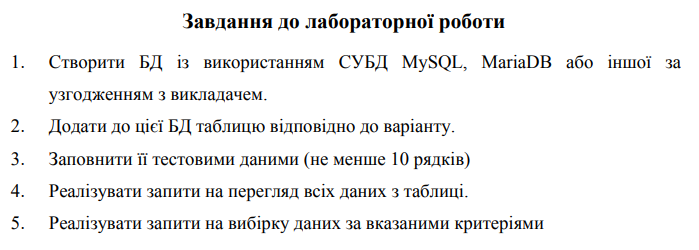
**Лабораторна робота №3**

Створення бази даних та sql запитів для обробки даних.

**Мета роботи:** отримати навички створення таблиці БД, ознайомитись з типами

даних, що можуть зберігатись в БД

*Завдання*



**Варіант 5:**

Book: id, Назва, Автор, Видавництво, Рік видання, Кількість сторінок, Ціна.

Вивести:

a) список книг заданого автора в порядку зростання року видання;

b) список книг, що видані заданим видавництвом;

c) список книг, що випущені після заданого року;

d) список авторів в алфавітному порядку;

e) список видавництв, книги яких зареєстровані в системі без повторів;

f) для кожного видавництва визначити список книг, виданих ним

**Хід роботи**

1. **Текст програми**

***com.Main.java***

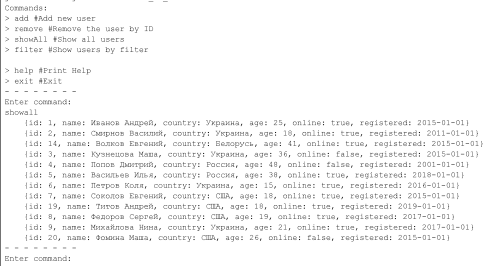
**public class** Main {  
 **private** BookDao **bookDao**;  
  
 **private int limitToShow** = 15;  
  
 **public static void** main(String[] args) {  
 Main main = **new** Main();  
 main.run();  
 }  
  
 **private void** run() {  
 init();  
 runCli();  
 }  
  
 **private void** init() {  
 EntityManagerFactory factory = Persistence.*createEntityManagerFactory*(**"MyPU"**);  
 EntityManager em = factory.createEntityManager();  
  
 **bookDao** = **new** BookDao(em);  
 }  
  
 **private void** runCli() {  
 cliPrintHelp();  
 hr();  
  
 **while** (**true**) {  
 String command = *promptLine*(**"Enter command: "**).trim().toLowerCase();  
  
 **if** (command.equals(**"exit"**)) **break**;  
 **else** useCommand(command);  
  
 hr();  
 }  
  
 }  
  
 **private void** useCommand(String command) {  
 **switch** (command) {  
 **case "showall"**:  
 cliShowAllBooks(); **break**;  
 **case "add"**:  
 cliAddBook(); **break**;  
 **case "remove"**:  
 cliRemoveBook(); **break**;  
 **case "filter"**:  
 cliShowBooksByFilter(); **break**;  
 **case "limit"**:  
 cliSetLimit(); **break**;  
 **case "help"**:  
 cliPrintHelp(); **break**;  
 **default**:  
 System.***out***.println(**"Command not found. Try again: "**);  
 **break**;  
 }  
 }  
  
 **private void** cliAddBook() {  
 Book book = createNewBook();  
 **bookDao**.add(book);  
 System.***out***.println(**"Done!"**);  
 }  
  
 **private void** cliRemoveBook() {  
 **int** bookId = *promptInt*(**"Enter the book id"**);  
 **bookDao**.delete(bookId);  
 System.***out***.println(**"Done!"**);  
 }  
  
 **private void** cliShowAllBooks() {  
 **var** books = **bookDao**.findAll(**limitToShow**);  
 printBooks(books);  
 }  
  
 **private void** cliShowBooksByFilter() {  
 printFiltersHelp();  
 String filter = *promptLine*(**"Select Filter:"**).toLowerCase();  
  
 **switch** (filter) {  
 **case "a"**: {  
 String author = *promptLine*(**"Author:"**);  
 **var** books = **bookDao**.findAllByAuthor(author, **limitToShow**);  
 printBooks(books);  
  
 **break**;  
 }  
 **case "b"**: {  
 String publisher = *promptLine*(**"Publisher:"**);  
 **var** books = **bookDao**.findAllByPublisher(publisher, **limitToShow**);  
 printBooks(books);  
  
