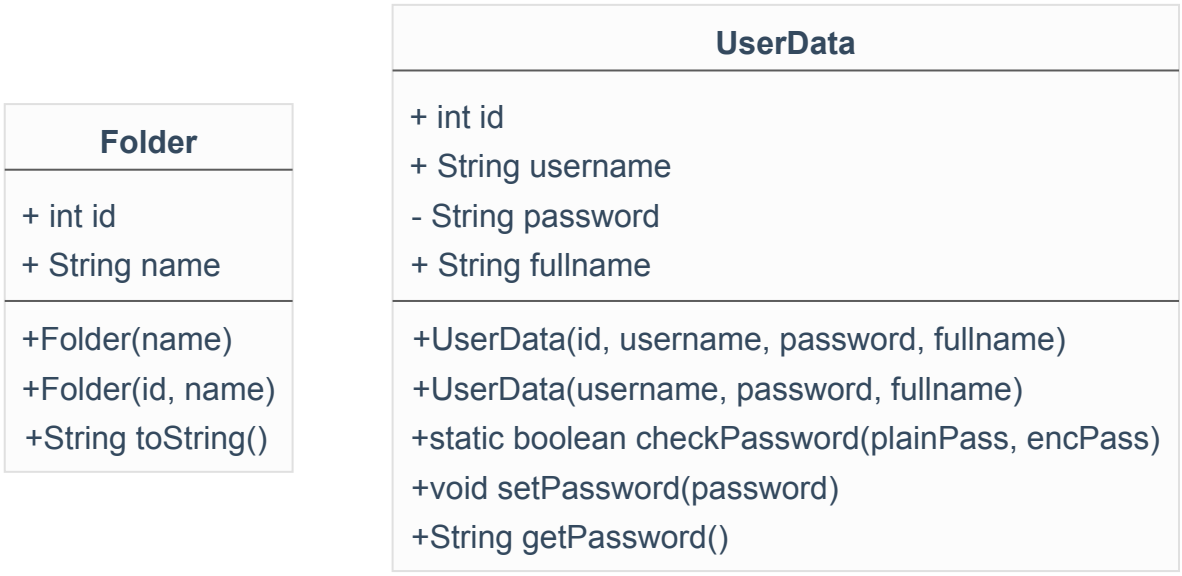
PBO - Modul Praktik 9 - 10

Untuk mengerjakan modul praktik ini dan selanjutnya (hingga GUI), sudah disediakan template yang terdiri dari beberapa class entity dan helpers sebagai patokan pengerjaan kita.

Class Diagram dari template tersebut adalah seperti di bawah ini:



PasswordStore
+ int id + String name + String username - String password + String hashkey - double score - int category + Folder folder
+PasswordStore(name, username, plainPass) +PasswordStore(name, username, plainPass, category, folder) +PasswordStore(id, name, username, encPass, hashkey, score, category, folder) +void setEncryptedPass(encryptedPass, hashkey) +void setPassword(plainPass) +String getEncPassword() +String getPassword() +void setCategory(category) +String getCategory() +int getCategoryCode() +void calculateScore(plainPass) +double getScore() +String fullString() +String toString()

Alur Pengerjaan

- Download template project dari link yang tersedia
- Buat koneksi basis data seperti yang dicontohkan pada PBO - Konsep Akses SQLite DB Berbasis DAO
- Buat juga class `SetupDB` untuk 3 tabel menggunakan SQL berikut:

```
CREATE TABLE if not exists folder (id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT);,

CREATE TABLE if not exists userdata (id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT, password TEXT, fullname TEXT);,

CREATE UNIQUE INDEX if not exists user_username_IDX ON userdata (username);,

CREATE TABLE if not exists passwordstore  (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT, username TEXT, password TEXT, hashkey TEXT,
        score REAL, category INTEGER, user_id INTEGER, folder_id INTEGER,
        CONSTRAINT passwordstore_user_FK FOREIGN KEY (user_id) REFERENCES userdata(id) ON
DELETE RESTRICT ON UPDATE RESTRICT,
        CONSTRAINT passwordstore_folder_FK FOREIGN KEY (folder_id) REFERENCES folder(id)
ON DELETE SET NULL ON UPDATE SET NULL);
```

```
CREATE TABLE if not exists additional (
        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        entry_key TEXT, entry_value TEXT, is_password INTEGER, password_id INTEGER,
CONSTRAINT additional_passwordstore_FK FOREIGN KEY (password_id) REFERENCES
passwordstore(id) ON DELETE CASCADE ON UPDATE CASCADE);
```

