НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА

|  |
| --- |
| з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»  на тему: «Організація робочого місця інвестора  у системі підтримки клієнтів будівельної компанії» |

|  |  |  |
| --- | --- | --- |
|  | Студента 2 курсу групи ІО-31  напряму підготовки  6.050102 «Комп’ютерна інженерія»  Штанька Андрія Олеговича  Керівник  Болдак Андрій Олександрович  (прізвище та ініціали)  Доцент кафедри ОТ  (посада, вчене звання, науковий ступінь)  Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  |
|  |  |

|  |
| --- |
| м. Київ - 2015 рік |

[РОЗДІЛ 1 3](#_Toc415530713)

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

2

*6.050102 «Комп’ютерна інженерія»*

Розроб.

Штанько

Перевір.

Болдак

Реценз.

Н. Контр.

Затверд.

Болдак

Система підтримки клієнтів будівельної компанії

Літ.

Аркушів

53

*Кафедра Обчислювальної техніки*

[ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ 3](#_Toc415530714)

[1.1. Введення 3](#_Toc415530715)

[1.2. Короткий огляд продукту 3](#_Toc415530716)

[1.3. Ділові правила і приписи 5](#_Toc415530717)

[1.4. Функціональність системи. 6](#_Toc415530718)

[1.5. Надійність. 6](#_Toc415530718)

РОЗДІЛ 2 7

[РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 7](#_Toc415530719)

[2.1. Загальна схема прецедентів 7](#_Toc415530720)

[2.2. Прецеденти для ролі інвестора 7](#_Toc415530721)

[2.3. Діаграма бізнес-сутностей 10](#_Toc415530722)

[2.4. Реляційна модель бази даних 11](#_Toc415530723)

[2.5. Специфікація таблиць бази даних 11](#_Toc415530724)

[РОЗДІЛ 3 15](#_Toc415530725)

[РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 15](#_Toc415530725)

[3.1. Документація класів програмного продукту 15](#_Toc415530726)

[3.2. Специфікація DAO-класів 21](#_Toc415530728)

[3.3. Класи контролерів та їх специфікація 22](#_Toc415530729)

[РОЗДІЛ 4 23](#_Toc415530730)

[ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ 23](#_Toc415530730)

[4.1. Функціонал робочого місця інвесора. 23](#_Toc415530731)

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 26](#_Toc415530732)

[ДОДАТОК А 27](#_Toc415530733)

[ДОДАТОК Б 28](#_Toc415530734)

[ДОДАТОК В 31](#_Toc415530735)



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

3

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

* 1. Введення

У цьому документі описуються запити зацікавлених осіб по відношенню до розроблювальної системи «Будівельна компанія» в якості яких виступають будь-яка фізична чи юридична особа, контакт-менеджер та клієнти цієї компанії.

Гість – майбутній клієнт;

Клієнт – фізична особа, що користується послугами будівельної компанії;

Контакт-менеджер – відповідає за зв’язок компанії з клієнтом;

* + 1. Мета

Метою документу є визначення основних вимог до функціональності, продуктивності, експлуатаційної придатності, а також визначення бізнес-правил і технологічних обмежень, пред’явлених предмету розробки.

* + 1. Контекст

Перелік вимог, зазначених у даному документі, є основою технічного завдання для розробки системи «Будівельна компанія».

* 1. Короткий огляд продукту

Система «Будівельна компанія» надає інтерфейс взаємодії між замовником, персоналом та клієнтами цієї компанії. Замовник має найвищий рівень доступу до організації системи та може контролювати усі внутрішні процеси. Персонал (контакт-менеджер) має доступ до бази даних клієнтів (редагування та оновлення даних клієнта, реєстрація нового клієнта), що дозволяє працівникам напряму спілкуватися з клієнтами. Клієнт має доступ лише до своєї анкети у базі даних та загальної інформації. Гість має доступ лише до загальної інформації.

* 1. **Ділові правила і приписи**
     1. Призначення системи

Система призначена для зберігання інформації про клієнтів компанії із подальшим використанням на різних рівнях доступу для полегшення організації комунікаційного процесу та економії часу при автоматичному заповненні необхідних документів.

* + 1. Політика взаємовідносин

Система «Будівельна компанія» надає можливість усім працівникам використовувати програмне забезпечення для фіксування результатів переговорів контакт-менеджер – інвестори.

Гість може переглядати список наявних пропозицій та читати форум.

Інвестор має постійний доступ до процесу будівництва за допомогою засобів відеоспостереження, має можливість призначати зустріч з контакт-менеджером, переглядати історію переговорів, а також отримувати повну інформацію про інвестоване майно. За політикою конфіденційності системи інвестор має доступ лише до свого профілю, маючи пару логін-пароль, яку він отримує після успішного підписання договору. Інвестор є повноцінним учасником форуму, де може залишати свої повідомлення, оцінювати якість роботи компанії.

Контакт-менеджер має власний робочий кабінет в системі у якій оброблює заявки та повідомлення від інвесторів.

* + 1. Сценарій реєстрації інвестора

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

4

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Назва**: реєстрація інвестора.

**Учасники**: інвестор, система.

**Попередні умови**:немає.

**Результат**: новий зареєстрований інвестор.

**Основний сценарій:**

1. Інвестор відправляє запит на реєстрацію, система готує реєстраційну форму.

2. Інвестор заповнює форму, система перевіряє правильність заповнення.

3. Система реєструє користувача.

* + 1. Сценарій призначення зустрічі з контакт-менеджером.

**Назва**: призначення зустрічі з контакт-менеджером.

**Учасники**: інвестор, контакт-менеджер, система.

**Попередні умови**: інвестор зареестрований у системі.

**Результат**: призначення зустрічі.

**Основний сценарій:**

1. Інвестор авторизується в системі.

2. Інвестор заходить у розділ системи «Робота з клієнтами».

3. Інвестор натискає на посилання «Залишити повідомлення для адміністрації».

4. Інвестор заповнює поле з назвою « Залишити повідомлення», де вказує причину зустрічі та та натискає кнопку «Надіслати повідомлення».

5. Контакт-менеджер отримує повідомлення.

6. Контакт-менеджер зв’язується з інвестором та призначає зустріч.

* + 1. Сценарій залишення повідомлення на форумі.

**Назва**: залишення повідомлення на форумі.

**Учасники**: інвестор, система.

**Попередні умови**: залишення повідомлення.

**Результат**: новий коментар на сторінці орендаря.

**Основний сценарій:**

1. Інвестор авторизується в системі.

2. Інвестор заходить у форум системи.

3. Інвестор знаходить потрібний розділ форуму.

4. Інвестор заповнює поле з назвою « Залишити коментар» та натискає кнопку «Надіслати повідомлення».

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

6

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* 1. **Функціональність системи**

Основні вимоги до функціональності, пред’явлені зацікавленими особами, відносяться до трьох категорій:

* Гість;
* Інвестор;
* Контакт-менеджер.
  + 1. Структура облікового запису інвестора

1. Паспортні дані – прізвище, ім’я та по-батькові інвестора, дата народження, адреса проживання, сімейний стан, прізвище, ім’я та по-батькові жінки (чоловіка) тощо.
2. Контактні дані – skype, e-mail, vk, facebook, мобільні номери чоловіка та/або жінки.
3. Дані про угоду – дата підписання, дата прийняття в експлуатацію, умови оплати, фотокопії попереднього договору та інших документів;
4. Зв’язок з контакт-менеджером – призначення зустрічі, надсилання повідомлень;
5. Доступ до залишених повідомлень на форумі, можливість редагування або видалення своїх повідомлень;
6. Переглядання онлайн відео з будівельного майданчику (різні ракурси);
7. Фінансовий баланс – чіткий звіт про вкладені інвестиції та відображення поточного залишку до оплати.

Система забезпечує інвестору використання зазначених вище інформації та можливостей.

* + 1. Можливості гостей

1. Перегляд форуму;
2. Перегляд наявних пропозицій;
3. Можливість призначати зустріч із контакт-менеджером.
   * 1. Можливості контакт-менеджера
4. Оброблювати запити – назначати час та місце зустрічі;
5. Оброблювати повідомлення;
6. Підписання договору купівлі-продажу;
7. Перевірка валідності введених даних інвестором та їх конфірмація

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

6

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* 1. Надійність.

Протягом усього терміну зберігання для облікових записів повинна бути забезпечена їх цілісність, незмінність і достовірність. Для забезпечення збереження та цілісності використовуватиметься метод резервного копіювання. Система повинна бути добре захищена від різного роду зловмисних атак із метою заволодіння інформації чи атак типу DDoS. Має використовуватися комплекс технологічних і адміністративних процедур, що перешкоджають випадковій або навмисній зміні збережених записів. Найкращим рішенням є використання електронного цифрового підпису, що дозволяє у будь-який момент перевірити незмінність збереженого запису в порівнянні з моментом його підписання.Також система повинна витримувати великі навантаження, обслуговуючи значну кількість користувачів.Крім цього повинна забезпечуватись конфіденційність персональної інформації. Надання доступу до збережених підписаних записів та персональних даних інвесторів здійснюють у відповідності з правами доступу.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

7

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

* 1. Загальна схема прецедентів

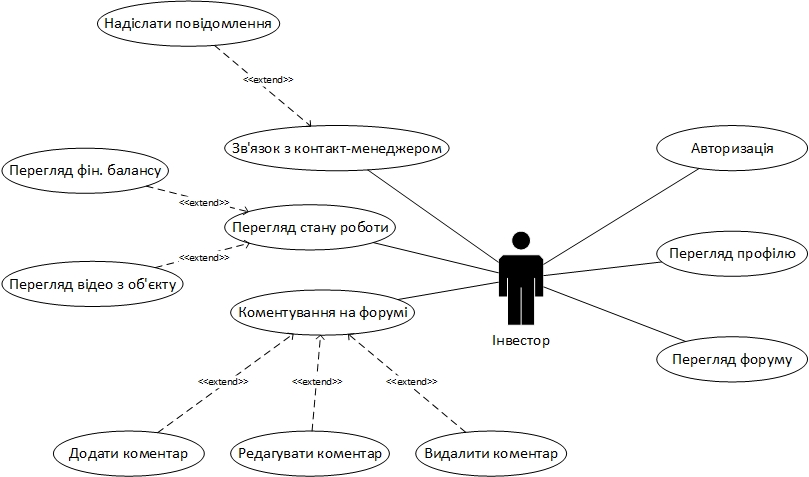
****Загальна схема прецедентів для ролі інвестора показує можливі послідовності дій актора. Основним видом діяльності інвесора є вхід у систему, видалення та додавання інвестицій. Також інвестор може зареєструватися у системі. Схема прецедентів представлена на рис. 2.1.

Рис. 2.1 – Загальна схема прецедентів для ролі інвестора

* 1. Прецеденти для ролі інвестора

Нижче описані процедури для ролі користувача з вказаними передумовами, результатом, виключними ситуаціями та детальним описом послідовності дій.

