValue(String str) {
                                                                                                                                             private JButton mButtonSave;
                                                                                                                                             private JButton mButtonDelete:
                                                                          if (str == null) return null;
                                                                                                                                             private JList mListHistoryProfile;
     // костиль (for Spring JDBC)
                                                                          if (str.equals("null")) return null:
                                                                          if (str.equals("\"\"")) return (T) "";
if (str.equals("")) return (T) "";
    if (value instanceof BigInteger) {
                                                                                                                                             private JTextField mTextFieldInfo1;
       return value.toString();
                                                                                                                                             private JTextField mTextFieldInfo2;
                                                                                                                                             private JTextField mTextFieldInfo3;
                                                                          str = str.substring(1, str.length() - 1);
                                                                                                                                             private JTextField mTextFieldInfo5;
     return Long.toString((Long) value);
                                                                          return (T) str:
                                                                                                                                             private JTextField mTextFieldInfo6;
                                                                                                                                             private JTextField mTextFieldInfo7:
                                                                                                                                             private JButton mButtonNewUser;
                                                                     package com.oyster.dao.annotation.utils.converter;
                                                                                                                                             private JComboBox mComboBoxAllHistory;
                                                                                                                                             private JList mListAllHistory;
   * перетворює String в Long
                                                                     import\ com. oyster. dao. annotation. utils. Value Converter;
                                                                                                                                             private JComboBox mComboBoxScheduleFaculty;
   * @param str String для конвертування
                                                                                                                                             private JScrollPane mListGroups;
                                                                     import java.util.UUID;
   * @рагат <T> тип параметру
                                                                                                                                             private JTable mTable1;

    @return стрічку як параметр

                                                                                                                                             private JLabel mLabel5;
                                                                                                                                             private JLabel mLable6;
                                                                      * конвертує сутність (String <==> UUID)
   @Override
                                                                                                                                             private JLabel mLable7;
                                                                                                                                             private JLabel mLable1;
  public <T> T toValue(String str) {
                                                                     public class UUIDConverter implements
    if (str.equals("null")) return null;
                                                                                                                                             private JLabel mLable2:
     return (T) new Long(Long.parseLong(str));
                                                                                                                                             private JLabel mLable3;
                                                                     ValueConverter {
                                                                                                                                             private JLabel mLablePhoto;
                                                                                                                                             private JLabel mLable8;
package com.oyster.dao.annotation.utils.converter;
                                                                        * перетворює UUID в String
                                                                                                                                             private JPasswordField mPasswordField1;
                                                                                                                                             private JLabel mLable4;
                                                                                                                                             private JTextField mTextFieldInfo4;
import com.oyster.dao.annotation.utils.ValueConverter;
                                                                        * @param value параметр для конвертації
                                                                                                                                             private JScrollPane mScrollPanePeople;
                                                                        * @param <T> тип параметру
                                                                        * @return параметр як стрічку
                                                                                                                                            private JList mListTab2Group;
* конвертує сутність (String <==> Integer)
                                                                        @Override
                                                                                                                                             private IProfile currentPerson;
                                                                       public <T> String toString(T value) {
public class IntConverter implements ValueConverter {
                                                                                                                                            private DocumentListener mDocumentListener = new
                                                                          if (value == null) return null;
                                                                          return "\"" + ((Object) value).toString() + "\"";
                                                                                                                                          DocumentListener() {
                                                                                                                                               @Override
   * перетворює Integer в String
                                                                                                                                               public void insertUpdate(DocumentEvent e) {
   * @param value параметр для конвертації
                                                                                                                                                 act();
   * @param <T> тип параметру
                                                                        * перетворює String в UUID

    @return параметр як стрічку

                                                                        * @param str String для конвертування
                                                                                                                                               @Override
   @Override
                                                                        * @рагат <T> тип параметру
                                                                                                                                               public void removeUpdate(DocumentEvent e) {
  public <T> String toString(T value) {
                                                                        * @return стрічку як параметр
                                                                                                                                                 act();
    if \ (value == null) \ return \ "null"; \\
                                                                        @Override
     // костиль (for Spring JDBC)
                                                                       public <T> T toValue(String str) {
     if (value instanceof Long) {
                                                                          if (str == null) return null;
                                                                                                                                               @Override
       return value.toString():
                                                                          if (str.equals("null")) return null;
                                                                                                                                               public void changedUpdate(DocumentEvent e) {
                                                                          str = str.substring(1, str.length() - 1);
                                                                          return\ (T)\ UUID.fromString(str);
```

return Integer.toString((Integer) value);

```
private void act() {
                                                                                                       public void actionPerformed(ActionEvent e) {
                                                                                                                                                                                                * метод спрацьовує при натисненні на кнопку
         mButtonSave.setEnabled(true);
                                                                                                          groupChangedAction();
                                                                                                                                                                                             "Зберегти
                                                                                                                                                                                               private void saveButtonClick() {
   private HistoryTab historyTab:
                                                                                             mTextFieldInfo1.getDocument().addDocumentListener(
  private ScheduleTab scheduleTab;
                                                                                                                                                                                                  if (currentPerson == null) {
                                                                                              mDocumentListener):
                                                                                                                                                                                                     return;
                                                                                             mTextFieldInfo2.getDocument().addDocumentListener(\\
    * конструктор, створює елементи інтерфейсу
                                                                                              mDocumentListener);
                                                                                                                                                                                                  if (currentPerson instanceof Admin) {
                                                                                                                                                                                                      Admin a = (Admin) currentPerson:
   public MainForm() {
                                                                                             mTextFieldInfo3.getDocument().addDocumentListener(
                                                                                                                                                                                                      saveUpdate(a.getProfile()):
      super((String)
                                                                                             mDocumentListener);
                                                                                                                                                                                                      saveUpdate(a.getWorkerInfo());
AppConst. APP\_CONFIG.getValue("progTitle"));
                                                                                              mTextFieldInfo4.getDocument().addDocumentListener(\\
                                                                                                                                                                                                   } else if (currentPerson instanceof Teacher) {
       add(rootPanel);
                                                                                              mDocumentListener);
                                                                                                                                                                                                      Teacher t = (Teacher) currentPerson;
                                                                                                                                                                                                      saveUpdate(t.getProfile());
      int width = (Integer)
                                                                                              mTextFieldInfo5.getDocument().addDocumentListener(
                                                                                                                                                                                                     saveUpdate(t.getWorkerInfo());
AppConst.APP_CONFIG.getValue("mainScreenWidth"
                                                                                              mDocumentListener):
                                                                                                                                                                                                   } else if (currentPerson instanceof Student) {
      int height = (Integer)
                                                                                             mTextFieldInfo6.getDocument().addDocumentListener(\\
                                                                                                                                                                                                     Student \; s = (Student) \; currentPerson; \\
AppConst.APP_CONFIG.getValue("mainScreenHeight"
                                                                                             mDocumentListener);
                                                                                                                                                                                                      saveUpdate(s);
                                                                                             mTextFieldInfo7.getDocument().addDocumentListener(\\
                                                                                                                                                                                                  updateUI();
       setPreferredSize(new Dimension(width, height));
                                                                                             mDocumentListener);
       setMinimumSize(new Dimension(width, height));
                                                                                              mPasswordField1.getDocument().addDocumentListener
      hardCoreInit():
                                                                                             (mDocumentListener);
                                                                                                                                                                                                * зберігає інформацію про профіль студента
                                                                                                    mButtonSave.addActionListener(new
      pack();
                                                                                              ActionListener() {
                                                                                                                                                                                                * @param s екземпляр класу Student
                                                                                                        @Override
setDefaultCloseOperation(WindowConstants.EXIT_ON
                                                                                                        public void actionPerformed(ActionEvent e) {
                                                                                                                                                                                               private void saveUpdate(Student s) {
_CLOSE):
                                                                                                          saveButtonClick():
                                                                                                                                                                                                  saveUpdate(s.getProfile());\\
      setLocationRelativeTo(null);
                                                                                                    });
                                                                                                                                                                                                  String course = mTextFieldInfo5.getText().trim();
                                                                                                                                                                                                   String bookNum =
       setVisible(true);
                                                                                                    mButtonDelete.addActionListener(new
                                                                                                                                                                                            mTextFieldInfo7.getText().trim();
                                                                                              ActionListener() {
                                                                                                        @Override
                                                                                                                                                                                                  String facultyName =
                                                                                                                                                                                            mTextFieldInfo4.getText().trim();
                                                                                                       public void actionPerformed(ActionEvent e) {
                                                                                                          deleteButtonClick();
                                                                                                                                                                                                  String groupName =
    * ініціалізує елементи інтерфейсу
                                                                                                                                                                                            mTextFieldInfo6.getText().trim();
                                                                                                    });
   private void hardCoreInit() {
                                                                                                                                                                                                  s.setCourse(Integer.parseInt(course));
                                                                                                    scheduleTab = new ScheduleTab(this,
                                                                                                                                                                                                  s.setBookNum(Integer.parseInt(bookNum));\\
      addJMenu();
                                                                                             mComboBoxScheduleFaculty, mListTab2Group,
      mButtonNewUser.addActionListener(new
                                                                                                                                                                                                  s.getGroup().setName(groupName);
ActionListener() {
                                                                                                                                                                                                  s.getGroup().getFaculty().setName(facultyName);\\
          @Override
                                                                                                    historyTab = new HistoryTab(this,
          public void actionPerformed(ActionEvent e) {
                                                                                              mComboBoxAllHistory,\,mListAllHistory);\\
                                                                                                                                                                                                     AppConst.DAO.update(s);
            newUserAction();
                                                                                                                                                                                                     AppConst.DAO.update(s.getGroup());
      });
      comboBoxChangeAction();
                                                                                                                                                                                            AppConst.DAO.update(s.getGroup().getFaculty());\\
                                                                                                  * метод спрацьовує при натисненні на кнопку
      mComboBoxProfileType.addActionListener(new
                                                                                              "Видалити"
                                                                                                                                                                                                  } catch (DAOException e) {
                                                                                                                                                                                                     e.printStackTrace();
ActionListener() {
                                                                                                 private void deleteButtonClick() {
          @Override
         public void actionPerformed(ActionEvent e) {
             comboBoxChangeAction();
                                                                                                    if (currentPerson == null) {
      });
                                                                                                                                                                                                * зберігає інформацію про профіль
      mListPeople.addListSelectionListener(new
                                                                                                    int dialogResult =
                                                                                                                                                                                                * @param p екземпляр класу Profile
ListSelectionListener() {
                                                                                             JOptionPane.showConfirmDialog(null,
          @Override
                                                                                                           "Підтвердити видалення акаунту: ",