 **break**;  
 }  
 **case "c"**: {  
 LocalDate date = *promptDate*(**"Publish Date:"**);  
 **var** books = **bookDao**.findAllByDate(date, **limitToShow**);  
 printBooks(books);  
  
 **break**;  
 }  
 **case "d"**:  
 **var** authors = **bookDao**.findAllAuthors(**limitToShow**);  
 authors.forEach(System.***out***::println);  
  
 **break**;  
 **case "e"**: {  
 **var** publisher = **bookDao**.findAllPublishers(**limitToShow**);  
 publisher.forEach(System.***out***::println);  
  
 **break**;  
 }  
 **case "f"**: {  
 **var** books = **bookDao**.findAll(**limitToShow**);  
  
 Map<String, HashSet<Book>> map = books.stream()  
 .collect(Collectors.*groupingBy*(Book::getPublisher, Collectors.*toCollection*(HashSet::**new**)));  
  
 map.forEach((publisher, publishersBook) -> {  
 System.***out***.println(**" - "** + publisher);  
 printBooks(publishersBook);  
 });  
  
 **break**;  
 }  
 **default**:  
 System.***out***.println(**"Can't find the filter"**);  
  
 **break**;  
 }  
  
 }  
  
 **private void** cliSetLimit() {  
 System.***out***.println(**"Current limit: "** + **limitToShow**);  
 **limitToShow** = *promptInt*(**"Enter new limit: "**);  
 }  
  
 **private** Book createNewBook() {  
 Book book = **new** Book();  
  
 book.setName(*promptLine*(**"Name:"**));  
 book.setAuthor(*promptLine*(**"Author:"**));  
 book.setPublisher(*promptLine*(**"Publisher"**));  
 book.setPublishDate(*promptDate*(**"Publish Date:"**));  
 book.setPages(*promptInt*(**"Pages:"**));  
 book.setPrice(*promptDouble*(**"Price:"**));  
  
 **return** book;  
 }  
  
 **private void** printBooks(Collection<Book> books) {  
 **if** (books.isEmpty()) {  
 System.***out***.println(**"There are no users"**);  
 **return**;  
 }  
  
 books.forEach((item) -> System.***out***.println(**"\t"** + item));  
 }  
  
 **private void** cliPrintHelp() {  
 System.***out***.println(**"Commands:"**);  
 System.***out***.println(**"> showAll #Show all books"**);  
 System.***out***.println(**"> add #Add new book"**);  
 System.***out***.println(**"> remove #Remove the book by ID"**);  
 System.***out***.println(**"> filter #Show books by filter"**);  
 System.***out***.println();  
 System.***out***.println(**"> limit #Set Limit to show"**);  
 System.***out***.println(**"> help #Print Help"**);  
 System.***out***.println(**"> exit #Exit"**);  
 }  
  
 **private void** printFiltersHelp() {  
 System.***out***.println(**"Filters:"**);  
 System.***out***.println(**"> a #Список книг заданого автора в порядку зростання року видання"**);  
 System.***out***.println(**"> b #Список книг, що видані заданим видавництвом"**);  
 System.***out***.println(**"> c #Список книг, що випущені після заданої дати"**);  
 System.***out***.println(**"> d #Список авторів в алфавітному порядку"**);  
 System.***out***.println(**"> e #Список видавництв, книги яких зареєстровані в системі без повторів"**);  
 System.***out***.println(**"> f #Для кожного видавництва визначити список книг, виданих ним"**);  
 }  
  
}

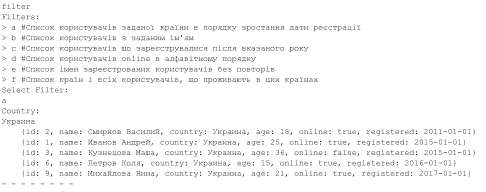
***schemas.Book.java***  
  
@Entity  
**public class** Book {  
 @Id  
 @Column(name = **"id"**, nullable = **false**)  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 **private int** id;  
  
 @Basic **private** String name;  
 @Basic **private** String author;  
 @Basic **private** String publisher;  
 @Basic **private** LocalDate publishDate;  
 @Basic **private int** pages;  
 @Basic **private double** price;  
  