**ID: UC 001**

**Назва:** Перша авторизація інвестора.

**Учасники:** Інвестор, система.

**Передумови:** Інвестор уклав договір із будівельною компанією.

**Результат:** Система надсилає запит на підтвердження даних про інвестора контакт-менеджеру.

**Основний сценарій:**

1. Інвестор відправляє запит на авторизацію.
2. Система надає форму із параметрами «Логін» та «Пароль» для авторизації.
3. Інвестор вводить свої дані (логін/пароль).
4. Інвестор натискає кнопку «Ввійти в систему».
5. Система перевіряє дані авторизації.
6. Система відправляє інвестора на сторінку «Паспортні дані».
7. Інвестор заповнює параметри сторінки «Паспортні дані» («Прізвище», «Ім’я» «По-батькові», заповнює параметри сторінки «Контактні дані»«Дата народження», «Адреса проживання», «Сімейний стан», «Прізвище, ім’я та по-батькові жінки (чоловіка)») та натискає «Далі».
8. Система відправляє інвестора на сторінку «Контактні дані».
9. Інвестор («Мобільний номер» , «Skype», «e-mail», «Vk», «Facebook», «Мобільний номер жінки(чоловіка)»)та натискає «Далі».
10. Система відправляє інвестора на сторінку «Дані про угоду».
11. Інвестор заповнює параметри сторінки «Дані про угоду» («Дата підписання», «Дата прийняття в експлуатацію», «Умови оплати», «Фотокопії попереднього договору та інших документів») та натискає «Завершити ввід даних інвестора».
12. Система відправляє запит на перевірку валідності введених даних інвестора контакт-менеджеру.

**Виключні ситуації:**

1. Введені персональні дані не вірні (логін/пароль).
2. Інвестор не заповнив якесь з полів.

**ID: UC 002**

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

8

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Назва:** Авторизація користувача.

**Учасники:** Користувач, система.

**Передумови:** Користувач не має доступа до системи.

**Результат:** Користувач отримує доступ до системи.

**Основний сценарій:**

1. Користувач відправляє запит на авторизацію.
2. Система надає форму із параметрами «Логін» та «Пароль» для авторизації.
3. Користувач вводить свої дані (логін/пароль).
4. Користувач натискає кнопку «Ввійти в систему».
5. Система перевіряє дані авторизації.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

9

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

1. Система дає доступ користувачу до системи.

**Виключні ситуації:**

Введені персональні дані не вірні (логін/пароль).

**ID: UC 003**

**Назва:** Перегляд профілю.

**Учасники:** Користувач, система.

**Передумови:** Користувач авторизований у системі.

**Результат:** Користувач переглядає свій профіль.

**Основний сценарій:**

1. Користувач натискає на кнопку “Переглянути профіль”.
2. Система відправляє користувача на сторінку «Мій профіль».
3. Користувач обирає категорію персональних даних користувача та натискає кнопку «Переглянути».
4. Система відправляє користувача на потрібну сторінку.
5. Користувач переглядає обрану інформацію.

**ID: UC 004**

**Назва:** Додавання інвестиції.

**Учасники:** Інвестор, система.

**Передумови:** Інвестор авторизований у системі, інвестор бажає додати інвестицію.

**Результат:** Система додає інвестицію інвестора.

**Основний сценарій:**

1. Інвестор натискає кнопку “Інвестиції”.
2. Система відправляє інвестора до інвестицій.
3. Інвестор натискає на кнопку « Додати»
4. Система надає інвесторові форму з параметрами «Об’єкт», «Опис».
5. Інвестор заповнює поля та натискає кнопку «Додати».
6. Система додає об’єкт інвестора та надає форму номеру контракту.
7. Інвестор вписує свій номер контракту.
8. Система додає нову інвестицію

**Виключні ситуації:**

1. Інвестор не заповнив поля.

**ID: UC 005**

**Назва:** Видалення коментаря.

**Учасники:** Інвестор, система.

**Передумови:** Інвестор авторизований у системі, інвестор бажає видалити інвестицію.

**Результат:** Система видаляє інвестицію.

**Основний сценарій:**

1. Інвестор натискає кнопку “Інвестиції”.
2. Система відправляє інвестора до інвестицій.
3. Інвестор обирає необхідну інвестицію та натискає кнопку «Видалити».
4. Система видаляє об’єкт інвестиції а потім саму інвестицію.

**Виключні ситуації:**

Інвестор не має інвестицій.

* 1. Діаграма бізнес-сутностей

Дана діаграма створюється на етапі бізнес моделювання. Вона відображає основні сутності та взаємозв’язки між ними. В даному випадку основними сутностями є «Інвестор», «Інвестиція», «Об’єкт», «Коментар», «Повідомлення», «Операція», які взаємодіють між собою та включають у себе допоміжні бізнес-сутності. Діаграма бізнес-сутностей проекту зображена на рис. 2.5.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

10

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

12

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

10

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

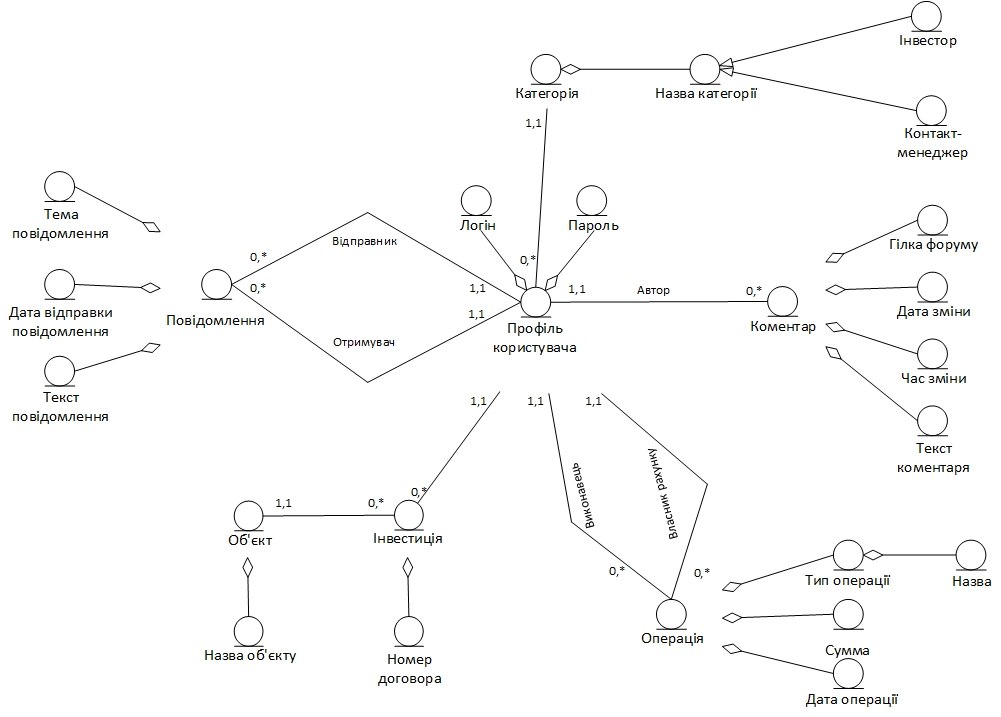
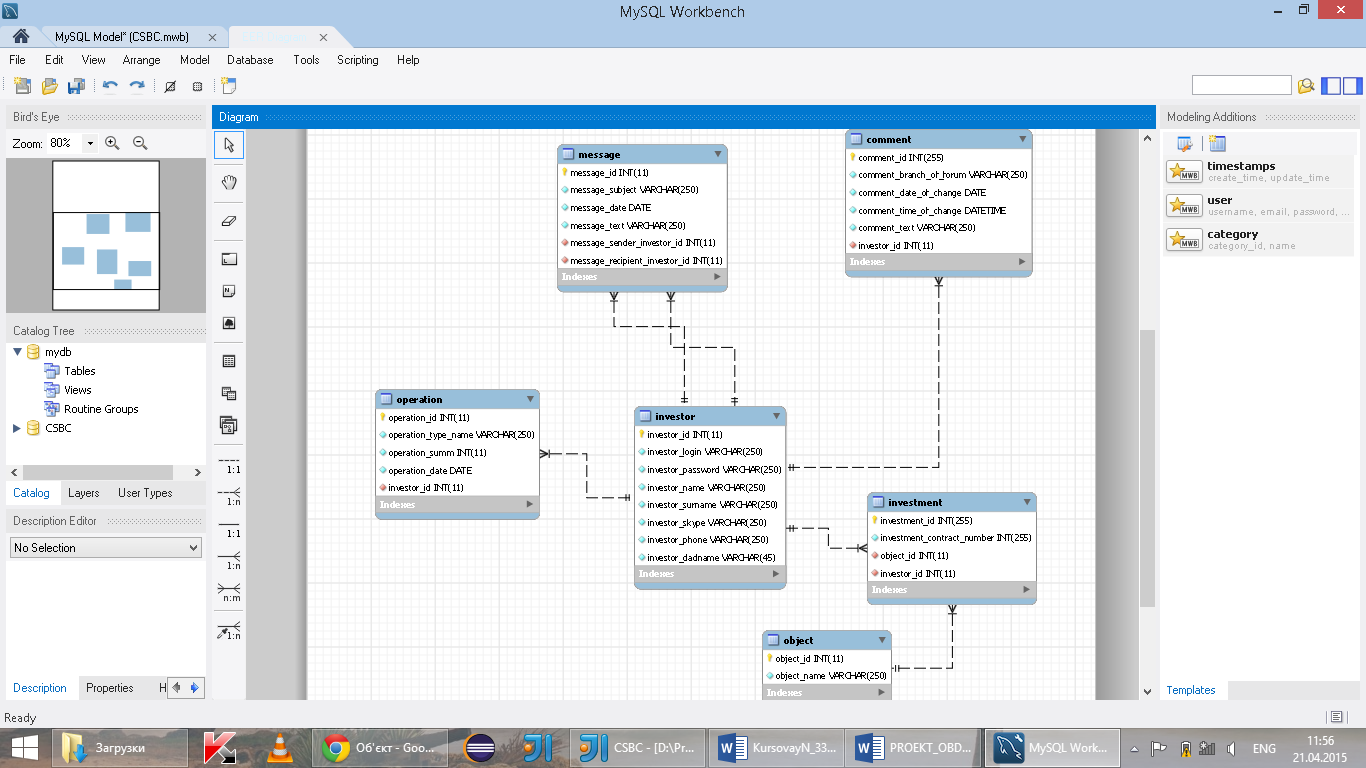


Рис. 2.5 – Діаграма бізнес-сутностей

* 1. Реляційна модель бази даних

Реляційна модель бази даних (рис 2.3) зображує структуру таблиць бази даних, взаємозв’язки між ними та поля кожної з таблиць. Наведена діаграма має багато схожого з діаграмою бізнес-сутностей. Кожній основній бізнес-сутності відповідає таблиця баз даних.