                                                                                                                                                                                               private void saveUpdate(Profile p) {
                                                                                                                                                                                                  String firstName
          public void valueChanged(ListSelectionEvent e)
                                                                                                           "Захист від дурака",
                                                                                                          JOptionPane.OK_CANCEL_OPTION,
                                                                                                                                                                                            mTextFieldInfo1.getText().trim();\\
             Object o = mListPeople.getSelectedValue();
                                                                                                          JOptionPane.WARNING_MESSAGE
                                                                                                                                                                                                  String secondName
                                                                                                                                                                                            mTextFieldInfo2.getText().trim();
             if (o instanceof Admin) {
                fillInfoFields((Admin) o);
                                                                                                                                                                                                  String password =
                                                                                                    if (dialogResult ==
             } else if (o instanceof Teacher) {
                                                                                                                                                                                            mPasswordField1.getText().trim();
                fillInfoFields((Teacher) o);
                                                                                             JOptionPane.CANCEL\_OPTION)\ \{
                                                                                                                                                                                                  String\ birthday = mTextFieldInfo3.getText().trim();
             } else if (o instanceof Student) {
                                                                                                       return:
                                                                                                                                                                                                  Long\ birthdayLong=0L;
               fillInfoFields((Student) o);
                                                                                                    }
                                                                                                                                                                                                  try {
                                                                                                                                                                                                     Date d = (Date) new
                                                                                                                                                                                            DateFormatter(AppConst.DATE_FORMAT).stringToV
                                                                                                    Context c = new Context();
                                                                                                    c.put("profile", currentPerson);
                                                                                                                                                                                                     birthdayLong = d.getTime();
                                                                                                    try {
                                                                                                                                                                                                   } catch (ParseException e) {
mComboBoxProfileStudentTypeFaculty.addActionLister \\
                                                                                                                                                                                                     e.printStackTrace();
                                                                                             CommandExecutor.getInstance().execute("deleteIProfile
ner(new ActionListener() {
          @Override
                                                                                              ", c, null);
                                                                                                                                                                                                  p.setFirstName(firstName);
          public void actionPerformed(ActionEvent e) {
                                                                                                     } catch (Exception e) {
                                                                                                                                                                                                  p.setSecondName(secondName);
                                                                                                       e.printStackTrace();
             facultyChangedAction();
                                                                                                                                                                                                  p.setBirthday(birthdayLong);
                                                                                                                                                                                                  p.setPassword(password);
      });
                                                                                                                                                                                                     AppConst.DAO.update(p);
                                                                                                    currentPerson = null;
                                                                                                                                                                                                  } catch (DAOException e) {
mComboBox Profile Student Type Group. add Action Listen Type Group. Add Action Listen Type Group and Type Group and Type Group and Type Group. The Action Listen Type Group and Type Gro
                                                                                                     comboBoxChangeAction():
                                                                                                                                                                                                     e.printStackTrace();
er(new ActionListener() {
          @Override
```

```
student.setProfile((Profile)
                                                                     AppConst.DAO.read(Profile.class,
  * зберігає інформацію про роботу
                                                                     student.getProfileId()));
                                                                                                                                                   DefaultListModel<Admin> model = new
                                                                                                                                         DefaultListModel<>();
                                                                              student.setGroup(group);
  * @param wi екземпляр класу WorkerInfo
                                                                                                                                                   for (Admin a : admins) {
                                                                         catch (Exception e) {
                                                                                                                                                     model addElement(a):
  private void saveUpdate(WorkerInfo wi) {
                                                                           e.printStackTrace();
    String position = mTextFieldInfo4.getText().trim();
                                                                                                                                                   mListPeople.setModel(model);
     String salary = mTextFieldInfo5.getText().trim();
                                                                         showFilteredStudents(students);
    String dateHiredStr =
                                                                                                                                                   mListPeople.setSelectedIndex(0);
mTextFieldInfo6.getText().trim();
    Long dateHired = 0L;
                                                                                                                                                   break:
                                                                        * метод відображає список студентів, що
    trv {
       Date d = (Date) new
                                                                     пройшли фільтри, встановлені користувачем
                                                                                                                                                case 2:
DateFormatter (AppConst.DATE\_FORMAT). stringToV
alue(dateHiredStr);
                                                                        * @param students список студентів
                                                                                                                                                   mScrollPanePeople.setVisible(true);
      dateHired = d.getTime();
                                                                                                                                                   mButtonDelete.setEnabled(true);
     } catch (ParseException e) {
                                                                      private void
      e.printStackTrace();
                                                                     showFilteredStudents(java.util.List<Student> students) {
                                                                                                                                                   java.util.List<Teacher> teachers = null;
                                                                         DefaultListModel<Student> model = new
     wi.setPosition(position);
                                                                     DefaultListModel<Student>();
    wi.setSalary(Integer.parseInt(salary));
                                                                         for (Student s : students) {
                                                                                                                                                     teachers =
                                                                                                                                         AppConst.DAO.select(Teacher.class, "");
    wi.setDateHired(dateHired);
                                                                           model.addElement(s);
    try {
                                                                                                                                                     for (Teacher teacher: teachers) {
       AppConst.DAO.update(wi);
                                                                         mListPeople.setModel(model);
                                                                                                                                                       teacher.setProfile((Profile)
    } catch (DAOException e) {
                                                                                                                                         AppConst.DAO.read(Profile.class,
      e.printStackTrace();
                                                                         if (students.size() > 0) {
                                                                                                                                         teacher.getProfileId()));
                                                                           mListPeople.setSelectedIndex(0);
                                                                                                                                                       teacher.setWorkerInfo((WorkerInfo)
                                                                                                                                          AppConst.DAO.read(WorkerInfo.class,
                                                                         updateUI();
                                                                                                                                         teacher.getWorkerInfoId()));
  * оновлює інтерфейс
                                                                                                                                                   } catch (DAOException e) {
                                                                                                                                                     e.printStackTrace();
  private void updateUI() {
                                                                       * метод спрацьовує при зміні типу користувача
    mButtonSave.setEnabled(false):
                                                                                                                                                   DefaultListModel<Teacher> teacherModel =
                                                                      private void comboBoxChangeAction() {
    validate();
                                                                                                                                         new DefaultListModel<>();
    repaint();
                                                                         mButtonSave.setEnabled(false);
                                                                                                                                                   for (Teacher teacher : teachers) {
                                                                         int selected =
                                                                                                                                                     teacherModel.addElement(teacher);
                                                                     mComboBoxProfileType.getSelectedIndex();
  * метод спрацьовує при зміні факультету
                                                                         if (selected < 3) {
                                                                                                                                                   mListPeople.setModel(teacherModel);
                                                                                                                                                   mListPeople.setSelectedIndex(0):
  private void facultyChangedAction() {
                                                                    mComboBoxProfileStudentTypeFaculty.setVisible(false
    final Faculty f = (Faculty)
                                                                                                                                                   break;
mComboBoxProfileStudentTypeFaculty.getSelectedIte
                                                                    mComboBoxProfileStudentTypeGroup.setVisible(false)\\
                                                                                                                                                case 3:
    java.util.List < Group > groups = null;
                                                                            mLable7.setVisible(false);
                                                                                                                                                   mScrollPanePeople.setVisible(true);
    try {
       groups = AppConst.DAO.select(Group.class,
                                                                            mTextFieldInfo7.setVisible(false);
                                                                                                                                                   mButtonDelete.setEnabled(true);
new DAOFilter() {
                                                                            mLable4.setText("Посада");
         @Override
                                                                            mLabel5.setText("Зарплата");
                                                                                                                                                   java.util.List \!\!<\!\! Faculty \!\!>\! faculties = null;
         public <T> boolean accept(T entity) {
                                                                            mLable6.setText("Працює із");
           Group g = (Group) entity;
                                                                                                                                                   try {
           return g.getFacultyId().equals(f.getId());
                                                                                                                                                     faculties =
                                                                         } else {
                                                                                                                                         AppConst.DAO.select(Faculty.class, "");
       });
                                                                                                                                                   } catch (DAOException e) {
       for (Group g : groups) {
                                                                    mComboBoxProfileStudentTypeFaculty.setVisible(true)\\
                                                                                                                                                     e.printStackTrace();
         g.setFaculty(f);
                                                                                                                                                   }
                                                                    mComboBoxProfileStudentTypeGroup.setVisible(true);\\
    } catch (Exception e) {
      e.printStackTrace();
                                                                                                                                         mComboBoxProfileStudentTypeFaculty.setModel(new
                                                                            mLable7.setVisible(true);
                                                                            mTextFieldInfo7.setVisible(true);
                                                                                                                                         DefaultComboBoxModel(faculties.toArray()));
                                                                            mLable4.setText("Факультет"):
                                                                            mLabel5.setText("Kypc");
mComboBoxProfileStudentTypeGroup.setModel(new
Default Combo Box Model (groups.to Array ()));\\
                                                                           mLable6.setText("Група");
                                                                                                                                         mComboBoxProfileStudentTypeFaculty.setSelectedInder \\
    if (groups.size() > 0) {
                                                                            mLable7.setText("H3K");
                                                                                                                                         x(0);
mComboBoxProfileStudentTypeGroup.setSelectedIndex\\
                                                                         switch
                                                                                                                                                   break;
(0);
                                                                    (mComboBoxProfileType.getSelectedIndex())\ \{\\
                                                                           case 0:
                                                                                                                                              updateUI();
    updateUI();
                                                                              mScrollPanePeople.setVisible(false);
                                                                              mButtonDelete.setEnabled(false);
                                                                              fillInfoFields (AppConst.getCurrentAdmin());\\
                                                                                                                                            * метод заповнює інформацію про профіль
                                                                              break;
  * метод спрацьовує при зміні групи
                                                                           case 1:
                                                                                                                                            * @param profile екемпляр класу Profile
                                                                              mScrollPanePeople.setVisible(true);
  private void groupChangedAction() {
    final Group group = (Group)
                                                                              mButtonDelete.setEnabled(true);
                                                                                                                                           private void fillInfoFields(Profile profile) {
mComboBoxProfileStudentTypeGroup.getSelectedItem(
                                                                                                                                              mTextFieldInfo1.setText(profile.getFirstName());\\
                                                                              java.util.List<Admin> admins = null;
    java.util.List<Student> students = null;
                                                                                                                                         mTextFieldInfo2.setText(profile.getSecondName());
    try {
      students = AppConst.DAO.select(Student.class,
                                                                                admins =
                                                                                                                                         mTextFieldInfo3.setText(AppConst.DATE_FORMAT.f
new DAOFilter() {
                                                                    AppConst.DAO.select(Admin.class, "");
                                                                                                                                         ormat(new Date(profile.getBirthday())));
                                                                                for (Admin a : admins) {
                                                                                                                                              mPasswordField1.setText(profile.getPassword());\\
         @Override
                                                                                   a.setProfile((Profile)
         public <T> boolean accept(T entity) {
                                                                     AppConst.DAO.read(Profile.class, a.getProfileId()));
           Student s = (Student) entity;
                                                                                                                                              loadHistory(profile);
           return\ s.getGroupId().equals(group.getId());
                                                                                  a.setWorkerInfo((WorkerInfo)
                                                                                                                                            }
                                                                    AppConst.DAO.read(WorkerInfo.class,
       }):
                                                                    a.getWorkerInfoId()));
       for (Student student : students) {
                                                                                                                                            * метод завантажує історію вибраного
                                                                              } catch (DAOException e) {
                                                                                                                                         користувача
                                                                                e.printStackTrace();
```

```
 @param profile користувач

                                                                                                                                              menuItem = new JMenuItem("Додати акаунт".