- Buat interface *Data Access Object* (**DAO**) untuk keempat entitiy yang ada sebagai berikut.

```
public interface FolderDAO{
        public int createFolder(String foldername);
        public ArrayList<Folder> listAllFolders();
}

public interface UserDAO{
        public int register(String username, String password, String fullname);
        public UserData login(String username, String password);
        public int update(int id, String fullname);
        public int update(int id, String password, String fullname);
        public int delete(int id);
}

public interface PasswordStoreDAO{
        public int addPassword(PasswordStore newPassword, UserData user);
        public ArrayList<PasswordStore> listPassword(UserData user);
        public ArrayList<PasswordStore> findPassword(String name, UserData user);
        public int updatePass(PasswordStore changedPass);
        public int deletePass(PasswordStore deletedPass);
}
```

- Buat implementasi interface DAO-nya. Berikut adalah contoh implementasi DAO untuk model User:

```
public class UserDaoSQLite implements UserDAO {

    public int register(String username, String password, String fullname){
        int id = 0;
        query = "insert into userdata values (null, ?, ?, ?)";
        try {
            stmt = conn.prepareStatement(query);
            UserData newUser = new UserData(username, password, fullname);
            stmt.setString(1, newUser.username);
            stmt.setString(2, newUser.getPassword());
            stmt.setString(3, newUser.fullname);
            stmt.executeUpdate();
            id = stmt.getGeneratedKeys().getInt(1);
        } catch (SQLException e) {
            // kalau ada yang bisa mendeteksi exception username sudah ada, tak tambahin
nilainya
            e.printStackTrace();
        }

        return id;
    }

    public UserData login(String username, String password){
        UserData user = null;
        query = "select * from userdata where username = ?";

        try {
```

```
            stmt = conn.prepareStatement(query);
            stmt.setString(1, username);
            result = stmt.executeQuery();
            if(!result.isAfterLast()){
                result.next();
                if(UserData.checkPassword(password, result.getString("password"))){
                    user = new UserData(result.getInt("id"),
                                result.getString("username"),
                                result.getString("password"),
                                result.getString("fullname"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return user;
    }

    public int update(int id, String fullname){
        query = "update userdata set fullname = ? where id = ?";

        try{
            stmt = conn.prepareStatement(query);
            stmt.setString(1, fullname);
            stmt.setInt(2, id);
            return stmt.executeUpdate();
        } catch (SQLException e){
            e.printStackTrace();
        }

        return 0;
    }

    public int update(int id, String password, String fullname){
        query = "update userdata set fullname = ?, password = ? where id = ?";

        try {
            stmt = conn.prepareStatement(query);
            UserData newUser = new UserData("", password, fullname);
            stmt.setString(1, newUser.fullname);
            stmt.setString(2, newUser.getPassword());
            stmt.setInt(3, id);
            return stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return 0;
    }

    public int delete(int id){
        query = "delete from userdata where id = ?";
        try {
            stmt = conn.prepareStatement(query);
            stmt.setInt(1, id);
            return stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
            return 0;
        }
    }
```

- Anda masih harus membuat implementasi untuk 2 interface DAO yang lain.
- Buat class untuk uji coba dengan melakukan:
    - Tambahkan 2 data user
    - Tambahkan 4 data folder
    - Tambahkan 5 data password untuk masing-masing user
    - Tampilkan semua data password untuk salah satu user dengan menampilkan: nama akun, username, nama category, dan nama folder