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

11

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Рис 2.6 – Реляційна модель

* 1. Специфікація таблиць бази даних

Специфікація таблиць бази даних включає в себе інформацію про назви колонок таблиці, їхній тип, інформацію про те, чи є ця колонка первинним ключем, чи поле може бути пустим, чи значення поля автоматично збільшується та коментар щодо призначення колонки. Таблиці зі специфікаціями наведені нижче.

Таблиця 2.1

Таблиця Investor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Так | Ідентифікатор інвестора |
| Name | VARCHAR | Ні | Так | Ні | Ім’я інвестора |
| Surname | VARCHAR | Ні | Так | Ні | Прізвище інвестора |
| Login | VARCHAR | Ні | Так | Ні | Логін інвестора |
| Password | VARCHAR | Ні | Так | Ні | Пароль інвестора |
| Skype | VARCHAR | Ні | Так | Ні | Skype інвестора |
| Phone | VARCHAR | Ні | Так | Ні | Контактний телефон інвестора |
| Dadname | VARCHAR | Ні | Так | Ні | По батькові інвестора |

Таблиця 2.2

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

12

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Таблиця Comment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Так | Ідентифікатор коменаря |
| Branch of forum | VARCHAR | Ні | Так | Ні | Гілка форуму |
| Date of change | DATE | Ні | Так | Ні | Дата відправки коментаря |
| Time of change | DATETIME | Ні | Так | Ні | Час відправки коментаря |
| Text | VARCHAR | Ні | Так | Ні | Текст коментаря |
| Investor id | INT | Ні | Так | Ні | Ідентифікатор інвестора |

Таблиця 2.3

Таблиця Message

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Так | Ідентифікатор повідомлення |
| Subject | VARCHAR | Ні | Так | Ні | Тема повідомлення |
| Date | DATE | Ні | Так | Ні | Дата відправки повідомлення |
| Text | VARCHAR | Ні | Так | Ні | Текст повідомлення |
| Sender id | INT | Ні | Так | Ні | Ідентифікатор відправника |
| Recipient id | INT | Ні | Так | Ні | Ідентифікатор отримувача |

Таблиця 2.4

Таблиця Investment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Так | Ідентифікатор інвестиції |
| Contract number | VARCHAR | Ні | Так | Ні | Номер контракту |
| Object id | INT | Ні | Так | Ні | Ідентифікатор об’єкту |
| Investor id | INT | Ні | Так | Ні | Ідентифікатор інвестора |

Таблиця 2.5

Таблиця Object

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

13

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Ні | Ідентифікатор об’єкту |
| Name | VARCHAR | Ні | Так | Ні | Назва об’єкту |
| Text | VARCHAR | Ні | Так | Ні | Опис об’єкту |

Таблиця 2.6

Таблиця Operation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Тип даних | Ключ | Не пуста | Авто-інкремент | Опис |
| Id | INT | Так | Так | Так | Ідентифікатор операції |
| Type | VARCHAR | Ні | Так | Ні | Тип операції |
| Summ | INT | Ні | Так | Ні | Загальна вартість операції |
| Date | DATE | Ні | Так | Ні | Дата операції |
| Investor id | INT | Ні | Так | Ні | Ідентифікатор інвестора |

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

14

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

15

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

* 1. Документація класів програмного продукту

Для реляційно-об’єктного відображення в програмі використовується заздалегідь написаний data access object (DAO). Він надає можливість легко встановити зв’язок з будь-якою базою даних та створити відображення між об’єктно-орієнтованою моделлю та традиційною реляційною моделлю баз даних. Детальна специфікація (JavaDoc) наведена нижче.

* + 1. Клас «Investor»

public class **Investor**

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#id) |
| private java.lang.String | [**login**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#login) |
| private java.lang.String | [**name**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#name) |
| private java.lang.String | [**password**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#password) |
| private java.lang.String | [**surname**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#surname) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#Investor--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#Investor-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String name, java.lang.String surname, java.lang.String login, java.lang.String password) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#getId--)() |
| java.lang.String | [**getLogin**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#getLogin--)() |
| java.lang.String | [**getName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#getName--)() |
| java.lang.String | [**getPassword**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#getPassword--)() |
| java.lang.String | [**getSurname**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#getSurname--)() |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#setId-int-)(int id) |
| void | [**setLogin**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#setLogin-java.lang.String-)(java.lang.String login) |
| void | [**setName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#setName-java.lang.String-)(java.lang.String name) |
| void | [**setPassword**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#setPassword-java.lang.String-)(java.lang.String password) |
| void | [**setSurname**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#setSurname-java.lang.String-)(java.lang.String surname) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html#toString--)() |

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

18

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Клас Message»

public class **Message**

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private java.sql.Date | [**date**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#date) |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#id) |
| private [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**recipient**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#recipient) |
| private [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**sender**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#sender) |
| private java.lang.String | [**subject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#subject) |
| private java.lang.String | [**text**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#text) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Message**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#Message--)() |
| [**Message**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#Message-java.lang.String-java.sql.Date-java.lang.String-com.dao.tableClasses.Investor-com.dao.tableClasses.Investor-)(java.lang.String subject, java.sql.Date date, java.lang.String text,[**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) sender, [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) recipient) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| java.sql.Date | [**getDate**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getDate--)() |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getId--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**getRecipient**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getRecipient--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**getSender**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getSender--)() |
| java.lang.String | [**getSubject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getSubject--)() |
| java.lang.String | [**getText**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#getText--)() |
| void | [**setDate**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setDate-java.sql.Date-)(java.sql.Date date) |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setId-int-)(int id) |
| void | [**setRecipient**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setRecipient-com.dao.tableClasses.Investor-)(**[Investor](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\Investor.html" \o "class in com.dao.tableClasses)** recipient) |
| void | [**setSender**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setSender-com.dao.tableClasses.Investor-)(**[Investor](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\Investor.html" \o "class in com.dao.tableClasses)** sender) |
| void | [**setSubject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setSubject-java.lang.String-)(java.lang.String subject) |
| void | [**setText**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#setText-java.lang.String-)(java.lang.String text) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Message.html#toString--)() |

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

16

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Клас «Comment»

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

17

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

public class **Comment**

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private java.lang.String | [**branchofforum**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#branchofforum) |
| private java.lang.String | [**comment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#comment) |
| private java.sql.Date | [**dateofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#dateofchange) |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#id) |
| private [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#investor) |
| private java.sql.Time | [**timeofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#timeofchange) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Comment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#Comment--)() |
| [**Comment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#Comment-java.lang.String-java.sql.Date-java.sql.Time-java.lang.String-com.dao.tableClasses.Investor-)(java.lang.String branchofforum, java.sql.Date dateofchange, java.sql.Time timeofchange, java.lang.String comment, [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) investor) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| java.lang.String | [**getBranchofforum**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getBranchofforum--)() |
| java.lang.String | [**getComment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getComment--)() |
| java.sql.Date | [**getDateofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getDateofchange--)() |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getId--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**getInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getInvestor--)() |
| java.sql.Time | [**getTimeofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#getTimeofchange--)() |
| void | [**setBranchofforum**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setBranchofforum-java.lang.String-)(java.lang.String branchofforum) |
| void | [**setComment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setComment-java.lang.String-)(java.lang.String comment) |
| void | [**setDateofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setDateofchange-java.sql.Date-)(java.sql.Date dateofchange) |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setId-int-)(int id) |
| void | [**setInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setInvestor-com.dao.tableClasses.Investor-)(**[Investor](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\Investor.html" \o "class in com.dao.tableClasses)** investor) |
| void | [**setTimeofchange**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#setTimeofchange-java.sql.Time-)(java.sql.Time timeofchange) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Comment.html#toString--)() |

* + 1. Клас «Operation»

public class **Operation**

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

18

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private java.sql.Date | [**date**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#date) |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#id) |
| private [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#investor) |
| private java.lang.String | [**name**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#name) |
| private int | [**summ**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#summ) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Operation**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#Operation--)() |
| [**Operation**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#Operation-java.lang.String-int-java.sql.Date-com.dao.tableClasses.Investor-)(java.lang.String name, int summ, java.sql.Date date,[**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) investor) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| java.sql.Date | [**getDate**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#getDate--)() |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#getId--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**getInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#getInvestor--)() |
| java.lang.String | [**getName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#getName--)() |
| int | [**getSumm**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#getSumm--)() |
| void | [**setDate**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#setDate-java.sql.Date-)(java.sql.Date date) |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#setId-int-)(int id) |
| void | [**setInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#setInvestor-com.dao.tableClasses.Investor-)(**[Investor](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\Investor.html" \o "class in com.dao.tableClasses)** investor) |
| void | [**setName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#setName-java.lang.String-)(java.lang.String name) |
| void | [**setSumm**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#setSumm-int-)(int summ) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Operation.html#toString--)() |

* + 1. Клас «Investment»

public class **Investment**

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private int | [**contractnumber**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#contractnumber) |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#id) |
| private [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#investor) |
| private [**OObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html) | [**object**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#object) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Investment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#Investment--)() |
| [**Investment**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#Investment-int-com.dao.tableClasses.OObject-com.dao.tableClasses.Investor-)(int contractnumber, [**OObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html) object, [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) investor) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| int | [**getContractnumber**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#getContractnumber--)() |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#getId--)() |
| [**Investor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investor.html) | [**getInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#getInvestor--)() |
| [**OObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html) | [**getOObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#getOObject--)() |
| void | [**setContractnumber**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#setContractnumber-int-)(int contractnumber) |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#setId-int-)(int id) |
| void | [**setInvestor**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#setInvestor-com.dao.tableClasses.Investor-)(**[Investor](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\Investor.html" \o "class in com.dao.tableClasses)** investor) |
| void | [**setOObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#setOObject-com.dao.tableClasses.OObject-)(**[OObject](file:///D:\\Program%20Files\\documentation\\com\\dao\\tableClasses\\OObject.html" \o "class in com.dao.tableClasses)** object) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\Investment.html#toString--)() |

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

19

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Клас «OObject»

public class **OObject**

extends java.lang.Object

### *Field Summary*

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private int | [**id**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#id) |
| private java.lang.String | [**name**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#name) |
| private java.lang.String | [**text**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#text) |

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**OObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#OObject--)() |
| [**OObject**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#OObject-java.lang.String-)(java.lang.String name) |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| int | [**getId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#getId--)() |
| java.lang.String | [**getName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#getName--)() |
| java.lang.String | [**getText**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#getText--)() |
| void | [**setId**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#setId-int-)(int id) |
| void | [**setName**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#setName-java.lang.String-)(java.lang.String name) |
| void | [**setText**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#setText-java.lang.String-)(java.lang.String text) |
| java.lang.String | [**toString**](file:///D:\Program%20Files\documentation\com\dao\tableClasses\OObject.html#toString--)() |

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

23

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

20

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

21

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* 1. Специфікація DAO-класів

Класи, що тут представлені, містять методи для роботи з базою даних. Діаграму цих класів можна побачити на рис. 3.3. Детальна специфікація наведена в додатку В.