                                                                    mTextFieldInfo6.setText(student.getGroup().getName())
                                                                                                                                                   KeyEvent
  private void loadHistory(final Profile profile) {
     final java.util.List<History> histories = new
                                                                     mTextFieldInfo7.setText(String.valueOf(student.getBoo
                                                                                                                                              menuItem.setMnemonic(KeyEvent.VK_N);
ArrayList<History>();
                                                                    kNum())):
                                                                                                                                              menuItem.setAccelerator(KeyStroke.getKeyStroke(\\
                                                                                                                                                   KeyEvent.VK N,
     Context c = new Context();
                                                                         currentPerson = student;
                                                                                                                                         ActionEvent.ALT_MASK));
    c.put("list", histories);
                                                                         updateUI();
                                                                                                                                              menuItem.addActionListener(new\ ActionListener()
c.put("sqlQuery", "select * from HISTORY_TBL
where author_id = \"" +
                                                                                                                                                 @Override
         profile.getId() + "\";");
                                                                                                                                                public void actionPerformed(ActionEvent e) {
                                                                        * метод спрацьовує при натисненні на клавішу
                                                                                                                                                   newUserAction():
                                                                     "Додати аккаунт'
    try {
                                                                                                                                              });
Command Executor.get Instance (). execute ("load History", \\
                                                                       private void newUserAction() {
                                                                                                                                              menu.add(menuItem);
                                                                     Object[] possibilities = {"Студент", "Виладач", "Адміністратор", "Група", "Факультет", "Предмет"};
String s = (String) JOptionPane.showInputDialog(
c, new Runnable() {
         @Override
                                                                                                                                              menu.addSeparator();
         public void run() {
            for (History h : histories) {
                                                                              MainForm.this,
                                                                                                                                              menuItem = new JMenuItem("Вихід",
              h.setAuthor(profile);
                                                                              "Виберіть категорію, яку хочете
                                                                                                                                                   new ImageIcon("images/middle.gif"));
                                                                     створити:\п",
                                                                                                                                              menuItem.setMnemonic(KeyEvent.VK\_W);
           DefaultListModel<History> historyModel =
                                                                              "Новий акаунт",
                                                                                                                                              menuItem.setAccelerator(KeyStroke.getKeyStroke(\\
new DefaultListModel<>():
                                                                              JOptionPane.PLAIN MESSAGE,
                                                                                                                                                   KevEvent.VK W.
           for (History h : histories) {
                                                                                                                                         ActionEvent.ALT_MASK));
                                                                              null.
              historyModel.addElement(h);
                                                                              possibilities,
                                                                                                                                              menuItem.addActionListener(new\ ActionListener()
                                                                               "Студент");
                                                                                                                                                 @Override
                                                                         if ((s != null) && (s.length() > 0)) {
                                                                                                                                                public void actionPerformed(ActionEvent e) {
mListHistoryProfile.setModel(historyModel);
                                                                                                                                                   MainForm.this.dispose();
                                                                            JDialog dialog = null;
         }
       });
                                                                            switch (s) {
                                                                                                                                              });
     } catch (Exception e) {
                                                                              case "Студент"
                                                                                                                                              menu.add(menuItem);
       e.printStackTrace();
                                                                                dialog = new \\
                                                                                                                                              menu = new JMenu("Допомога").
                                                                    NewStudentCustomDialog(this);
                                                                                                                                              menu.setMnemonic(KeyEvent.VK_H);
                                                                                break;
                                                                                                                                              menuBar.add(menu);
   * метод заповнює інформацію про роботу
                                                                              case "Виладач":
                                                                                                                                              menuItem = new JMenuItem("Посібник
                                                                                dialog = new
                                                                                                                                          користувача",
   * @param workerInfo екемпляр класу WorkerInfo
                                                                     NewTeacherCustomDialog(this);
                                                                                                                                                   KeyEvent.VK\_K);
                                                                                                                                              menuItem.setAccelerator(KevStroke.getKevStroke(
                                                                                break:
  private void fillInfoFields(WorkerInfo workerInfo) {
                                                                                                                                                   KeyEvent.VK_K,
                                                                                                                                         ActionEvent.ALT_MASK));
                                                                              case "Адміністратор":
mTextFieldInfo4.setText(workerInfo.getPosition());\\
                                                                                                                                              menu.add(menuItem);
                                                                                dialog = new
                                                                                                                                              menuItem = new JMenuItem("Προ προграму");
mTextFieldInfo5.setText(String.valueOf(workerInfo.get
                                                                    NewAdminCustomDialog(this);
                                                                                                                                              menuItem.setAccelerator(KeyStroke.getKeyStroke(
Salary()));
                                                                                break;
                                                                                                                                                   KeyEvent.VK_A,
                                                                              case "Група":
                                                                                                                                         ActionEvent.ALT_MASK));
mTextFieldInfo6.setText(AppConst.DATE_FORMAT.f
ormat(new Date(workerInfo.getDateHired())));
                                                                                                                                              menu.add(menuItem);
                                                                                dialog = new
                                                                    NewGroupCustomDialog(this);
                                                                                                                                              setJMenuBar(menuBar);
                                                                                break;
                                                                                                                                           }
   * метод заповнює інформацію про адміністратора
                                                                              case "Факультет":
                                                                                                                                         package com.oyster.ui;
   * @param admin екемпляр класу Admin
                                                                                dialog = new \\
                                                                                                                                         import com.oyster.app.AppConst;
                                                                    NewFacultyCustomDialog(this);
                                                                                                                                         import com.oyster.app.model.*;
  private void fillInfoFields(Admin admin) {
                                                                                                                                         import com.oyster.core.controller.CommandExecutor;
                                                                                break;
     fillInfoFields(admin.getProfile());
                                                                                                                                         import com.oyster.core.controller.command.Context;
     fillInfoFields(admin.getWorkerInfo());
                                                                              case "Предмет":
                                                                                                                                         import com.oyster.dao.DAOFilter;
                                                                                                                                         import com.oyster.dao.exception.DAOException;
    currentPerson = admin;
                                                                                dialog = new
                                                                                                                                         import javax.swing.*;
    updateUI();
                                                                     NewSubjectCustomDialog(this);
                                                                                break;
                                                                                                                                         import javax.swing.event.ListSelectionEvent;
                                                                                                                                         import javax.swing.event.ListSelectionListener;
                                                                            dialog.pack();
                                                                                                                                         import javax.swing.event.TableModelEvent;
                                                                            dialog.setVisible(true);
   * метод заповнює інформацію про викладача
                                                                                                                                         import\ javax. swing. event. Table Model Listener;
                                                                                                                                         import\ javax. swing. table. Abstract Table Model;
                                                                                                                                         import javax.swing.table.DefaultTableCellRenderer;
   * @param teacher екемпляр класу Teacher
                                                                                                                                         import javax.swing.table.TableCellRenderer;
                                                                                                                                         import javax.swing.table.TableColumn;
  private void fillInfoFields(Teacher teacher) {
     fill Info Fields (teacher.get Profile ());\\
                                                                        * ініціалізує меню у програмі
                                                                                                                                         import java.awt.*;
     fillInfoFields(teacher.getWorkerInfo());
                                                                                                                                         import java.awt.event.ActionEvent;
                                                                       private void addJMenu() {
                                                                                                                                         import java.awt.event.ActionListener;
     currentPerson = teacher;
                                                                                                                                         import java.util.ArrayList;
    updateUI();
                                                                         JMenuBar menuBar;
                                                                                                                                         import java.util.UUID;
                                                                          JMenu menu, submenu;
                                                                         JMenuItem menuItem;
                                                                         JRadioButtonMenuItem rbMenuItem;
                                                                                                                                          * Клас відповідає табові, що відображає розклад
   * метод заповнює інформацію про студента
                                                                         JCheckBoxMenuItem cbMenuItem:
                                                                                                                                         груп
   * @param student екемпляр класу Student
                                                                         menuBar = new JMenuBar();
                                                                                                                                         public class ScheduleTab {
  private void fillInfoFields(Student student) {
                                                                         menu = new JMenu("Файл");
                                                                                                                                            private JFrame frame:
    fillInfoFields(student.getProfile());
                                                                         menu.setMnemonic(KeyEvent.VK_F);
                                                                                                                                            private JComboBox comboBoxFaculty;
                                                                                                                                            private JComboBox comboBoxSubject;
                                                                                                                                            private JComboBox comboBoxTeacher;
mTextFieldInfo4.setText(student.getGroup().getFaculty(
                                                                    menu.getAccessibleContext().setAccessibleDescription(
                                                                               "The only menu in this program that has menu
                                                                                                                                            private JList groupList;
                                                                                                                                           private JTable table;
mTextFieldInfo5.setText(String.valueOf(student.getCourset)) \\
                                                                         menuBar.add(menu);
                                                                                                                                            private java.util.List<Subject> subjects = null;
rse()));
                                                                                                                                            private java.util.List<Teacher> teachers = null;
```

```
private boolean DEBUG = false;
                                                                                                                                                    System.out.println("Cell " + e.getFirstRow() + ".
