Внимание! У разных вариантов разный текст задания!

Доработать программу из [лабораторной работы №6](https://se.ifmo.ru/courses/programming#lab6) следующим образом:

1. Организовать хранение коллекции в реляционной СУБД (PostgresQL). Убрать хранение коллекции в файле.
2. Для генерации поля id использовать средства базы данных (sequence).
3. Обновлять состояние коллекции в памяти только при успешном добавлении объекта в БД
4. Все команды получения данных должны работать с коллекцией в памяти, а не в БД
5. Организовать возможность регистрации и авторизации пользователей. У пользователя есть возможность указать пароль.
6. Пароли при хранении хэшировать алгоритмом SHA-256
7. Запретить выполнение команд не авторизованным пользователям.
8. При хранении объектов сохранять информацию о пользователе, который создал этот объект.
9. Пользователи должны иметь возможность просмотра всех объектов коллекции, но модифицировать могут только принадлежащие им.
10. Для идентификации пользователя отправлять логин и пароль с каждым запросом.

Необходимо реализовать многопоточную обработку запросов.

1. Для многопоточного чтения запросов использовать Fixed thread pool
2. Для многопотчной обработки полученного запроса использовать Fixed thread pool
3. Для многопоточной отправки ответа использовать ForkJoinPool
4. Для синхронизации доступа к коллекции использовать синхронизацию чтения и записи с помощью synchronized

**Порядок выполнения работы:**

1. В качестве базы данных использовать PostgreSQL.
2. Для подключения к БД на кафедральном сервере использовать хост pg, имя базы данных - studs, имя пользователя/пароль совпадают с таковыми для подключения к серверу.

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Выводы по работе.

**Вопросы к защите лабораторной работы:**

1. Многопоточность. Класс Thread, интерфейс Runnable. Модификатор synchronized.
2. Методы wait(), notify() класса Object, интерфейсы Lock и Condition.
3. Классы-сихронизаторы из пакета java.util.concurrent.
4. Модификатор volatile. Атомарные типы данных и операции.
5. Коллекции из пакета java.util.concurrent.
6. Интерфейсы Executor, ExecutorService, Callable, Future
7. Пулы потоков
8. JDBC. Порядок взаимодействия с базой данных. Класс DriverManager. Интерфейс Connection
9. Интерфейсы Statement, PreparedStatement, ResultSet, RowSet
10. Шаблоны проектирования.