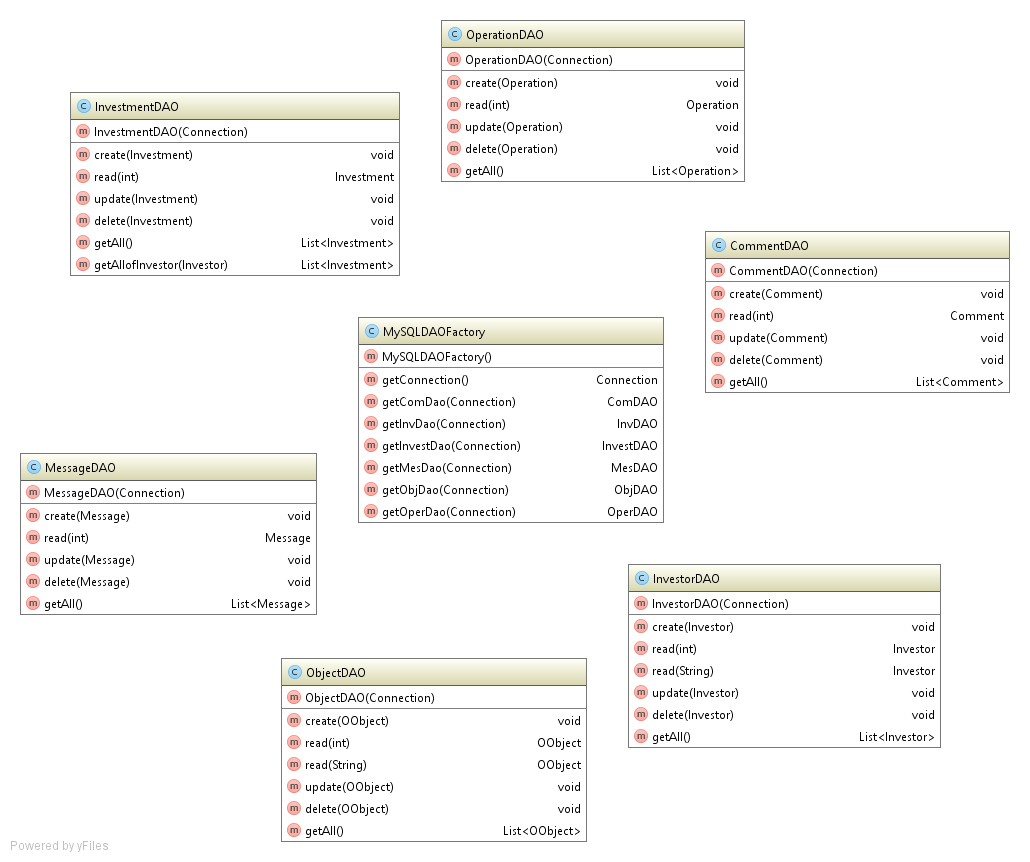


Рис 3.2 – Діаграма DAO класів

* 1. Класи контролерів та їх специфікація

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

22

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Дані класи призначені для створення зв’язку між сервером та клієнтом. Діаграму цих класів можна побачити на рис. 3.4. Детальна специфікація наведена в додатку В.

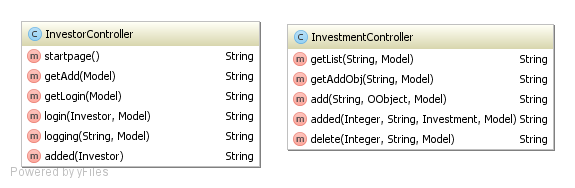
****

Рис. 3.3 – Діаграма класів контролерів

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

23

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

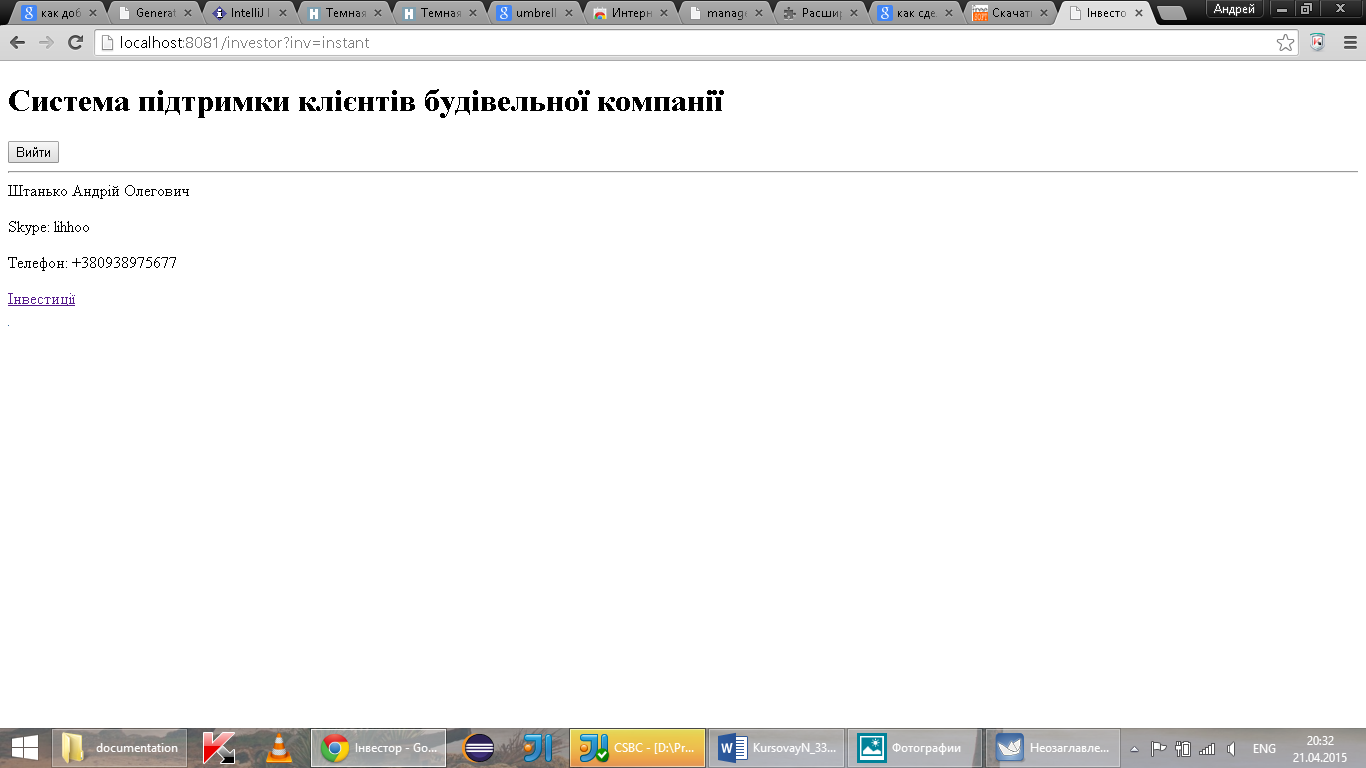
Підпис

Дата

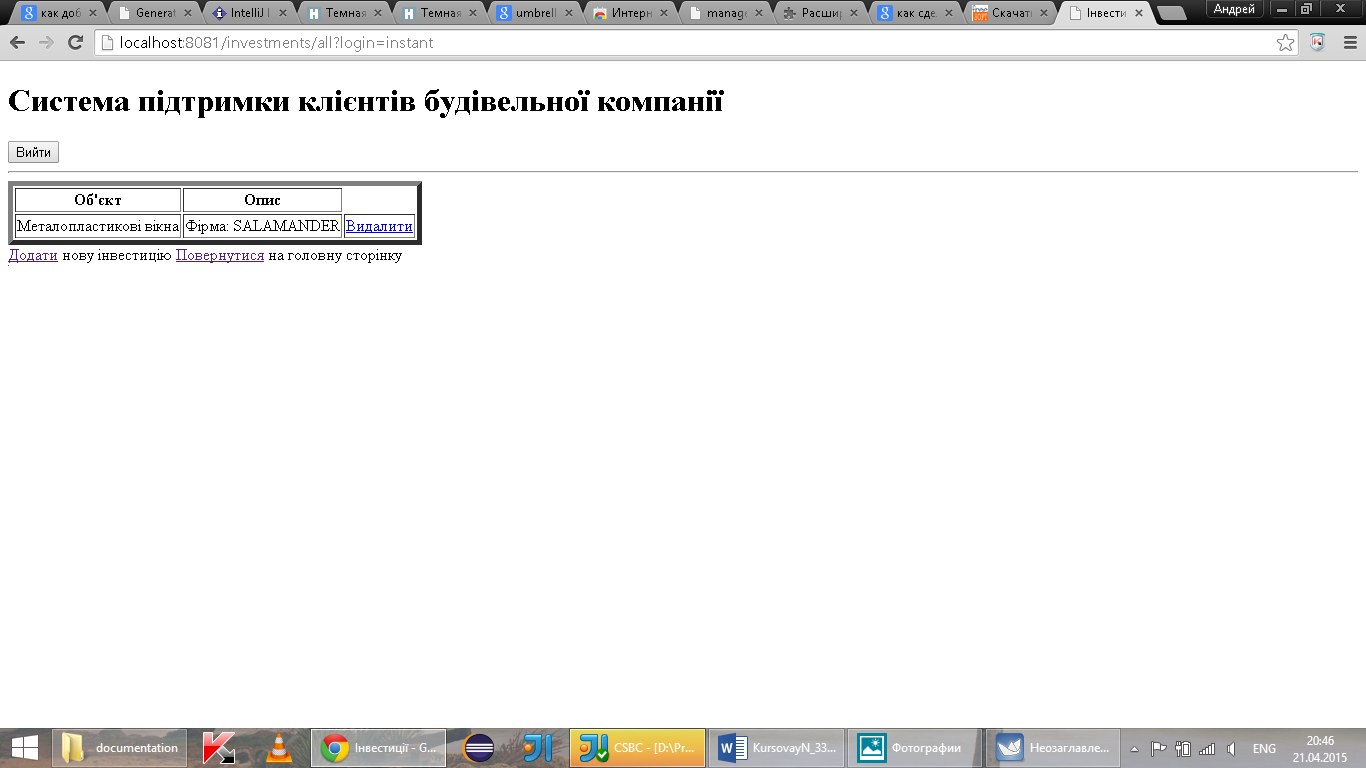
ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

Для ілюстрації роботи програми в цьому розділі наведено графічні сценарії роботи проекту.