  private String[] daysOfWeek = new
                                                                          * реагує на зміну розкладу
улучае экінаді акузот учек — не у 
String[[{"Понеділок", "Вівторок", "Середа", 
"Четвер", "П'ятниця", "Субота"};
                                                                           @рагат е подія
                                                                                                                                                          + e.getColumn() + " changed. The new
                                                                                                                                              value:
  private ArrayList<Object[]> data;
                                                                         private void
  private String[] columnNames = {"Пара",
                                                                      table Value Changed Action (Table Model Event\ e)\ \{
                                                                                                                                             table.getModel().getValueAt(e.getFirstRow(),\\
       "Предмет",
                                                                           if (e.getType() == TableModelEvent.UPDATE
                                                                                                                                                         e.getColumn()));
        "Викладач",
                                                                                | e.getType() == TableModelEvent.INSERT
        "Аудиторія",
  private ArrayList<Classes> classes;
                                                                              Group \; g = (Group)
  private Teacher emptyTeacher;
                                                                      groupList.getSelectedValue();
                                                                                                                                                * заповнює таблицю розкладу
  private Subject emptySubject:
                                                                              int row = e.getFirstRow();
   * Конструктор, що приймає елементи інтерфейсу
                                                                              int column = e.getColumn();
                                                                                                                                               private void fillTable() {
для подальших операцій із ними
                                                                              Absence absence = null;
                                                                                                                                                  final\ java.util.List < Classes > classes List = new
   * @рагат frame вікно програми
                                                                                                                                             ArravList<>():
   * @param comboBoxFaculty вибірка факультетів
                                                                                                                                                  Group g = (Group) groupList.getSelectedValue();
                                                                              Classes c = classes.get(row);
   * @param groupList список груп
                                                                              if (c == null) {
                                                                                                                                                  if (g == null) {
   * @param table таблиця розкладу
                                                                                c = new Classes();
                                                                                                                                                    table.setVisible(false);
                                                                                c.setId(UUID.randomUUID());
                                                                                                                                                    return;
  public ScheduleTab(JFrame frame, JComboBox
                                                                                                                                                  } else {
                                                                                absence = new Absence();
                                                                                absence.setId(UUID.randomUUID());
                                                                                                                                                    table.setVisible(true);
comboBoxFaculty, JList\ groupList, JTable\ table)\ \{
                                                                                absence.setGroupId(g.getId());
     this.frame = frame;
     this.comboBoxFaculty = comboBoxFaculty;
                                                                                absence.setClassId(c.getId());
     this.groupList = groupList;
                                                                                c.setTime(row);
                                                                                                                                                  Context c = new Context();
     this.table = table:
                                                                                                                                                  c.put("list", classesList);
                                                                                                                                             c.put("sqlQuery", "select b.* from ABSENCE_TBL a " +
                                                                                   AppConst.DAO.insert(absence);
    hardcoreInit():
                                                                                } catch (DAOException e1) {
                                                                                                                                                       " left join CLASSES_TBL b on (a.class_id =
                                                                                   e1.printStackTrace();
                                                                                                                                             b.classes_id and " +
                                                                                                                                                       " a.group_id = \"" + g.getId() + "\")");
   * ініціалізує компоненти
                                                                                classes.set(row, c);
                                                                                                                                                  try {
  private void hardcoreInit() {
    emptyTeacher = new Teacher();
                                                                              switch (column) {
                                                                                                                                             CommandExecutor.getInstance().execute("loadSchedule
     emptyTeacher.setProfile(new Profile());
                                                                                case 1:
                                                                                                                                              ", c, new Runnable() {
                                                                                  Subject s = (Subject)
                                                                                                                                                       @Override
    emptySubject = new Subject();
                                                                      table.getModel().getValueAt(row,\\
                                                                                                                                                       public void run() {
                                                                                                                                                         MyTableModel\ model = (MyTableModel)
     emptySubject.setName("");
                                                                                       column);
                                                                                   c.setSubject(s):
                                                                                                                                             table.getModel():
    comboBoxFaculty.addActionListener(new
                                                                                  c.setSubjectId(s.getId());
                                                                                                                                                         for (Classes c : classesList) {
                                                                                                                                                            if (c != null) {
ActionListener() {
       @Override
                                                                                   checkAndSave(c);
                                                                                                                                                              classes.set(c.getTime(), c);
       public void actionPerformed(ActionEvent e) {
                                                                                   break:
                                                                                                                                                              int\ row = c.getTime();
         faculty Changed Action ();\\
                                                                                                                                                              if (c.getAudience() > 0) {
                                                                                case 2:
                                                                                                                                                                 model.setValueAt(c.getAudience(),
     });
                                                                                                                                             row, 3);
                                                                                   Teacher t = (Teacher)
     groupList.addListSelectionListener(new
                                                                      table.getModel().getValueAt(row,\\
ListSelectionListener() {
                                                                                       column):
                                                                                                                                             model.set Value At (get Subject By Id (c.get Subject Id ()),\\
       @Override
       public void valueChanged(ListSelectionEvent e)
                                                                                   c.setTeacher(t);
                                                                                   c.setTeacherId(t.getId());
                                                                                                                                             model.set Value At (get Teacher By Id (c.get Teacher Id ()),\\
          reloadTable();
                                                                                   checkAndSave(c);
     });
                                                                                                                                                         updateUI();
                                                                                   break:
     reloadAll();
                                                                                case 3:
                                                                                                                                                    });
                                                                                                                                                  } catch (Exception e) {
                                                                                   Integer i = null;
                                                                                                                                                    e.printStackTrace();
                                                                                   try {
   * перевіряє змінений елемент рзкладу та
зберігає/видаляє його за потреби
                                                                      Integer.parseInt(table.getModel().getValueAt(row,\\
   * @рагат с елемент розкладу
                                                                      column).toString());
                                                                                     if (i < 0) {
  private void checkAndSave(Classes c) {
                                                                                       throw new
                                                                                                                                                * повертає предмет за ключем
                                                                      IllegalArgumentException("Аудиторія повинна бути
                                                                                                                                                * @param id ключ предмету
    try {
       AppConst.DAO.replace(c);
                                                                       >= 0");
                                                                                                                                                * @return предмет
       Group g = (Group)
                                                                                   } catch (NumberFormatException ne) {
                                                                                                                                               private Subject getSubjectById(UUID id) {
groupList.getSelectedValue();
                                                                                     // ok, empty string then
                                                                                   } catch (Exception ex) {
                                                                                                                                                  for (Subject s : subjects) {