 @Override  
 **public** String toString() {  
 **return "id="** + id +  
 **", name='"** + name + **'\''** +  
 **", author='"** + author + **'\''** +  
 **", publisher='"** + publisher + **'\''** +  
 **", publishDate="** + publishDate +  
 **", pages="** + pages +  
 **", price="** + price;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (!(o **instanceof** Book)) **return false**;  
 Book book = (Book) o;  
 **return** id == book.id;  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** Objects.hash(id);  
 }  
  
 **public int** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.id = id;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
  
 **public** String getAuthor() {  
 **return** author;  
 }  
  
 **public void** setAuthor(String author) {  
 **this**.author = author;  
 }  
  
 **public** String getPublisher() {  
 **return** publisher;  
 }  
  
 **public void** setPublisher(String publisher) {  
 **this**.publisher = publisher;  
 }  
  
 **public** LocalDate getPublishDate() {  
 **return** publishDate;  
 }  
  
 **public void** setPublishDate(LocalDate publishDate) {  
 **this**.publishDate = publishDate;  
 }  
  
 **public int** getPages() {  
 **return** pages;  
 }  
  
 **public void** setPages(**int** pages) {  
 **this**.pages = pages;  
 }  
  
 **public double** getPrice() {  
 **return** price;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.price = price;  
 }  
  
}

dao.BookDao.java

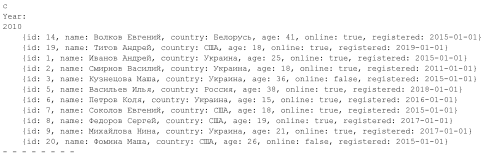
**public class** BookDao {  
 **private** EntityManager **em**;  
  
 **public** BookDao(EntityManager em) {  
 **this**.**em** = em;  
 }  
  
 **public** List<Book> findAll(**int** limit) {  
 **return em**.createQuery(**"select b from Book b"**, Book.**class**)  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public void** add(Book book) {  
 **em**.getTransaction().begin();  
 **em**.persist(book);  
 **em**.getTransaction().commit();  
 }  
  
 **public void** delete(**int** id) {  
 **em**.getTransaction().begin();  
 **var** toDelete = **em**.find(Book.**class**, id);  
 **em**.remove(toDelete);  
 **em**.getTransaction().commit();  
 }  
  
 **public** List<Book> findAllByAuthor(String author, **int** limit) {  
 **var** query = **em**.createQuery(**"select b from Book b where b.author like :author order by b.publishDate"**, Book.**class**);  
 query.setParameter(**"author"**, **"%"** + author + **"%"**);  
  
 **return** query  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public** List<Book> findAllByPublisher(String publisher, **int** limit) {  
 **var** query = **em**.createQuery(**"select b from Book b where b.publisher like :publisher"**, Book.**class**);  
 query.setParameter(**"publisher"**, **"%"** + publisher + **"%"**);  
  
 **return** query  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public** List<Book> findAllByDate(LocalDate date, **int** limit) {  
 **var** query = **em**.createQuery(**"select b from Book b where b.publishDate > :date"**, Book.**class**);  
 query.setParameter(**"date"**, date);  
  
 **return** query  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public** List<String> findAllAuthors(**int** limit) {  
 **return em**.createQuery(**"select distinct b.author from Book b order by b.author"**, String.**class**)  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public** List<String> findAllPublishers(**int** limit) {  
 **return em**.createQuery(**"select distinct b.publisher from Book b"**, String.**class**)  
 .setMaxResults(limit)  
 .getResultList();  
 }  
  
 **public void** delete(Book book) {  
 delete(book.getId());  
 }  
  
}

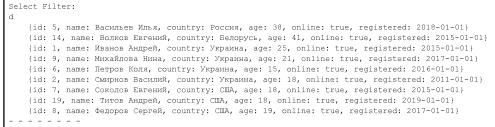
1. **Результат виконання роботи**



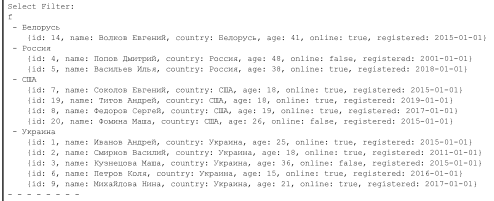












**Висновок:** на цій лабораторній роботі я отримав навички створення таблиці БД, ознайомся з типами даних, що можуть зберігатись в БД.