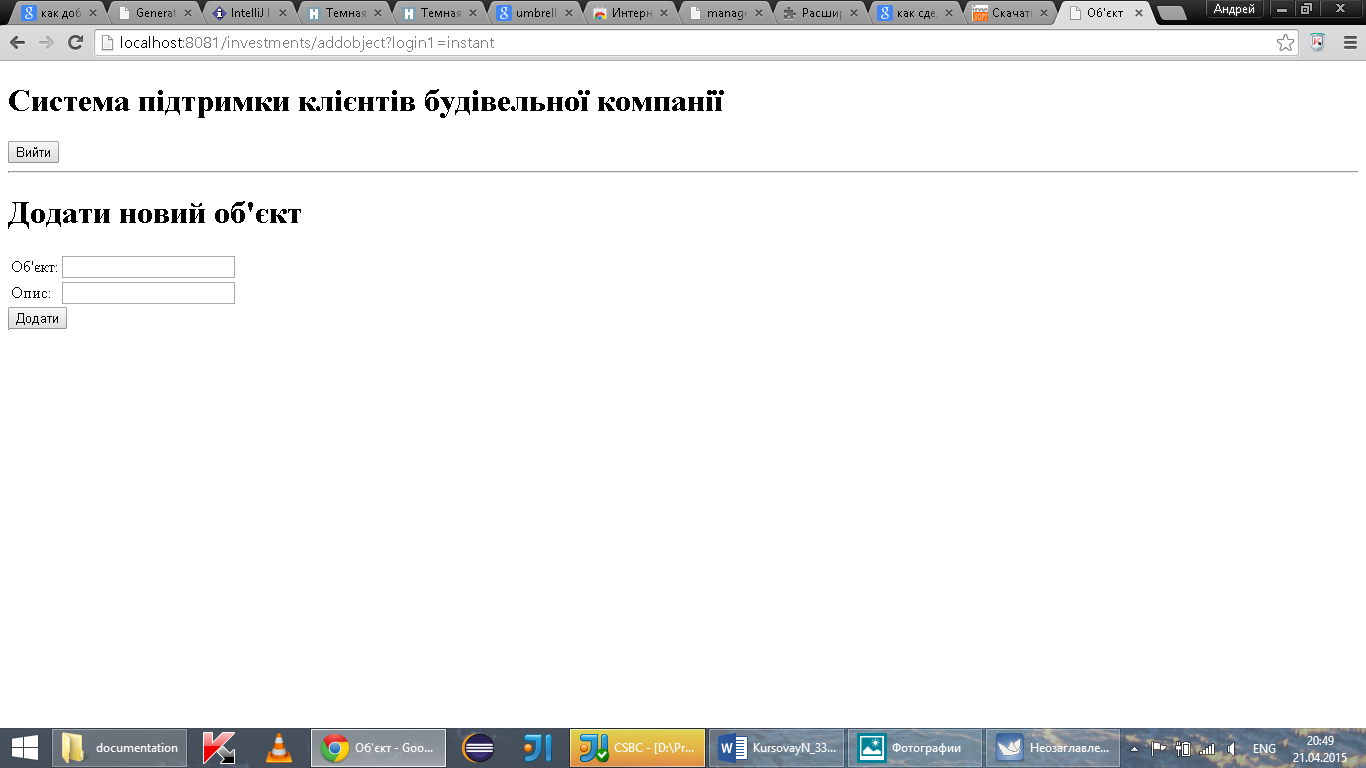
* 1. Додавання нової інвестиції.
     1. На своїй сторінці інвестор натискає посилання «Інвестиції»



* + 1. Система переходить на сторінку зі списком усіх інвестицій інвестора



* + 1. Інвестор натискає на посилання «Додати», і система надає форму для створення нового об’єкту



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

24

6.050102 «Комп'ютерна інженерія»

Змн.

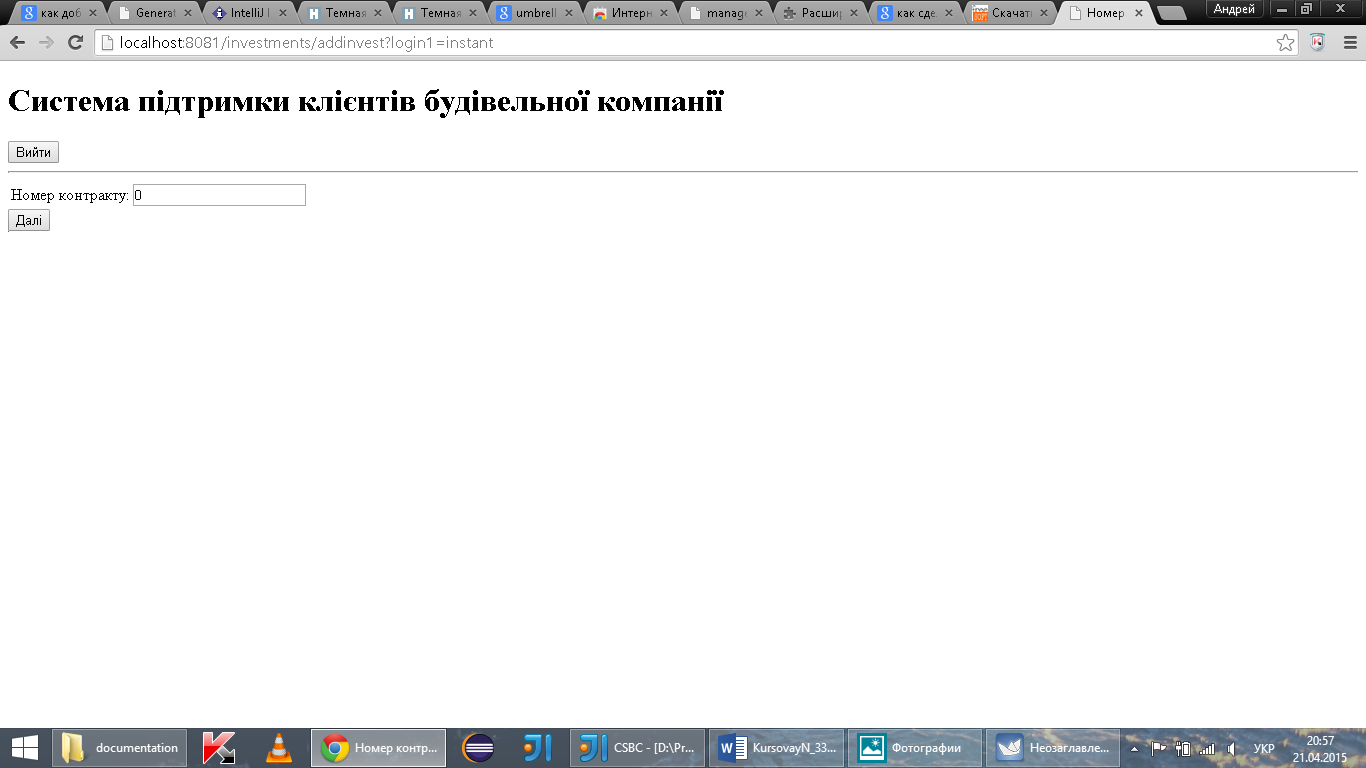
№ докум.

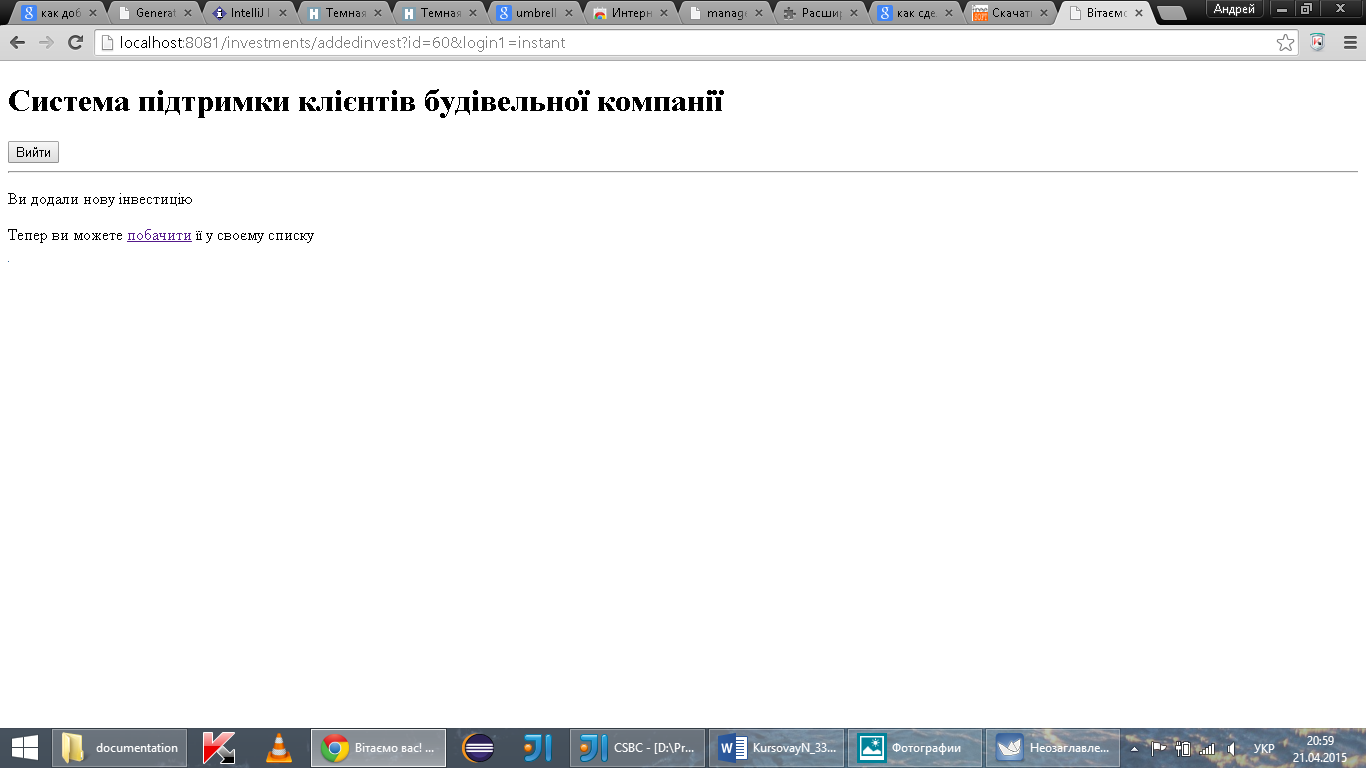
Арк.