       AppConst.SESSION_TIME++;
                                                                                     ex.printStackTrace():
                                                                                                                                                    if \ (s \ != null \ \&\& \ s.getId().equals(id)) \ \{\\
                                                                                     Utils.showErrorDialog(frame,
                                                                                                                                                       return s;
       if (AppConst.SESSION_TIME > 20) {
                                                                      Utils.makePretty(ex.getMessage()));
          History h = new History(
                                                                                     if (absence != null) {
              UUID.randomUUID(),
                                                                                          AppConst.DAO.delete(absence);
                                                                                                                                                  return emptySubject;
AppConst.getCurrentAdmin().getProfileId(),
                                                                                        \}\; catch\; (DAOException\; e1)\; \{
                                                                                          e1.printStackTrace();
               "Вніс зміни до розкладу групи " +
g.getName()
                                                                                                                                                * повертає викладача за ключем
          AppConst.DAO.insert(h);
                                                                                                                                                * @param id ключ викладача
          AppConst.SESSION_TIME = 0;
                                                                                                                                                * @return викладача
                                                                                   c.setAudience(i):
                                                                                                                                                private Teacher getTeacherById(UUID id) {
     } catch (DAOException e) {
                                                                                   checkAndSave(c);
       e.printStackTrace();
                                                                                                                                                  for (Teacher t : teachers) {
                                                                                                                                                    if (t != null && t.getId().equals(id)) {
                                                                                                                                                       return t;
```

}	java.util.List <faculty> faculties = null;</faculty>	subjects = AppConst.DAO.select(Subject.class,
return emptyTeacher;		"");
}	try {	<pre>} catch (DAOException e) { e.printStackTrace();</pre>
/**	faculties = AppConst.DAO.select(Faculty.class,	}
* регує на зміну факультету, завантажує групи ти	"");	,
розклад	} catch (DAOException e) {	comboBoxSubject = new JComboBox();
*/	e.printStackTrace();	Describing the History (and the Carles of Carl
private void facultyChangedAction() { final Faculty f = (Faculty)	}	comboBoxSubject.addItem(emptySubject);
comboBoxFaculty.getSelectedItem();	comboBoxFaculty.setModel(new	for (Subject s : subjects) {
	DefaultComboBoxModel(faculties.toArray()));	comboBoxSubject.addItem(s);
		System.out.println(": : " + s.toString());
java.util.List <group> groups = null;</group>	comboBoxFaculty.setSelectedIndex(0);	}
try {	}	oukiestColumn estCollEditor(new
groups = AppConst.DAO.select(Group.class, new DAOFilter() {		subjectColumn.setCellEditor(new DefaultCellEditor(comboBoxSubject));
@Override	/**	DefaultTableCellRenderer renderer =
<pre>public <t> boolean accept(T entity) {</t></pre>	* створює таблицю розкладу	new DefaultTableCellRenderer();
Group $g = (Group)$ entity;	*/	renderer.setToolTipText("Click for combo box");
return g.getFacultyId().equals(f.getId());	<pre>private void createScheduleTable() {</pre>	subjectColumn.setCellRenderer(renderer);
}	1. (20)	}
}); for (Group a : groups) {	data = new ArrayList<>(30);	/**
for (Group g : groups) { g.setFaculty(f);	for (int $i = 0$; $i < 6$; $i++$) {	* ініціалізує колонку викладачів
}	101 (1111 - 0,1 1 0,1 11)	* @param table таблиця розкладу
} catch (Exception e) {	data.add(new Object[]{daysOfWeek[i], "", "",	* @param teacherColumn колонка розкладу
e.printStackTrace();	""});	*/
}		<pre>public void setUpTeacherColumn(JTable table,</pre>
	for (int $j = 1$; $j < 6$; $j++$) {	TableColumn teacherColumn) {
DefaultListModel <group> model = new</group>	<pre>data.add(new Object[]{new Integer(j), "", "", (Integer) null});</pre>	
DefaultListModel <group>();</group>	(integer) num {);	try {
for (Group g : groups) {	}	teachers = AppConst.DAO.select(Teacher.class
model.addElement(g);	•	"");
}	}	for (Teacher t : teachers) {
groupList.setModel(model);		t.setProfile((Profile)
if (cucumo circo) > 0) (/** * initial right and the control of the control	AppConst.DAO.read(Profile.class, t.getProfileId()));
<pre>if (groups.size() > 0) { groupList.setSelectedIndex(0);</pre>	* ініціалізує розміри колонок таблиці * @param table таблиця розкладу	t.setWorkerInfo((WorkerInfo) AppConst.DAO.read(WorkerInfo.class,
groupEist.setSetectedHdex(0);	*/	t.getWorkerInfoId()));
updateUI();	private void initColumnSizes(JTable table) {	}
}	MyTableModel model = (MyTableModel)	} catch (DAOException e) {
	table.getModel();	e.printStackTrace();
/**	TableColumn column = null;	}
* оновлює інтерфейс */	Component comp = null; int headerWidth = 0;	comboBoxTeacher = new JComboBox();
private void updateUI() {	int cellWidth = 0;	combodox reacher – new reombodox(),
frame.validate();	Object[] longValues = model.longValues;	comboBoxTeacher.addItem(emptyTeacher);
frame.repaint();	TableCellRenderer headerRenderer =	
}	table.getTableHeader().getDefaultRenderer();	for (Teacher s : teachers) {
/**		comboBoxTeacher.addItem(s);
* перезавантажує таблицю розкладу	for (int i = 0; i < 4; i++) { column =	System.out.println(": : " + s.toString()); }
*/	table.getColumnModel().getColumn(i);	teacherColumn.setCellEditor(new
private void reloadTable() {	(-),	DefaultCellEditor(comboBoxTeacher));
•	comp =	DefaultTableCellRenderer renderer =
classes = new ArrayList<>(30);	header Renderer.get Table Cell Renderer Component (<pre>new DefaultTableCellRenderer();</pre>
for (int $i = 0$; $i < 30$; $i++$) {	null, column.getHeaderValue(),	renderer.setToolTipText("Click for combo box");
classes.add(null);	false, false, 0, 0);	teacherColumn.setCellRenderer(renderer);
1	headerWidth = comp.getPreferredSize().width;	1
table.setModel(new MyTableModel());	comp =	
table.set Preferred Scrollable Viewport Size (new	table.getDefaultRenderer (model.getColumnClass(i)).	/**
Dimension(500, 900));	getTableCellRendererComponent(* модель для таблиці
table.setFillsViewportHeight(true);	table, longValues[i],	*/
table.getModel().addTableModelListener(new	false, false, 0, i); cellWidth = comp.getPreferredSize().width;	class MyTableModel extends AbstractTableModel {
TableModelListener() {	cen widin – comp.gen reienredsize().widin,	<pre>public final Object[] longValues = {"Sharon",</pre>
@Override	if (DEBUG) {	"Campione",
<pre>public void tableChanged(TableModelEvent e) {</pre>	System.out.println("Initializing width of	"None of the above",
tableValueChangedAction(e);	column "	new Integer(20), Boolean.TRUE};
}	+ i + ". "	W
<u>});</u>	<pre>+ "headerWidth = " + headerWidth + "; cellWidth = " + cellWidth);</pre>	<pre>public int getColumnCount() { return columnNames.length;</pre>
createScheduleTable();	+ , cen widin = + cen widin),	return columnivames.iengui,
ereaceseneaute ruste(),	ı	j
initColumnSizes(table);		<pre>public int getRowCount() {</pre>
setUpSubjectColumn(table,	column.set Preferred Width (Math.max (header Width,	return data.size();
table.getColumnModel().getColumn(1));	cellWidth));	}
setUpTeacherColumn(table,	}	militie Coding and Colombia Managing and C
table.getColumnModel().getColumn(2));)	<pre>public String getColumnName(int col) { return columnNames[col];</pre>
fillTable();	/**	return columninames[col];
}	* ініціалізує колонку предметів	,
	* @param table таблиця розкладу	<pre>public Object getValueAt(int row, int col) {</pre>
/**	* @param subjectColumn колонка розкладу	return data.get(row)[col];
* перезавантажує всі компоненти	*/	}
*/ private void relead AUO (public void setUpSubjectColumn(JTable table,	public Class setCaluma Class Cart at 1
private void reloadAll() {	TableColumn subjectColumn) {	<pre>public Class getColumnClass(int c) { return getValueAt(0, c).getClass();</pre>
reloadTable();		}
	try {	

```
public boolean isCellEditable(int row, int col) {
                                                                         comboBox.setModel(new
       if (col < 1 || row \% 6 == 0) {
                                                                                                                                         import javax.swing.*;
                                                                                                                                         import javax.swing.text.JTextComponent;
                                                                    DefaultComboBoxModel(new Object[]{"Власна",
                                                                                                                                         import java.awt.event.ActionEvent;
                                                                     "Уся"}));
                                                                         comboBox.addActionListener(new
                                                                                                                                         import java.awt.event.ActionListener;
       return true;
                                                                     ActionListener() {
                                                                            @Override
    public void setValueAt(Object value, int row, int
                                                                           public void actionPerformed(ActionEvent e) {
                                                                                                                                          * Клас відповідає за вікно, що здійснює авторизацію
                                                                             comboboxChangedAction();
                                                                                                                                         користувача
       if (DEBUG) {
         System.out.println("Setting value at " + row +
                                                                         });
                                                                                                                                         public class LoginFrame extends JFrame implements
"." + col
                                                                                                                                         ActionListener {
              + " to " + value
                                                                         trv {
                                                                           List<Profile> list =
              + " (an instance of "
                                                                                                                                           private JTextField userText;
              + value.getClass() + ")");
                                                                    AppConst.DAO.select(Profile.class, "");
                                                                                                                                           private JPasswordField passwordText;
                                                                            for (Profile p : list) {
                                                                             profiles.put(p.getId(), p);
                                                                                                                                           private String userName = null;
       data.get(row)[col] = value:
                                                                                                                                           private String userPassword = null;
       fireTableCellUpdated(row, col);
                                                                         } catch (DAOException e) {
                                                                           e.printStackTrace();
       if (DEBUG) {
                                                                                                                                            * Конструктор форми
         System.out.println("New value of data2:");
         printDebugData();
                                                                         comboBox.setSelectedIndex(0);
                                                                                                                                           public LoginFrame() {
                                                                                                                                             super((String)
                                                                                                                                         AppConst.APP_CONFIG.getValue("progTitle"));
                                                                                                                                              JPanel panel = new JPanel();
     private void printDebugData() {
                                                                       * реагує на зміну опції Власна/Уся історія
                                                                                                                                              add(panel);
       int numRows = getRowCount();
                                                                                                                                              placeComponents(panel);
       int\ numCols = getColumnCount();
                                                                      private void comboboxChangedAction() {
                                                                                                                                              setLocationRelativeTo(null);
       for (int i = 0; i < numRows; i++) {
                                                                         switch (comboBox.getSelectedIndex()) {
         System.out.print(" row " + i + ":");
          for (int j = 0; j < numCols; j++) {
                                                                             loadHistory("select * from HISTORY_TBL
                                                                                                                                            * ініціалізує компоненти
            System.out.print(" "+ data.get(i)[j]);\\
                                                                    where author_id = \"" +
                                                                                                                                            * @param panel панель, на якуі помістити
         System.out.println();
                                                                     AppConst.getCurrentAdmin().getProfileId() + "\";");
                                                                                                                                         компоненти
                                                                              break;
                                                                                                                                           private void placeComponents(JPanel panel) {
                                                                           case 1:
                                                                                                                                              panel.setLayout(null);
                                                                              loadHistory("");
                                                                                                                                              JLabel userLabel = new JLabel("Username");
package com.ovster.ui:
                                                                              break:
                                                                                                                                              userLabel.setBounds(10, 10, 80, 25);
                                                                         }
                                                                                                                                              panel.add(userLabel);
import com.oyster.app.AppConst;
import com.oyster.app.model.History;
                                                                                                                                              userText = new ITextField(20):
import com.oyster.app.model.Profile;
import com.oyster.core.controller.CommandExecutor;
                                                                                                                                              userText.setBounds(100, 10, 170, 25);
                                                                       * Завантажує історію користувача, виконуюючи
import com.oyster.core.controller.command.Context;
                                                                                                                                              panel.add(userText);
import com.oyster.dao.exception.DAOException;
                                                                    sql-запит
                                                                                                                                              JLabel passwordLabel = new JLabel("Password");
import javax.swing.*;
                                                                       * @param sql SQL-запит для виконання
                                                                                                                                              passwordLabel.setBounds(10, 40, 80, 25);
import java.awt.event.ActionEvent;
                                                                                                                                              panel.add(passwordLabel);
                                                                      private void loadHistory(String sql) {
import java.awt.event.ActionListener;
import java.util.*;
                                                                                                                                              passwordText = new JPasswordField(20);
                                                                         Context c = new Context();
                                                                                                                                              passwordText.setBounds(100, 40, 170, 25);
                                                                         histories = new ArrayList<>();
                                                                                                                                              panel.add(passwordText);
* Клас відповідає табові, що відображає історію
                                                                         c.put("list", histories);
коритсувачів
                                                                         c.put("sqlQuery", sql);
                                                                                                                                              JButton loginButton = new JButton("login");
                                                                                                                                              loginButton.setBounds(160, 80, 100, 25);
public class HistoryTab {
                                                                                                                                              loginButton.addActionListener(this);
                                                                                                                                             panel.add(loginButton);
  private JFrame frame;
                                                                    Command Executor.get Instance (). execute ("load History", \\
  private JComboBox comboBox:
                                                                    c, new Runnable() {
  private JList historyList;
                                                                              @Override
                                                                                                                                            * спрацьовує на подію е
                                                                             public void run() {
  java.util.List<History> histories;
                                                                                System.out.println(histories.size());
                                                                                                                                            * @рагат е подія
  private Map<UUID, Profile> profiles;
                                                                                for (History h : histories) {
                                                                                                                                           @Override
                                                                                                                                           public void actionPerformed(ActionEvent e) {
   * Конструктор, що приймає елементи інтерфейсу
                                                                    h.setAuthor(profiles.get(h.getAuthorId()));\\
для подальших операцій із ними
                                                                                DefaultListModel<History> model = new
                                                                                                                                              StringBuilder errorMsg = new
                                                                    StringBuilder("Введіть ");
   * @param frame
                      вікно програми
                                                                                for (History h : histories) {
   * @рагат сотвоВох елемент вибору типу
                                                                                                                                              boolean errorOccurred = false:
                                                                                  model.addElement(h);
                                                                                                                                              JTextComponent focusComponent = userText;
історії
   * @param historyList список історії
                                                                                                                                              userName = "admin-root";//userText.getText();
  public HistoryTab(JFrame frame, JComboBox
                                                                                                                                              userPassword = "admin";//new
                                                                                historyList.setModel(model);
                                                                                frame.validate();
comboBox, JList historyList) {
                                                                                                                                         String(passwordText.getPassword());\\
     this.frame = frame;
                                                                                frame.repaint();
                                                                                                                                              if (userName.trim().length() == 0) {
     this.comboBox = comboBox:
    this.historyList = historyList;
                                                                                                                                                errorMsg.append(" логін");
                                                                           });
                                                                         } catch (Exception e) {
                                                                                                                                                errorOccurred = true;
                                                                           e.printStackTrace();
    profiles = new HashMap<>();
                                                                                                                                             if \ (userPassword.trim().length() == 0) \ \{\\
    hardcoreInit():
                                                                      }
                                                                                                                                                if (errorOccurred) {
                                                                    package com.oyster.ui;
                                                                                                                                                  errorMsg.append(" та");
   * ініціалізує компоненти
                                                                    import\ com. oyster. app. App Const;
                                                                                                                                                errorOccurred = true;
                                                                    import\ com. oyster. core. controller. Command Executor;
                                                                                                                                                errorMsg.append(" пароль");
  private void hardcoreInit() {
                                                                    import com.oyster.core.controller.command.Context;
                                                                                                                                                focusComponent = passwordText;
```

}*/	private JTextField mTextFieldInfo5;	@Override
1/11/11	private JTextField mTextFieldInfo6;	public void actionPerformed(ActionEvent e) {
errorMsg.append("!");	private JTextField mTextFieldInfo7;	newUserAction();
	private JButton mButtonNewUser;	}
if (!errorOccurred) {	private JComboBox mComboBoxAllHistory;	});
tryToLogIn();	private JList mListAllHistory;	comboBoxChangeAction();
} else {	private JComboBox mComboBoxScheduleFaculty;	mComboBoxProfileType.addActionListener(new
focusComponent.selectAll();	private JScrollPane mListGroups;	ActionListener() {
JOptionPane.showMessageDialog(private JTable mTable1;	@Override
LoginFrame.this,	private JLabel mLabel5;	public void actionPerformed(ActionEvent e) {
		· · · · · · · · · · · · · · · · · · ·
errorMsg.toString(),	private JLabel mLable6;	comboBoxChangeAction();
"Спробуйте ще раз",	private JLabel mLable7;	}
JOptionPane.ERROR_MESSAGE	private JLabel mLable1;	});
);	private JLabel mLable2;	
userName = null;	private JLabel mLable3;	mListPeople.addListSelectionListener(new
userPassword = null;	private JLabel mLablePhoto;	ListSelectionListener() {
	*	
focusComponent.requestFocusInWindow();	private JLabel mLable8;	@Override
}	private JPasswordField mPasswordField1;	public void valueChanged(ListSelectionEvent e)
}	private JLabel mLable4;	{
	private JTextField mTextFieldInfo4;	Object o = mListPeople.getSelectedValue();
/**	private JScrollPane mScrollPanePeople;	if (o instanceof Admin) {
* викликає команду здійснення авторизації	private JList mListTab2Group;	fillInfoFields((Admin) o);
· · · · · · · · · · · · · · · · · · ·	private 3Eist mEist 1a02Group,	
*/		} else if (o instanceof Teacher) {
private void tryToLogIn() {	private IProfile currentPerson;	fillInfoFields((Teacher) o);
		} else if (o instanceof Student) {
Context $c = \text{new Context}();$	private DocumentListener mDocumentListener = new	fillInfoFields((Student) o);
c.put("username", userName);	DocumentListener() {	1
-)
<pre>c.put("password", userPassword);</pre>	@Override	. 1
	<pre>public void insertUpdate(DocumentEvent e) {</pre>	});
try {	act();	
	}	
CommandExecutor.getInstance().execute("logIn", c,	•	mComboBoxProfileStudentTypeFaculty.addActionListo
new Runnable() {	@Override	ner(new ActionListener() {
		· · · · · · · · · · · · · · · · · · ·
@Override	<pre>public void removeUpdate(DocumentEvent e) {</pre>	@Override
public void run() {	act();	<pre>public void actionPerformed(ActionEvent e) {</pre>
LoginFrame.this.dispose();	}	facultyChangedAction();
)
ì)·
,	00 11	}) ;
<pre>});</pre>	@Override	
} catch (Exception e) {	<pre>public void changedUpdate(DocumentEvent e) {</pre>	
e.printStackTrace();	act();	mComboBoxProfileStudentTypeGroup.addActionLister