Підпис

Дата

* + 1. Інвестор заповнює ніформацю про об’єкт, та натискає кнопку «Додати», система надає форму для введення номеру контракту інвестора



* + 1. Інвестор вводить номер свого контракту та натискає кнопкі «Далі», система створює нову інвестицію та виводить повідомлення про успішне додавання нової інвестиції

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

25

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Інвестор натискає на посилання «побачити», переходить до списку усіх своїх інвестицій і бачтиь нову інвестицію

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

26

1

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

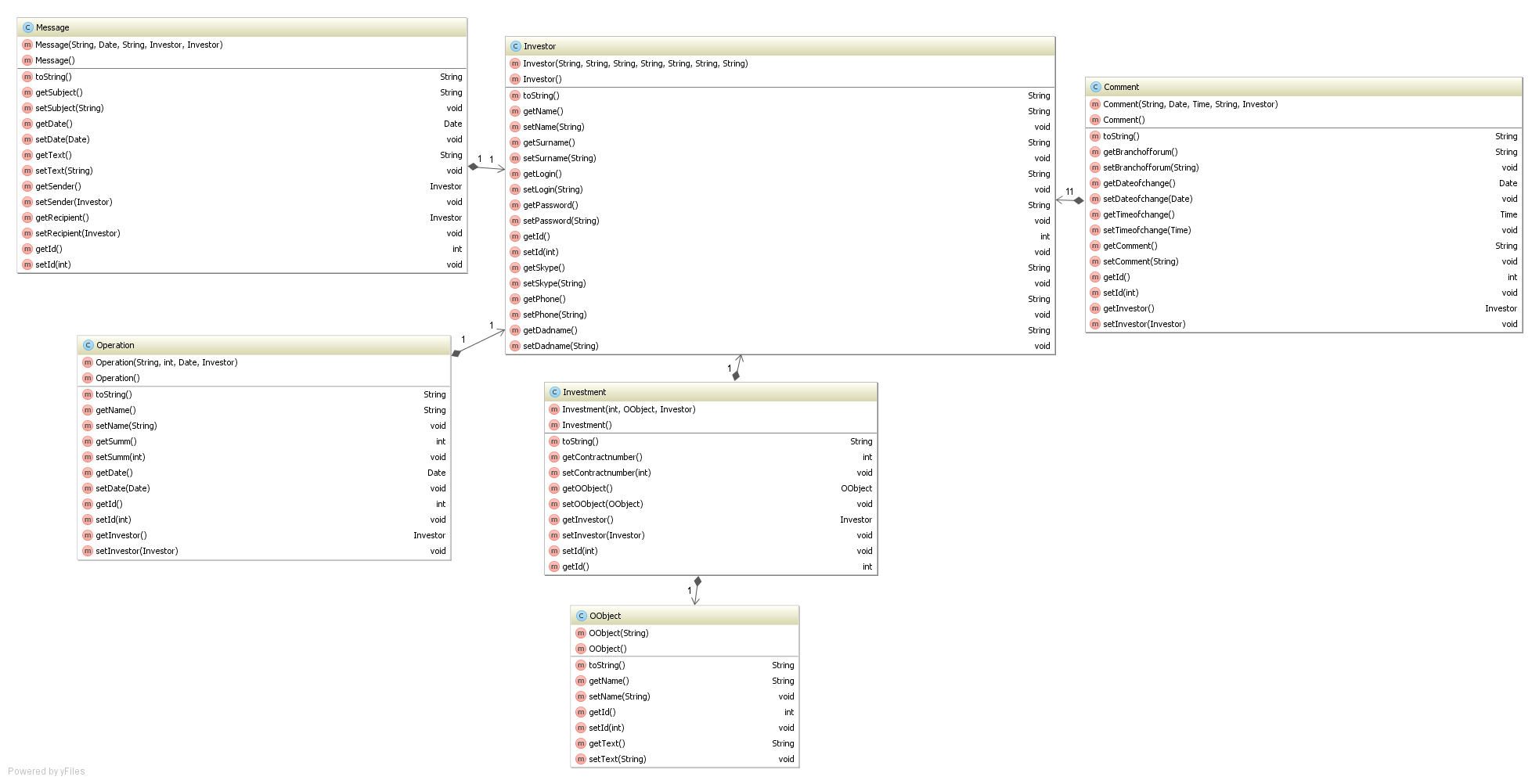
Підпис

Дата

1. Spring Framevork. – Посилання: https://spring.io/
2. Apache Tomcat. – Посилання: <http://tomcat.apache.org>
3. Maven. – Посилання: <https://maven.apache.org>

ДОДАТОК А

Діаграма класів



ДОДАТОК Б

SQL код для створення таблиць бази даних:

ET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

CREATE SCHEMA IF NOT EXISTS `CSBC` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

CREATE TABLE IF NOT EXISTS `CSBC`.`comment` (

`comment\_id` INT(255) NOT NULL AUTO\_INCREMENT,

`comment\_branch\_of\_forum` VARCHAR(250) NOT NULL,

`comment\_date\_of\_change` DATE NOT NULL,

`comment\_time\_of\_change` DATETIME NOT NULL,

`comment\_text` VARCHAR(250) NOT NULL,

`investor\_id` INT(11) NOT NULL,

PRIMARY KEY (`comment\_id`),

INDEX `profile\_id` USING BTREE (`investor\_id` ASC),

CONSTRAINT `comment\_pp`

FOREIGN KEY (`investor\_id`)

REFERENCES `CSBC`.`investor` (`investor\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

CREATE TABLE IF NOT EXISTS `CSBC`.`investment` (

`investment\_id` INT(255) NOT NULL AUTO\_INCREMENT,

`investment\_contract\_number` INT(255) NOT NULL,

`object\_id` INT(11) NOT NULL,

`investor\_id` INT(11) NOT NULL,

PRIMARY KEY (`investment\_id`),

INDEX `object\_id` USING BTREE (`object\_id` ASC),

INDEX `investment\_pp` (`investor\_id` ASC),

CONSTRAINT `investment\_pp`

FOREIGN KEY (`investor\_id`)

REFERENCES `CSBC`.`investor` (`investor\_id`),

CONSTRAINT `investment\_oo`

FOREIGN KEY (`object\_id`)

REFERENCES `CSBC`.`object` (`object\_id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

CREATE TABLE IF NOT EXISTS `CSBC`.`message` (

`message\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`message\_subject` VARCHAR(250) NOT NULL,

`message\_date` DATE NOT NULL,

`message\_text` VARCHAR(250) NOT NULL,

`message\_sender\_investor\_id` INT(11) NOT NULL,

`message\_recipient\_investor\_id` INT(11) NOT NULL,

PRIMARY KEY (`message\_id`),

INDEX `profile\_id` (`message\_sender\_investor\_id` ASC, `message\_recipient\_investor\_id` ASC),

INDEX `message\_mrp` (`message\_recipient\_investor\_id` ASC),

CONSTRAINT `message\_mrp`

FOREIGN KEY (`message\_recipient\_investor\_id`)

REFERENCES `CSBC`.`investor` (`investor\_id`),

CONSTRAINT `message\_msp`

FOREIGN KEY (`message\_sender\_investor\_id`)

REFERENCES `CSBC`.`investor` (`investor\_id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

CREATE TABLE IF NOT EXISTS `CSBC`.`object` (

`object\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`object\_name` VARCHAR(250) NOT NULL,

PRIMARY KEY (`object\_id`),

INDEX `investment\_id` (`object\_id` ASC))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

CREATE TABLE IF NOT EXISTS `CSBC`.`operation` (

`operation\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`operation\_type\_name` VARCHAR(250) NOT NULL,

`operation\_summ` INT(11) NOT NULL,

`operation\_date` DATE NOT NULL,

`investor\_id` INT(11) NOT NULL,

PRIMARY KEY (`operation\_id`),

INDEX `operation\_pp` (`investor\_id` ASC),

CONSTRAINT `operation\_pp`

FOREIGN KEY (`investor\_id`)

REFERENCES `CSBC`.`investor` (`investor\_id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

CREATE TABLE IF NOT EXISTS `CSBC`.`investor` (

`investor\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`investor\_login` VARCHAR(250) NOT NULL,

`investor\_password` VARCHAR(250) NOT NULL,

`investor\_name` VARCHAR(250) NOT NULL,

`investor\_surname` VARCHAR(250) NOT NULL,

`investor\_skype` VARCHAR(255) NOT NULL,

`investor\_phone` VARCHAR(255) NOT NULL,

`investor\_dadname` VARCHAR(255) NOT NULL,

PRIMARY KEY (`investor\_id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_general\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

ДОДАТОК В

InvestorController.java

@Controller  
@RequestMapping(**"/"**)  
**public class** InvestorController {  
  
 @RequestMapping(value = **"/startpage"**, method = RequestMethod.***GET***)  
 **public** String startpage(){  
 **return "/startpage"**;  
 }  
  
 @RequestMapping(value = **"/registration"**, method = RequestMethod.***GET***)  
 **public** String getAdd(Model model) **throws** SQLException {  
  
  
 model.addAttribute(**"investorAttribute"**, **new** Investor());  
  
  
 **return "addinvestor"**;  
 }  
  
 @RequestMapping(value = **"/login"**, method = RequestMethod.***GET***)  
 **public** String getLogin(Model model){  
 model.addAttribute(**"login"**,**new** Investor());  
 **return "login"**;  
 }  
  
  
  
 @RequestMapping(value = **"/logging"**, method = RequestMethod.***POST***)  
 **public** String login (@ModelAttribute (**"login"**) Investor investor, Model model) **throws** SQLException {  
  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 InvestorDAO investorDAO = **new** InvestorDAO(daoFactory.getConnection());  
 List<Investor> list = investorDAO.getAll();  
 String Ret = **new** String();  
 **for** (**int** i=0;i<list.size();i++) {  
 **if** (investor.getLogin().equals(list.get(i).getLogin()) && investor.getPassword().equals(list.get(i).getPassword())) {  
 model.addAttribute(**"inv"**, investorDAO.read(investor.getLogin()));  
 Ret = **"sucsesspage"**;  
 **break**;  
 } **else** Ret = **"errorpage"**;  
  
 }  
 **return** Ret;  
 }  
 @RequestMapping(value = **"/investor"**, method = RequestMethod.***GET***)  
 **public** String logging (@RequestParam(value = **"inv"**) String login, Model model) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 InvestorDAO investorDAO = **new** InvestorDAO(daoFactory.getConnection());  
 model.addAttribute(**"inv"**,investorDAO.read(login));  
 **return "investorpage"**;  
 }  
  
  
 @RequestMapping(value = **"/addedpage"**, method = RequestMethod.***POST***)  
 **public** String added(@ModelAttribute(**"investorAttribute"**) Investor investor) **throws** SQLException {  
  
  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 InvestorDAO investorDAO = **new** InvestorDAO(daoFactory.getConnection());  
 investorDAO.create(investor);  
  