}	}	er(new ActionListener() {
}	,	@Override
1	private void act() {	public void actionPerformed(ActionEvent e) {
package com.oyster.ui;	mButtonSave.setEnabled(true);	groupChangedAction();
	}	}
import com.oyster.app.AppConst;	};	<pre>});</pre>
import com.oyster.app.model.*;	,,	177
	' . II'	
import com.oyster.core.controller.CommandExecutor;	private HistoryTab historyTab;	
import com.oyster.core.controller.command.Context;	private ScheduleTab scheduleTab;	mTextFieldInfo1.getDocument().addDocumentListener
import com.oyster.dao.DAOFilter;		mDocumentListener);
import com.oyster.dao.exception.DAOException;		
	/**	mTextFieldInfo2.getDocument().addDocumentListener
import com.oyster.ui.dialogs.*;	,	
	 конструктор, створює елементи інтерфейсу 	mDocumentListener);
import javax.swing.*;	*/	
import javax.swing.event.DocumentEvent;	public MainForm() {	mTextFieldInfo3.getDocument().addDocumentListener
import javax.swing.event.DocumentListener;	super((String)	mDocumentListener);
	* · · · · · · · · · · · · · · · · · · ·	,,
import javax.swing.event.ListSelectionEvent;	AppConst.APP_CONFIG.getValue("progTitle"));	T T T T T T T T T T T T T T T T T T T
import javax.swing.event.ListSelectionListener;		mTextFieldInfo4.getDocument().addDocumentListener
import javax.swing.text.DateFormatter;	add(rootPanel);	mDocumentListener);
import java.awt.*;		
import java.awt.event.ActionEvent;	int width = (Integer)	mTextFieldInfo5.getDocument().addDocumentListener
import java.awt.event.ActionListener;	AppConst.APP_CONFIG.getValue("mainScreenWidth"	mDocumentListener);
	•	inDocumentListener),
import java.awt.event.KeyEvent;);	
import java.text.ParseException;	int height = (Integer)	mTextFieldInfo6.getDocument().addDocumentListener
import java.util.ArrayList;	AppConst.APP_CONFIG.getValue("mainScreenHeight"	mDocumentListener);
import java.util.Date;);	
1 7	<i>"</i>	mTextFieldInfo7.getDocument().addDocumentListener
	setPreferredSize(new Dimension(width, height));	mDocumentListener);
At at		inDocumentListener),
/**	setMinimumSize(new Dimension(width, height));	
 * Клас відповідає табові, що реалізує вікно 		mPasswordField1.getDocument().addDocumentListene
кправління профілями	hardCoreInit();	(mDocumentListener);
*		
* @author bamboo	pack();	mButtonSave.addActionListener(new
	pack(),	
*/		ActionListener() {
public class MainForm extends JFrame {		@Override
	setDefaultCloseOperation(WindowConstants.EXIT_ON	public void actionPerformed(ActionEvent e) {
private JPanel rootPanel;	_CLOSE);	saveButtonClick();
private JTabbedPane mTabbedPaneMain;	_=====/;	1
•	and and an D. L. S. W. C. W.	1
private JComboBox mComboBoxProfileType;	setLocationRelativeTo(null);	});
private JComboBox		
mComboBoxProfileStudentTypeFaculty;	setVisible(true);	mButtonDelete.addActionListener(new
private JList mListPeople;	}	ActionListener() {
	J	
private JComboBox		@Override
mComboBoxProfileStudentTypeGroup;		<pre>public void actionPerformed(ActionEvent e) {</pre>
private JPanel mPanelProfile;	/**	deleteButtonClick();
private JPanel mPanelLeftControl;	* ініціалізує елементи інтерфейсу	}
private JButton mButtonSave;	*/)·
•));
private JButton mButtonDelete;	private void hardCoreInit() {	
private JList mListHistoryProfile;		scheduleTab = new ScheduleTab(this,
private JTextField mTextFieldInfo1;	addJMenu();	mComboBoxScheduleFaculty, mListTab2Group,
private JTextField mTextFieldInfo2;	mButtonNewUser.addActionListener(new	mTable1);
= -		**
private JTextField mTextFieldInfo3;	ActionListener() {	

historyTab = new HistoryTab(this,		@Override
mComboBoxAllHistory, mListAllHistory);	try {	<pre>public <t> boolean accept(T entity) {</t></pre>
}	AppConst.DAO.update(s);	Group $g = (Group)$ entity;
	AppConst.DAO.update(s.getGroup());	return g.getFacultyId().equals(f.getId());
/**		}
* метод спрацьовує при натисненні на кнопку	AppConst.DAO.update(s.getGroup().getFaculty());	});
"Видалити"	} catch (DAOException e) {	for (Group g : groups) {
*/	e.printStackTrace();	g.setFaculty(f);
private void deleteButtonClick() {	}	}
	}	} catch (Exception e) {
if (currentPerson == null) {		e.printStackTrace();
return;	/**	}
}	* зберігає інформацію про профіль	
	*	mComboBoxProfileStudentTypeGroup.setModel(new
int dialogResult =	* @param p екземпляр класу Profile	DefaultComboBoxModel(groups.toArray()));
JOptionPane.showConfirmDialog(null,	*/	if $(groups.size() > 0)$ {
"Підтвердити видалення акаунту: ",	private void saveUpdate(Profile p) {	
"Захист від дурака",	String firstName =	mComboBoxProfileStudentTypeGroup.setSelectedIndex
JOptionPane.OK_CANCEL_OPTION,	mTextFieldInfo1.getText().trim();	(0);
JOptionPane.WARNING_MESSAGE	String secondName =	}
);	mTextFieldInfo2.getText().trim();	updateUI();
	String password =	}
if (dialogResult ==	mPasswordField1.getText().trim();	
JOptionPane.CANCEL_OPTION) {	String birthday = mTextFieldInfo3.getText().trim();	/**
return;	Long birthdayLong = 0L;	* метод спрацьовує при зміні групи
}	try {	*/
,	Date d = (Date) new	private void groupChangedAction() {
Context $c = new Context()$;	DateFormatter(AppConst.DATE_FORMAT).stringToV	final Group group = (Group)
c.put("profile", currentPerson);	alue(birthday);	mComboBoxProfileStudentTypeGroup.getSelectedItem(
e.put(prome , current erson),	birthdayLong = d.getTime(););
tray (} catch (ParseException e) {	
try {	The state of the s	java.util.List <student> students = null;</student>
	e.printStackTrace();	try {
CommandExecutor.getInstance().execute("deleteIProfile	}	students = AppConst.DAO.select(Student.class,
", c, null);	<pre>p.setFirstName(firstName);</pre>	new DAOFilter() {
} catch (Exception e) {	<pre>p.setSecondName(secondName);</pre>	@Override
e.printStackTrace();	<pre>p.setBirthday(birthdayLong);</pre>	<pre>public <t> boolean accept(T entity) {</t></pre>
}	p.setPassword(password);	Student $s = (Student)$ entity;
	try {	return s.getGroupId().equals(group.getId());
currentPerson = null;	AppConst.DAO.update(p);	}
,	} catch (DAOException e) {	});
comboBoxChangeAction();	e.printStackTrace();	for (Student student : students) {
tomboboxchangeAction(),	c.prinistack race(),	student.setProfile((Profile)
ı	1	
/**	1	AppConst.DAO.read(Profile.class,
•	All the	student.getProfileId()));
* метод спрацьовує при натисненні на кнопку	/**	student.setGroup(group);
"Зберегти"	* зберігає інформацію про роботу	}
*/	*	} catch (Exception e) {
private void saveButtonClick() {	* @param wi екземпляр класу WorkerInfo	e.printStackTrace();
	*/	}
if (currentPerson == null) {	private void saveUpdate(WorkerInfo wi) {	showFilteredStudents(students);
return;	String position = mTextFieldInfo4.getText().trim();	}
}	String salary = mTextFieldInfo5.getText().trim();	
,	String dateHiredStr =	/**
if (currentPerson instanceof Admin) {	mTextFieldInfo6.getText().trim();	* метод відображає список студентів, що
Admin a = (Admin) currentPerson;	Long dateHired = 0L;	пройшли фільтри, встановлені користувачем
saveUpdate(a.getProfile());	-	проишли фільтри, встановлент користувачем *
	try { Date d = (Date) new	* Charana atudanta arrivaari arrivaariin
saveUpdate(a.getWorkerInfo());	· · · · · · · · · · · · · · · · · · ·	* @param students список студентів
	DateFormatter(AppConst.DATE_FORMAT).stringToV	*/
} else if (currentPerson instanceof Teacher) {	alue(dateHiredStr);	private void
Teacher t = (Teacher) currentPerson;	dateHired = d.getTime();	showFilteredStudents(java.util.List <student> students) {</student>
saveUpdate(t.getProfile());	} catch (ParseException e) {	DefaultListModel <student> model = new</student>
<pre>saveUpdate(t.getWorkerInfo());</pre>	e.printStackTrace();	DefaultListModel <student>();</student>
	}	for (Student s : students) {
} else if (currentPerson instanceof Student) {	wi.setPosition(position);	model.addElement(s);
Student s = (Student) currentPerson;	wi.setSalary(Integer.parseInt(salary));	}
saveUpdate(s);	wi.setDateHired(dateHired);	mListPeople.setModel(model);
}	try {	
updateUI();	AppConst.DAO.update(wi);	if $(students.size() > 0)$ {
}	} catch (DAOException e) {	mListPeople.setSelectedIndex(0);
j	e.printStackTrace();	}
		,
/**	}	updateUI();
•	}	}
* зберігає інформацію про профіль студента	All the	All de
*	/**	/**
* @param s екземпляр класу Student	* оновлює інтерфейс	* метод спрацьовує при зміні типу користувача
*/	*/	*/
private void saveUpdate(Student s) {	private void updateUI() {	private void comboBoxChangeAction() {
saveUpdate(s.getProfile());	mButtonSave.setEnabled(false);	
	validate();	mButtonSave.setEnabled(false);
String course = mTextFieldInfo5.getText().trim();	repaint();	int selected =
String bookNum =	}	mComboBoxProfileType.getSelectedIndex();
mTextFieldInfo7.getText().trim();	,	if (selected < 3) {
mread foldings, got reat(), unit(),	/**	II (Science < 3) (
String facultyName -	,	mComboRovProfileStudentTimeEnables antVisite1-(6-1
String facultyName = mToxtFieldInfo4 gotToxt() trim();	* метод спрацьовує при зміні факультету */	mComboBoxProfileStudentTypeFaculty.setVisible(false
mTextFieldInfo4.getText().trim();	*/);
String groupName =	private void facultyChangedAction() {	G 1 D D 7 C 1 7 T
mTextFieldInfo6.getText().trim();	final Faculty $f = (Faculty)$	mComboBoxProfileStudentTypeGroup.setVisible(false)
	mComboBoxProfileStudentTypeFaculty.getSelectedIte	;
s.setCourse(Integer.parseInt(course));	m();	mLable7.setVisible(false);
s.setBookNum(Integer.parseInt(bookNum));	<pre>java.util.List<group> groups = null;</group></pre>	mTextFieldInfo7.setVisible(false);
	try {	mLable4.setText("Посада");
s.getGroup().setName(groupName);	groups = AppConst.DAO.select(Group.class,	mLabel5.setText("Зарплата");
c cotGroup() cotEcoulty() cotNomo(focultyNomo);	now DAOFilson() (mI abla6 satTaxt("Ilpaniac ia"):

	try {	}
} else {	faculties =	/**
	AppConst.DAO.select(Faculty.class, ""); } catch (DAOException e) {	 * метод заповнює інформацію про адміністратора
mComboBoxProfileStudentTypeFaculty.setVisible(true)	e.printStackTrace();	*
;	}	* @param admin екемпляр класу Admin */
mComboBoxProfileStudentTypeGroup.setVisible(true);		private void fillInfoFields(Admin admin) {
mLable7.setVisible(true);	mComboBoxProfileStudentTypeFaculty.setModel(new	fillInfoFields(admin.getProfile());
mTextFieldInfo7.setVisible(true);	Default Combo Box Model (faculties. to Array ()));	fillInfoFields(admin.getWorkerInfo());
mLable4.setText("Факультет");		.D
mLabel5.setText("Курс"); mLable6.setText("Група");	mComboBox ProfileStudent TypeFaculty. set Selected Inde	<pre>currentPerson = admin; updateUI();</pre>
mLable7.setText("H3K");	x(0);	update 01(),
}		,
switch	break;	/**
(mComboBoxProfileType.getSelectedIndex()) {	}	 * метод заповнює інформацію про викладача
<pre>case 0: mScrollPanePeople.setVisible(false);</pre>	updateUI();	* @param teacher екемпляр класу Teacher
mButtonDelete.setEnabled(false);	j	*/
fillInfoFields(AppConst.getCurrentAdmin());	/**	private void fillInfoFields(Teacher teacher) {
break;	* метод заповнює інформацію про профіль	fillInfoFields(teacher.getProfile());
case 1:	*	fillInfoFields(teacher.getWorkerInfo());
mScrollPanePeople.setVisible(true);	 * @param profile екемпляр класу Profile */ 	currentPerson = teacher;
mButtonDelete.setEnabled(true);	private void fillInfoFields(Profile profile) {	updateUI();
	mTextFieldInfo1.setText(profile.getFirstName());	}
java.util.List <admin> admins = null;</admin>		
	mTextFieldInfo2.setText(profile.getSecondName());	/**
try { admins =	mTextFieldInfo3.setText(AppConst.DATE_FORMAT.f	* метод заповнює інформацію про студента *
AppConst.DAO.select(Admin.class, "");	ormat(new Date(profile.getBirthday())));	* @param student екемпляр класу Student
for (Admin a : admins) {	mPasswordField1.setText(profile.getPassword());	*/
a.setProfile((Profile)		private void fillInfoFields(Student student) {
AppConst.DAO.read(Profile.class, a.getProfileId()));	loadHistory(profile);	fillInfoFields(student.getProfile());
a.setWorkerInfo((WorkerInfo)	}	manuficial district and Tout (at a double and Common and Complete)
AppConst.DAO.read(WorkerInfo.class, a.getWorkerInfoId()));	/**	mTextFieldInfo4.setText(student.getGroup().getFaculty().getName());
}	* метод завантажує історію вибраного).gen (mine()),
} catch (DAOException e) {	користувача	mTextFieldInfo5.setText(String.valueOf(student.getCou
e.printStackTrace();	*	rse()));
}	* @param profile користувач	Total College Control (state of Control College)
DefaultListModel <admin> model = new</admin>	private void loadHistory(final Profile profile) {	mTextFieldInfo6.setText(student.getGroup().getName()) .
DefaultListModel<>();	private void roadristory(mai i rome prome) (,
for (Admin a : admins) {	final java.util.List <history> histories = new</history>	mTextFieldInfo7.setText(String.valueOf(student.getBoo
model.addElement(a);	ArrayList <history>();</history>	kNum()));
} mListPeople.setModel(model);	Context c = new Context();	currentPerson = student;
initistr eopie.setwiodei(modei),	c.put("list", histories);	updateUI();
mListPeople.setSelectedIndex(0);	c.put("sqlQuery", "select * from HISTORY_TBL	}
	where author_id = \"" +	
break;	profile.getId() + "\";");	/**
case 2:	try {	* метод спрацьовує при натисненні на клавішу"Додати аккаунт"
case 2.	шу (*/
mScrollPanePeople.setVisible(true);	CommandExecutor.getInstance().execute("loadHistory",	private void newUserAction() {
mButtonDelete.setEnabled(true);	c, new Runnable() {	Object[] possibilities = {"Студент", "Виладач",
	@Override	"Адміністратор", "Група", "Факультет", "Предмет"};
java.util.List <teacher> teachers = null;</teacher>	<pre>public void run() { for (History h : histories) {</pre>	String s = (String) JOptionPane.showInputDialog(MainForm.this,
try {	h.setAuthor(profile);	"Виберіть категорію, яку хочете
teachers =	}	створити:\n",
AppConst.DAO.select(Teacher.class, "");	DefaultListModel < History > history Model =	"Новий акаунт",
for (Teacher teachers: teachers) {	new DefaultListModel<>();	JOptionPane.PLAIN_MESSAGE,
teacher.setProfile((Profile) AppConst.DAO.read(Profile.class,	<pre>for (History h : histories) { historyModel.addElement(h);</pre>	null, possibilities,
teacher.getProfileId()));	instorywoder.addEienen(n),	роззілінся, "Студент");
teacher.setWorkerInfo((WorkerInfo)	•	· "
AppConst.DAO.read(WorkerInfo.class,		if $((s != null) && (s.length() > 0)) {$
teacher.getWorkerInfoId()));	mListHistoryProfile.setModel(historyModel);	
} } catch (DAOException e) {	} });	JDialog dialog = null; switch (s) {
e.printStackTrace();	} catch (Exception e) {	саѕе "Студент":
}	e.printStackTrace();	,
	}	dialog = new
DefaultListModel <teacher> teacherModel =</teacher>	}	NewStudentCustomDialog(this);
new DefaultListModel<>(); for (Teacher teacher: teachers) {	/**	break;
teacher Model.addElement(teacher);	* метод заповнює інформацію про роботу	case "Виладач":
}	*	dialog = new
mListPeople.setModel(teacherModel);	* @param workerInfo екемпляр класу WorkerInfo	NewTeacherCustomDialog(this);
mListPeople.setSelectedIndex(0);	*/	break;
break;	private void fillInfoFields(WorkerInfo workerInfo) {	case "Адміністратор":
oreax,	mTextFieldInfo4.setText(workerInfo.getPosition());	radminicipatop .