  
 **return "addedpage"**;  
 }  
  
}

InvestmentController.java

@Controller  
@RequestMapping(**"/investments"**)  
**public class** InvestmentController {  
 @RequestMapping(value = **"/all"**, method = RequestMethod.***GET***)  
 **public** String getList(@RequestParam(value = **"login"**, required = **true**)String login, Model model) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 InvestmentDAO investmentDAO = **new** InvestmentDAO(daoFactory.getConnection());  
 InvestorDAO investorDAO = **new** InvestorDAO(daoFactory.getConnection());  
 Investor investor = investorDAO.read(login);  
 List<Investment> investments = investmentDAO.getAllofInvestor(investor);  
 model.addAttribute(**"investments"**, investments);  
 model.addAttribute(**"login1"**,login);  
  
 **return "list"**;  
 }  
  
  
  
 @RequestMapping(value = **"/addobject"**, method = RequestMethod.***GET***)  
 **public** String getAddObj(@RequestParam(value = **"login1"**, required = **true**) String login,Model model) **throws** SQLException {  
 model.addAttribute(**"obj"**, **new** OObject());  
 model.addAttribute(**"login1"**,login);  
  
 **return "addobject"**;  
  
 }  
  
 @RequestMapping(value = **"/addinvest"**, method = RequestMethod.***POST***)  
 **public** String add(@RequestParam(value = **"login1"**, required = **true**) String login,@ModelAttribute(**"obj"**) OObject object, Model model) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 ObjectDAO objectDAO = **new** ObjectDAO(daoFactory.getConnection());  
 objectDAO.create(object);  
 model.addAttribute(**"object"**, objectDAO.read(object.getName()));  
 System.***out***.println(objectDAO.read(object.getName()));  
 model.addAttribute(**"invest"**, **new** Investment());  
 model.addAttribute(**"login1"**,login);  
  
  
  
 **return "addinvest"**;  
 }  
  
 @RequestMapping(value = **"/addedinvest"**, method = RequestMethod.***POST***)  
 **public** String added(@RequestParam(value = **"id"** , required = **true**) Integer id,@RequestParam(value = **"login1"**, required = **true**) String login, @ModelAttribute(**"invest"**) Investment investment, Model model) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 ObjectDAO objectDAO = **new** ObjectDAO(daoFactory.getConnection());  
 InvestmentDAO investmentDAO = **new** InvestmentDAO(daoFactory.getConnection());  
 InvestorDAO investorDAO = **new** InvestorDAO(daoFactory.getConnection());  
 investment.setOObject(objectDAO.read(id));  
 investment.setInvestor(investorDAO.read(login));  
 investmentDAO.create(investment);  
 model.addAttribute(**"login1"**,login);  
 **return "addedinvest"**;  
 }  
  
 @RequestMapping(value = **"/delete"**, method = RequestMethod.***GET***)  
 **public** String delete(@RequestParam(value = **"id"**) Integer id,@RequestParam(value = **"login1"**, required = **true**) String login, Model model) **throws** SQLException {  
 MySQLDAOFactory mySQLDAOFactory = **new** MySQLDAOFactory();  
 InvestmentDAO investmentDAO = **new** InvestmentDAO(mySQLDAOFactory.getConnection());  
 ObjectDAO objectDAO = **new** ObjectDAO(mySQLDAOFactory.getConnection());  
 objectDAO.delete(investmentDAO.read(id).getOObject());  
 investmentDAO.delete(investmentDAO.read(id));  
  
 model.addAttribute(**"login1"**,login);  
 **return "delete"**;  
 }  
}

InvestorDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.InvDAO;  
**import** com.dao.tableClasses.Investor;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;

**public class** InvestorDAO **implements** InvDAO {  
  
 **private final** Connection **connection**;  
  
 **public** InvestorDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(Investor investor) {  
 String sql = **"INSERT INTO investor (investor\_login, investor\_password, investor\_name , investor\_surname,investor\_skype,investor\_phone,investor\_dadname) VALUES (?,?,?,?,?,?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, investor.getLogin());  
 prStm.setString(2, investor.getPassword());  
 prStm.setString(3, investor.getName());  
 prStm.setString(4, investor.getSurname());  
 prStm.setString(5, investor.getSkype());  
 prStm.setString(6, investor.getPhone());  
 prStm.setString(7, investor.getDadname());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Investor read(**int** key) **throws** SQLException {  
 String sql = **"SELECT \* FROM investor WHERE investor\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Investor investor = **new** Investor();  
 investor.setLogin(rs.getString(**"investor\_login"**));  
 investor.setPassword(rs.getString(**"investor\_password"**));  
 investor.setName(rs.getString(**"investor\_name"**));  
 investor.setSurname(rs.getString(**"investor\_surname"**));  
 investor.setId(rs.getInt(**"investor\_id"**));  
 investor.setSkype(rs.getString(**"investor\_skype"**));  
 investor.setPhone(rs.getString(**"investor\_phone"**));  
 investor.setDadname(rs.getString(**"investor\_dadname"**));  
 **return** investor;  
 }  
  
 @Override  
 **public** Investor read(String key) **throws** SQLException {  
 String sql = **"SELECT \* FROM investor WHERE investor\_login = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setString(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Investor investor = **new** Investor();  
 investor.setLogin(rs.getString(**"investor\_login"**));  
 investor.setPassword(rs.getString(**"investor\_password"**));  
 investor.setName(rs.getString(**"investor\_name"**));  
 investor.setSurname(rs.getString(**"investor\_surname"**));  
 investor.setId(rs.getInt(**"investor\_id"**));  
 investor.setSkype(rs.getString(**"investor\_skype"**));  
 investor.setPhone(rs.getString(**"investor\_phone"**));  
 investor.setDadname(rs.getString(**"investor\_dadname"**));  
 **return** investor;  
 }  
  
 @Override  
 **public void** update(Investor investor) {  
 String sql = **"UPDATE investor SET investor\_login=?, investor\_password=?, investor\_name=?, investor\_surname=?,investor\_skype=?,investor\_phone=?,investor\_dadname=? WHERE investor\_id="** + investor.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, investor.getLogin());  
 prStm.setString(2, investor.getPassword());  
 prStm.setString(3, investor.getName());  
 prStm.setString(4, investor.getSurname());  
 prStm.setString(5, investor.getSkype());  
 prStm.setString(6, investor.getPhone());  
 prStm.setString(4, investor.getDadname());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(Investor investor) {  
 String sql = **"DELETE FROM investor WHERE investor\_id="** + investor.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<Investor> getAll() **throws** SQLException {  
 String sql = **"SELECT \* FROM investor"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<Investor> list = **new** LinkedList<Investor>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Investor investor = **new** Investor();  
 investor.setLogin(rs.getString(**"investor\_login"**));  
 investor.setPassword(rs.getString(**"investor\_password"**));  
 investor.setName(rs.getString(**"investor\_name"**));  
 investor.setSurname(rs.getString(**"investor\_surname"**));  
 investor.setId(rs.getInt(**"investor\_id"**));  
 investor.setSkype(rs.getString(**"investor\_skype"**));  
 investor.setPhone(rs.getString(**"investor\_phone"**));  
 investor.setDadname(rs.getString(**"investor\_dadname"**));  
 list.add(investor);  
 }  
 **return** list;  
 }  
}

MessageDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.InvDAO;  
**import** com.dao.interfaces.MesDAO;  
**import** com.dao.interfaces.ObjDAO;  
**import** com.dao.tableClasses.Message;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;

**public class** MessageDAO **implements** MesDAO {  
  
 **private final** Connection **connection**;  
  
 **public** MessageDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(Message message) {  
 String sql = **"INSERT INTO message (message\_subject, message\_date, message\_text, message\_sender\_investor\_id, message\_recipient\_investor\_id) VALUES (?,?,?,?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, message.getSubject());  
 prStm.setDate(2, message.getDate());  
 prStm.setString(3, message.getText());  
 prStm.setInt(4, message.getSender().getId());  
 prStm.setInt(5, message.getRecipient().getId());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Message read(**int** key) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM message WHERE message\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Message message = **new** Message();  
 message.setSubject(rs.getString(**"message\_subject"**));  
 message.setDate(rs.getDate(**"message\_date"**));  
 message.setText(rs.getString(**"message\_text"**));  
 message.setSender(in.read(rs.getInt(**"message\_sender\_investor\_id"**)));  
 message.setRecipient(in.read(rs.getInt(**"message\_recipient\_investor\_id"**)));  
 message.setId(rs.getInt(**"message\_id"**));  
 **return** message;  
 }  
  
 @Override  
 **public void** update(Message message) {  
 String sql = **"UPDATE message SET message\_subject=?, message\_date=?, message\_date=?, message\_text=?,message\_sender\_investor\_id=?,message\_recipient\_investor\_id=? WHERE investment\_id="** + message.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, message.getSubject());  
 prStm.setDate(2, message.getDate());  
 prStm.setString(3, message.getText());  
 prStm.setInt(4, message.getSender().getId());  
 prStm.setInt(5, message.getRecipient().getId());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(Message message) {  
 String sql = **"DELETE FROM message WHERE message\_id="** + message.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<Message> getAll() **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 ObjDAO ob = daoFactory.getObjDao(con);  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM message"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<Message> list = **new** LinkedList<Message>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Message message = **new** Message();  
 message.setSubject(rs.getString(**"message\_subject"**));  
 message.setDate(rs.getDate(**"message\_date"**));  
 message.setText(rs.getString(**"message\_text"**));  
 message.setSender(in.read(rs.getInt(**"message\_sender\_investor\_id"**)));  
 message.setRecipient(in.read(rs.getInt(**"message\_recipient\_investor\_id"**)));  
 message.setId(rs.getInt(**"message\_id"**));  
 list.add(message);  
 }  
 **return** list;  
 }  
}

CommentDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.ComDAO;  
**import** com.dao.interfaces.InvDAO;  
**import** com.dao.tableClasses.Comment;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;

**public class** CommentDAO **implements** ComDAO {  
  
 **private final** Connection **connection**;  
  
 **public** CommentDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(Comment comment) **throws** SQLException {  
 String sql = **"INSERT INTO comment (comment\_branch\_of\_forum, comment\_date\_of\_change, comment\_time\_of\_change , comment\_text , investor\_id) VALUES (?,?,?,?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, comment.getBranchofforum());  
 prStm.setDate(2, comment.getDateofchange());  
 prStm.setTime(3, comment.getTimeofchange());  
 prStm.setString(4, comment.getComment());  
 prStm.setInt(5, comment.getInvestor().getId());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Comment read(**int** key) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 InvDAO IDAO = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM comment WHERE comment\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Comment comment = **new** Comment();  
 comment.setBranchofforum(rs.getString(**"comment\_branch\_of\_forum"**));  
 comment.setDateofchange(rs.getDate(**"comment\_date\_of\_change"**));  
 comment.setTimeofchange(rs.getTime(**"comment\_time\_of\_change"**));  
 comment.setComment(rs.getString(**"comment\_text"**));  
 comment.setInvestor(IDAO.read(rs.getInt(**"investor\_id"**)));  
 comment.setId(rs.getInt(**"comment\_id"**));  
 **return** comment;  
 }  
  