case 3:		dialog = new
	mTextFieldInfo5.setText(String.valueOf(workerInfo.get	NewAdminCustomDialog(this);
mScrollPanePeople.setVisible(true);	Salary()));	break;
mButtonDelete.setEnabled(true);	mTextFieldInfo6.setText(AppConst.DATE_FORMAT.f	case "Група":
java.util.List <faculty> faculties = null;</faculty>	ormat(new Date(workerInfo.getDateHired())));	py

dialog = new NewGroupCustomDialog(this);	setJMenuBar(menuBar);	String msgString1 = "Ім'я адміністратора: "; String msgString2 = "Прізвище адміністратора :
break;	}	",
case "Факультет":	} package com.oyster.ui.dialogs;	String msgString3 = "День народження: "; String msgString4 = "Посада: ";
dialog — marri	imment com existen onn Ann Court.	String msgString5 = "Зарплата / рік : "; String msgString6 = "Дата прийняття на роботу :
dialog = new NewFacultyCustomDialog(this);	<pre>import com.oyster.app.AppConst; import com.oyster.core.controller.CommandExecutor;</pre>	зинід ніздэніндо — дата прииняття на росоту . ";
break;	import com.oyster.core.controller.command.Context;	String msgString7 = "Пароль : ";
case "Предмет":	import javax.swing.*;	Object[] array = {msgString1, textField1,
21.1	import javax.swing.text.JTextComponent;	msgString2, textField2,
dialog = new NewSubjectCustomDialog(this);	import javax.swing.text.NumberFormatter;	msgString3, textField3, msgString4, textField4,
break;	<pre>import java.awt.*; import java.awt.event.ComponentAdapter;</pre>	msgString5, textField5,
}	import java.awt.event.ComponentEvent;	msgString6, textField6,
dialog.pack();	import java.awt.event.WindowAdapter;	msgString7, mJPasswordField};
dialog.setVisible(true);	import java.awt.event.WindowEvent;	<i>5.</i> , , ,
}	import java.beans.PropertyChangeEvent;	Object[] options = {btnString1, btnString2};
}	import java.beans.PropertyChangeListener;	
	import java.text.NumberFormat;	optionPane = new JOptionPane(array,
/** *····	import java.text.ParseException;	JOptionPane.INFORMATION_MESSAGE,
* ініціалізує меню у програмі */	import java.util.Date;	JOptionPane.YES_NO_OPTION,
private void addJMenu() {		null, options,
private void addiviend() (/**	options[0]);
JMenuBar menuBar;	* Клас відповідає за діалог, який збирає усі дані	setContentPane(optionPane);
JMenu menu, submenu;	необхідні для створення нового адміністратора	
JMenuItem menuItem;	*/	
JRadioButtonMenuItem rbMenuItem;	public class NewAdminCustomDialog extends JDialog	setDefaultCloseOperation(DO_NOTHING_ON_CLOS
JCheckBoxMenuItem cbMenuItem;	implements PropertyChangeListener {);
		addWindowListener(new WindowAdapter() {
menuBar = new JMenuBar();	private String userName = null;	public void windowClosing(WindowEvent we)
monu = nov. IMonu("dbo%"").	private String userSurmane = null;	optionPane.setValue(new Integer(
menu = new JMenu("Файл"); menu.setMnemonic(KeyEvent.VK_F);	private String userBirthDateStr = null;	JOptionPane.CLOSED_OPTION));
menu.setvineniome(recyzvent. v r_1),	private Date userBirthDate = null;	});
menu.getAccessibleContext().setAccessibleDescription(1	addComponentListener(new ComponentAdapter()
"The only menu in this program that has menu	private String adminPosition;	{
items");		public void componentShown(ComponentEvent
menuBar.add(menu);	private String adminSalary;	ce) {
menuItem = new JMenuItem("Додати акаунт",	private Integer adminSalaryInt;	<pre>textField1.requestFocusInWindow(); }</pre>
KeyEvent	private String adminDateHiredStr;	});
.VK_N	private Date adminDateHired;	optionPane.addPropertyChangeListener(this);
);	-	setPreferredSize(new Dimension(350, 450));
menuItem.setMnemonic(KeyEvent.VK_N);	private String userPassword;	setMinimumSize(new Dimension(350, 450));
menuItem.setAccelerator(KeyStroke.getKeyStroke(}
KeyEvent.VK_N,	private JTextField textField1;	
ActionEvent.ALT_MASK));	private JTextField textField2;	/**
menuItem.addActionListener(new ActionListener()	private JTextField textField3; private JTextField textField4;	* Метод спрацьовує на зміну властивостей у
{ @Override	private JTextField textField4; private JTextField textField5;	JOptionsPane *
public void actionPerformed(ActionEvent e) {	private JTextField textField6;	* @рагат е дані про подію
newUserAction();	private JPasswordField mJPasswordField;	*/
}	•	public void propertyChange(PropertyChangeEvent e)
});	private JOptionPane optionPane;	{
menu.add(menuItem);		String prop = e.getPropertyName();
	private String btnString1 = "Створити";	10.0 XV VI A
menu.addSeparator();	private String btnString2 = "Відмінити";	if (isVisible()
menuItem = new JMenuItem("Вихід",	/**	&& (e.getSource() == optionPane) &&
new ImageIcon("images/middle.gif"));	* Створює нове ділогове вікно	(JOptionPane.VALUE_PROPERTY.equals(prop)
menuItem.setMnemonic(KeyEvent.VK_W);	*	(boptom and vineed_i Not Ext inequals(prop)
menuItem.setAccelerator(KeyStroke.getKeyStroke(* @рагат аFrame вкно, що викликало діалог 	JOptionPane.INPUT_VALUE_PROPERTY.equals(pro
KeyEvent.VK_W,	*/))) {
ActionEvent.ALT_MASK));	<pre>public NewAdminCustomDialog(Frame aFrame) {</pre>	Object value = optionPane.getValue();
menuItem.addActionListener(new ActionListener()	super(aFrame, true);	:67
{ @Override	super.setLocationRelativeTo(null);	if (value ==
public void actionPerformed(ActionEvent e) {	setTitle("Створити профіль адміністратора");	JOptionPane.UNINITIALIZED_VALUE) { return;
MainForm.this.dispose();	зеттие Створити профиль админетратора),	}
}		optionPane.setValue(
});	textField1 = new JTextField(10);	JOptionPane.UNINITIALIZED_VALUE);
menu.add(menuItem);	textField2 = new JTextField(15);	
	textField3 = new	if (btnString1.equals(value)) {
menu = new JMenu("Допомога");	JFormattedTextField(AppConst.DATE_FORMAT);	userName = textField1.getText().trim();
menu.setMnemonic(KeyEvent.VK_H);	textField4 = new JTextField(15);	userSurmane = textField2.getText().trim();
menuBar.add(menu);	NumberFormat format =	userBirthDateStr = textField3.getText().trim() adminPosition = textField4.getText().trim();
menuItem = new JMenuItem("Посібник	NumberFormat format = NumberFormat.getInstance();	adminSalary = textField5.getText().trim();
користувача",	NumberFormatter formatter = new	adminDateHiredStr =
KeyEvent.VK_K);	NumberFormatter(format);	textField6.getText().trim();
menuItem.setAccelerator(KeyStroke.getKeyStroke(formatter.setValueClass(Integer.class);	userPassword = new
KeyEvent.VK_K,	formatter.setMinimum(1);	String(mJPasswordField.getPassword()).trim();
ActionEvent.ALT_MASK));	formatter.setMaximum(Integer.MAX_VALUE);	
menu.add(menuItem);	formatter.setCommitsOnValidEdit(true);	boolean errorOccured = false;
monultons = nov. D.f(UIT	textField5 = new JFormattedTextField(formatter);	JTextComponent focusComponent =
menuItem = new JMenuItem("Про програму"); menuItem.setAccelerator(KeyStroke.getKeyStroke(textField6 = new	textField1;
KeyEvent.VK_A,	JFormattedTextField(AppConst.DATE_FORMAT);	
ActionEvent.ALT_MASK));	mJPasswordField = new JPasswordField(15);	StringBuilder errorMsg = new
menu.add(menuItem);		StringBuilder("Введіть ");

```
ex.printStackTrace();
         if (userName.length() == 0) {
           errorMsg.append(" ім'я адміністратора");
            errorOccured = true;
                                                                                if (userPassword.length() == 0) {
                                                                                                                                                        clearAndHide();
                                                                                  if (errorOccured) {
                                                                                                                                                      } else {
                                                                                     errorMsg.append(", та");
                                                                                                                                                        focus Component.select All();\\
                                                                                                                                                        JOptionPane.showMessageDialog(
NewAdminCustomDialog.this,
         if (userSurmane.length() == 0) {
            if (errorOccured) {
                                                                                  errorOccured = true;
                                                                                  errorMsg.append(" пароль");
                                                                                                                                                             errorMsg.toString(),
              errorMsg.append(", та");
                                                                                  focusComponent = mJPasswordField;
                                                                                                                                                              "Спробуйте ще раз",
           errorOccured = true;
                                                                                                                                                             JOptionPane.ERROR_MESSAGE
           errorMsg.append(" прізвище
адміністратора");
                                                                               errorMsg.append("!");
                                                                                                                                                        userName = null;
           focusComponent = textField2;
                                                                                                                                                        userSurmane = null:
                                                                               if (!errorOccured) {
                                                                                                                                                        userBirthDateStr = null; \\
         if \ (userBirthDateStr.length() == 0) \ \{\\
                                                                                                                                                         adminPosition = null;
           if (errorOccured) {
                                                                                                                                                         adminSalary = null;
                                                                                     adminSalaryInt = (Integer)
              errorMsg.append(", та");
                                                                                                                                                        adminDateHiredStr = null; \\
                                                                      ((JFormattedTextField) textField5)
                                                                                                                                                        userPassword = null:
                                                                                                                                                        focusComponent.requestFocusInWindow();
           errorOccured = true;
           errorMsg.append(" дату народження у
                                                                      .getFormatter().stringToValue(adminSalary);
фрматі день/місяць/рік");
                                                                                                                                                    } else {
           focusComponent = textField3;
                                                                                    userBirthDate = (Date)
                                                                                                                                                      userName=null;\\
                                                                      ((JFormattedTextField)\ textField3)
                                                                                                                                                      userSurmane = null:
                                                                                                                                                      userBirthDateStr = null;
         if (adminPosition.length() == 0) {
                                                                      .getFormatter().stringToValue(userBirthDateStr);
                                                                                                                                                      adminPosition = null;
            if (errorOccured) {
                                                                                                                                                      adminSalary = null;
              errorMsg.append(", та");
                                                                                     adminDateHired = (Date)
                                                                                                                                                      adminDateHiredStr = null; \\
                                                                      ((JFormattedTextField) textField6)
                                                                                                                                                      userPassword = null:
           errorOccured = true;
                                                                                                                                                      clearAndHide();
           errorMsg.append(" посаду");
                                                                      .getFormatter().stringToValue(adminDateHiredStr);
           focusComponent = textField4;
                                                                                  } catch (ParseException e1) {
                                                                                    e1.printStackTrace();
         if (adminSalary.length() == 0) {
                                                                                                                                               * Метод очищує всі поля діалогу
           if (errorOccured) {
              errorMsg.append(", та");
                                                                                  Context c = new Context();
                                                                                  c.put("name", userName);
                                                                                                                                              public void clearAndHide() {
           errorOccured = true;
                                                                                  c.put("surname", userSurmane);
                                                                                                                                                 textField1.setText(null);
           errorMsg.append(" зарплату");
                                                                                  c.put("birthday", userBirthDate.getTime());\\
                                                                                                                                                 textField2.setText(null);
                                                                                  c.put("position", adminPosition);
           focusComponent = textField5;\\
                                                                                                                                                 textField3.setText(null);
                                                                                 c.put("salary", adminSalaryInt);
c.put("dateHired",
                                                                                                                                                 textField4.setText(null);
         if (adminDateHiredStr.length() == 0) {
                                                                                                                                                 textField5.setText(null);
            if (errorOccured) {
                                                                      adminDateHired.getTime());
                                                                                                                                                 textField6.setText(null);
              errorMsg.append(", та");
                                                                                 c.put("password", userPassword);
                                                                                                                                                 mJPasswordField.setText(null);
                                                                                                                                                 setVisible(false);
           errorOccured = true:
                                                                                                                                                 dispose();
           errorMsg.append(" дату прийняття у
фрматі день/місяць/рік");
                                                                      Command Executor.get Instance (). execute ("register Admi
           focusComponent = textField6;
                                                                      n", c, null);
                                                                                  } catch (Exception ex) {
```