 @Override  
 **public void** update(Comment comment) {  
 String sql = **"UPDATE comment SET comment\_branch\_of\_forum=?, comment\_date\_of\_change=?, comment\_time\_of\_change=?, comment\_text=? investor\_id=? WHERE comment\_id="** + comment.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, comment.getBranchofforum());  
 prStm.setDate(2, comment.getDateofchange());  
 prStm.setTime(3, comment.getTimeofchange());  
 prStm.setString(4, comment.getComment());  
 prStm.setInt(5, comment.getInvestor().getId());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(Comment comment) {  
 String sql = **"DELETE FROM comment WHERE comment\_id="** + comment.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<Comment> getAll() **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 InvDAO IDAO = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM comment"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<Comment> list = **new** LinkedList<Comment>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Comment comment = **new** Comment();  
 comment.setBranchofforum(rs.getString(**"comment\_branch\_of\_forum"**));  
 comment.setDateofchange(rs.getDate(**"comment\_date\_of\_change"**));  
 comment.setTimeofchange(rs.getTime(**"comment\_time\_of\_change"**));  
 comment.setComment(rs.getString(**"comment\_text"**));  
 comment.setInvestor(IDAO.read(rs.getInt(**"investor\_id"**)));  
  
 comment.setId(rs.getInt(**"comment\_id"**));  
 list.add(comment);  
 }  
 **return** list;  
 }  
}

OperationDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.InvDAO;  
**import** com.dao.interfaces.OperDAO;  
**import** com.dao.tableClasses.Operation;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
**public class** OperationDAO **implements** OperDAO {  
  
 **private final** Connection **connection**;  
  
 **public** OperationDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(Operation operation) {  
 String sql = **"INSERT INTO operation (operation\_type\_name, operation\_summ, operation\_date , investor\_id) VALUES (?,?,?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, operation.getName());  
 prStm.setInt(2, operation.getSumm());  
 prStm.setDate(3, operation.getDate());  
 prStm.setInt(4, operation.getInvestor().getId());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Operation read(**int** key) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM operation WHERE operation\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Operation operation = **new** Operation();  
 operation.setName(rs.getString(**"operation\_type\_name"**));  
 operation.setSumm(rs.getInt(**"operation\_summ"**));  
 operation.setDate(rs.getDate(**"operation\_date"**));  
 operation.setInvestor(in.read(rs.getInt(**"investor\_id"**)));  
  
 operation.setId(rs.getInt(**"operation\_id"**));  
 **return** operation;  
 }  
  
 @Override  
 **public void** update(Operation operation) {  
 String sql = **"UPDATE operation SET i=?, operation\_type\_name=?, operation\_summ=?, operation\_date=?, investor\_id=? WHERE investor\_id="** + operation.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, operation.getName());  
 prStm.setInt(2, operation.getSumm());  
 prStm.setDate(3, operation.getDate());  
 prStm.setInt(4, operation.getInvestor().getId());  
 prStm.setInt(4, operation.getInvestor().getId());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(Operation operation) {  
 String sql = **"DELETE FROM operation WHERE operation\_id="** + operation.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<Operation> getAll() **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM operation"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<Operation> list = **new** LinkedList<Operation>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Operation operation = **new** Operation();  
 operation.setName(rs.getString(**"operation\_type\_name"**));  
 operation.setSumm(rs.getInt(**"operation\_summ"**));  
 operation.setDate(rs.getDate(**"operation\_date"**));  
 operation.setInvestor(in.read(rs.getInt(**"investor\_id"**)));  
  
 operation.setId(rs.getInt(**"operation\_id"**));  
 list.add(operation);  
 }  
 **return** list;  
 }  
}

InvestmentDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.InvDAO;  
**import** com.dao.interfaces.InvestDAO;  
**import** com.dao.interfaces.ObjDAO;  
**import** com.dao.tableClasses.Investment;  
**import** com.dao.tableClasses.Investor;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
**public class** InvestmentDAO **implements** InvestDAO {  
  
 **private final** Connection **connection**;  
  
 **public** InvestmentDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(Investment investment) {  
 String sql = **"INSERT INTO investment (investment\_contract\_number, object\_id, investor\_id) VALUES (?,?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setInt(1, investment.getContractnumber());  
 prStm.setInt(2, investment.getOObject().getId());  
 prStm.setInt(3, investment.getInvestor().getId());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Investment read(**int** key) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 ObjDAO ob = daoFactory.getObjDao(con);  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM investment WHERE investment\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 Investment investment = **new** Investment();  
 investment.setContractnumber(rs.getInt(**"investment\_contract\_number"**));  
 investment.setOObject(ob.read(rs.getInt(**"object\_id"**)));  
 investment.setInvestor(in.read(rs.getInt(**"investor\_id"**)));  
 investment.setId(rs.getInt(**"investment\_id"**));  
 **return** investment;  
 }  
  
 @Override  
 **public void** update(Investment investment) {  
 String sql = **"UPDATE investment SET investment\_contract\_number=?, object\_id=?, investor\_id=? WHERE investment\_id="** + investment.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setInt(1, investment.getContractnumber());  
 prStm.setInt(2, investment.getOObject().getId());  
 prStm.setInt(3, investment.getInvestor().getId());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(Investment investment) {  
 String sql = **"DELETE FROM investment WHERE investment\_id="** + investment.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<Investment> getAll() **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 ObjDAO ob = daoFactory.getObjDao(con);  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM investment"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<Investment> list = **new** LinkedList<Investment>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Investment investment = **new** Investment();  
 investment.setContractnumber(rs.getInt(**"investment\_contract\_number"**));  
 investment.setOObject(ob.read(rs.getInt(**"object\_id"**)));  
 investment.setInvestor(in.read(rs.getInt(**"investor\_id"**)));  
 investment.setId(rs.getInt(**"investment\_id"**));  
 list.add(investment);  
 }  
 **return** list;  
 }  
  
 @Override  
 **public** List<Investment> getAllofInvestor(Investor investor) **throws** SQLException {  
 MySQLDAOFactory daoFactory = **new** MySQLDAOFactory();  
 Connection con = daoFactory.getConnection();  
 ObjDAO ob = daoFactory.getObjDao(con);  
 InvDAO in = daoFactory.getInvDao(con);  
 String sql = **"SELECT \* FROM investment WHERE investor\_id=?"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, investor.getId());  
 List<Investment> list = **new** LinkedList<Investment>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 Investment invest = **new** Investment();  
 invest.setContractnumber(rs.getInt(**"investment\_contract\_number"**));  
 invest.setOObject(ob.read(rs.getInt(**"object\_id"**)));  
 invest.setInvestor(in.read(rs.getInt(**"investor\_id"**)));  
 invest.setId(rs.getInt(**"investment\_id"**));  
 list.add(invest);  
 }  
 **return** list;  
 }  
}

ObjectDAO.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.ObjDAO;  
**import** com.dao.tableClasses.OObject;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
**public class** ObjectDAO **implements** ObjDAO {  
  
 **private final** Connection **connection**;  
  
 **public** ObjectDAO(Connection connection) {  
 **this**.**connection** = connection;  
 }  
  
 @Override  
 **public void** create(OObject object) {  
 String sql = **"INSERT INTO object (object\_name,object\_text) VALUES (?,?)"**;  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, object.getName());  
 prStm.setString(2, object.getText());  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** OObject read(**int** key) **throws** SQLException {  
 String sql = **"SELECT \* FROM object WHERE object\_id = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setInt(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 OObject object = **new** OObject();  
 object.setName(rs.getString(**"object\_name"**));  
 object.setText(rs.getString(**"object\_text"**));  
 object.setId(rs.getInt(**"object\_id"**));  
 **return** object;  
 }  
  
 @Override  
 **public** OObject read(String key) **throws** SQLException {  
 String sql = **"SELECT \* FROM object WHERE object\_name = ?;"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 stm.setString(1, key);  
 ResultSet rs = stm.executeQuery();  
 rs.next();  
 OObject object = **new** OObject();  
 object.setName(rs.getString(**"object\_name"**));  
 object.setText(rs.getString(**"object\_text"**));  
 object.setId(rs.getInt(**"object\_id"**));  
 **return** object;  
 }  
  
 @Override  
 **public void** update(OObject object) {  
 String sql = **"UPDATE object SET object\_name=?,object\_text=? WHERE object\_id="** + object.getId();  
 **try** {  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.setString(1, object.getName());  
 prStm.setString(2, object.getText());  
 prStm.execute();  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(OObject object) {  
 String sql = **"DELETE FROM object WHERE object\_id="** + object.getId();  
 **try**{  
 PreparedStatement prStm = **connection**.prepareStatement(sql);  
 prStm.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** List<OObject> getAll() **throws** SQLException {  
 String sql = **"SELECT \* FROM object"**;  
 PreparedStatement stm = **connection**.prepareStatement(sql);  
 List<OObject> list = **new** LinkedList<OObject>();  
 ResultSet rs = stm.executeQuery();  
 **while**(rs.next()) {  
 OObject object = **new** OObject();  
 object.setName(rs.getString(**"object\_name"**));  
 object.setName(rs.getString(**"object\_text"**));  
 object.setId(rs.getInt(**"object\_id"**));  
 list.add(object);  
 }  
 **return** list;  
 }  
}

MySQLDAOFactory.java

**package** com.dao.managers;  
  
**import** com.dao.interfaces.\*;  
  
**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
  
**public class** MySQLDAOFactory **implements** DAOFactory {  
  
 **private** String **user** = **"root"**;  
 **private** String **password** = **"aceraspire"**;  
 **private** String **url** = **"jdbc:mysql://localhost:3306/CSBC"**;  
 **private** String **driver** = **"com.mysql.jdbc.Driver"**;  
  
 **public** MySQLDAOFactory() {  
 **try** {  
 Class.*forName*(**driver**);*//Регистрируем драйвер* } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Connection getConnection() **throws** SQLException {  
 **return** DriverManager.*getConnection*(**url**, **user**, **password**);  
 }  
  
 @Override  
 **public** ComDAO getComDao(Connection connection) {  
  
 **return new** CommentDAO(connection);  
 }  
  
 @Override  
 **public** InvDAO getInvDao(Connection connection) {  
  
 **return new** InvestorDAO(connection);  
 }  
 @Override  
 **public** InvestDAO getInvestDao(Connection connection) {  
 **return new** InvestmentDAO(connection);  
 }  
 @Override  
 **public** MesDAO getMesDao(Connection connection) {  
 **return new** MessageDAO(connection);  
 }  
  
 @Override  
 **public** ObjDAO getObjDao(Connection connection) {  
 **return new** ObjectDAO(connection);  
 }  
  
 @Override  
 **public** OperDAO getOperDao(Connection connection) {  
 **return new** OperationDAO(connection);  
 }  
  
}