SmartLibrary — Tam Java OOP + JDBC + SQLite Proje

Proje Klasör Yapısı

```
SmartLibrary/
 ├── src/
 │    ├── Main.java
 │    ├── database/
 │    │     └── Database.java
 │    ├── models/
 │    │     ├── Book.java
 │    │     ├── Student.java
 │    │     └── Loan.java
 │    ├── repositories/
 │    │     ├── BookRepository.java
 │    │     ├── StudentRepository.java
 │    │     └── LoanRepository.java
 │    └── utils/
 │          └── ConsoleHelper.java
 ├── SmartLibrary.db  (uygulama çalıştırıldığında oluşturulur)
 ├── README.md
 └── .gitignore
```

---

src/models/Book.java

```java
package models;

public class Book {
    private int id;
    private String title;
    private String author;
    private int year;

    public Book() {}

    public Book(int id, String title, String author, int year) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.year = year;
    }
```

```
    public Book(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }

    // getters & setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
    public int getYear() { return year; }
    public void setYear(int year) { this.year = year; }

    @Override
    public String toString() {
        return String.format("[ID:%d] %s — %s (%d)", id, title, author,
year);
    }
}
```

---

src/models/Student.java

```
package models;

public class Student {
    private int id;
    private String name;
    private String department;

    public Student() {}

    public Student(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public Student(String name, String department) {
        this.name = name;
        this.department = department;
    }

    // getters & setters
    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 getDepartment() { return department; }
    public void setDepartment(String department) { this.department =
department; }

    @Override
    public String toString() {
        return String.format("[ID:%d] %s — %s", id, name, department);
    }
}
```

---

src/models/Loan.java

```
package models;

public class Loan {
    private int id;
    private int bookId;
    private int studentId;
    private String dateBorrowed;
    private String dateReturned;

    public Loan() {}

    public Loan(int id, int bookId, int studentId, String dateBorrowed,
String dateReturned) {
        this.id = id;
        this.bookId = bookId;
        this.studentId = studentId;
        this.dateBorrowed = dateBorrowed;
        this.dateReturned = dateReturned;
    }

    public Loan(int bookId, int studentId, String dateBorrowed) {
        this.bookId = bookId;
        this.studentId = studentId;
        this.dateBorrowed = dateBorrowed;
    }

    // getters & setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public int getBookId() { return bookId; }
    public void setBookId(int bookId) { this.bookId = bookId; }
    public int getStudentId() { return studentId; }
    public void setStudentId(int studentId) { this.studentId = studentId; }
    public String getDateBorrowed() { return dateBorrowed; }
    public void setDateBorrowed(String dateBorrowed) { this.dateBorrowed =
dateBorrowed; }
```

```
    public String getDateReturned() { return dateReturned; }
    public void setDateReturned(String dateReturned) { this.dateReturned =
dateReturned; }

    @Override
    public String toString() {
        return String.format("[LoanID:%d] BookID:%d StudentID:%d Borrowed:%s
Returned:%s",
                id, bookId, studentId, dateBorrowed, (dateReturned == null ?
"-" : dateReturned));
    }
}
```

---

src/database/Database.java

```
package database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Database {
    private static final String DB_URL = "jdbc:sqlite:SmartLibrary.db";

    public static Connection connect() throws SQLException {
        return DriverManager.getConnection(DB_URL);
    }

    public static void createTables() {
        String bookTable = "CREATE TABLE IF NOT EXISTS books ("
                + "id INTEGER PRIMARY KEY AUTOINCREMENT,"
                + "title TEXT NOT NULL,"
                + "author TEXT NOT NULL,"
                + "year INTEGER" + ");";

        String studentTable = "CREATE TABLE IF NOT EXISTS students ("
                + "id INTEGER PRIMARY KEY AUTOINCREMENT,"
                + "name TEXT NOT NULL,"
                + "department TEXT" + ");";

        String loanTable = "CREATE TABLE IF NOT EXISTS loans ("
                + "id INTEGER PRIMARY KEY AUTOINCREMENT,"
                + "bookId INTEGER NOT NULL,"
                + "studentId INTEGER NOT NULL,"
                + "dateBorrowed TEXT NOT NULL,"
                + "dateReturned TEXT" + ");";

        try (Connection conn = connect(); Statement stmt =
```

```
conn.createStatement()) {
            stmt.execute(bookTable);
            stmt.execute(studentTable);
            stmt.execute(loanTable);
        } catch (SQLException e) {
            System.out.println("[DB] Tablolar oluşturulurken hata: " +
e.getMessage());
        }
    }
}
```

---

src/repositories/BookRepository.java

```
package repositories;

import database.Database;
import models.Book;

import java.sql.*;
import java.util.ArrayList;
import java.util.Optional;

public class BookRepository extends BaseRepository {

    public void add(Book book) {
        String sql = "INSERT INTO books(title, author, year) VALUES(?,?,?)";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setString(1, book.getTitle());
            ps.setString(2, book.getAuthor());
            ps.setInt(3, book.getYear());
            ps.executeUpdate();
            System.out.println("Kitap eklendi.");
        } catch (SQLException e) {
            System.out.println("[BookRepo] Ekleme hatası: " +
e.getMessage());
        }
    }

    public void update(Book book) {
        String sql = "UPDATE books SET title = ?, author = ?, year = ? WHERE
id = ?";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setString(1, book.getTitle());
            ps.setString(2, book.getAuthor());
            ps.setInt(3, book.getYear());
            ps.setInt(4, book.getId());
            int changed = ps.executeUpdate();
```

```
            System.out.println(changed > 0 ? "Kitap güncellendi." :
"Güncelleme yapılmadı (ID bulunamadı).");
        } catch (SQLException e) {
            System.out.println("[BookRepo] Güncelleme hatası: " +
e.getMessage());
        }
    }

    public void delete(int id) {
        String sql = "DELETE FROM books WHERE id = ?";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setInt(1, id);
            int changed = ps.executeUpdate();
            System.out.println(changed > 0 ? "Kitap silindi." : "Silme
başarısız (ID bulunamadı).");
        } catch (SQLException e) {
            System.out.println("[BookRepo] Silme hatası: " + e.getMessage());
        }
    }

    public Optional<Book> getById(int id) {
        String sql = "SELECT * FROM books WHERE id = ?";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return Optional.of(new Book(
                            rs.getInt("id"),
                            rs.getString("title"),
                            rs.getString("author"),
                            rs.getInt("year")
                    ));
                }
            }
        } catch (SQLException e) {
            System.out.println("[BookRepo] getById hatası: " +
e.getMessage());
        }
        return Optional.empty();
    }

    public ArrayList<Book> getAll() {
        ArrayList<Book> list = new ArrayList<>();
        String sql = "SELECT * FROM books";
        try (Connection conn = Database.connect(); Statement st =
conn.createStatement(); ResultSet rs = st.executeQuery(sql)) {
            while (rs.next()) {
                list.add(new Book(rs.getInt("id"), rs.getString("title"),
rs.getString("author"), rs.getInt("year")));
```

```
            }
        } catch (SQLException e) {
            System.out.println("[BookRepo] getAll hatası: " +
e.getMessage());
        }
        return list;
    }
}
```

---

src/repositories/StudentRepository.java

```
package repositories;

import database.Database;
import models.Student;

import java.sql.*;
import java.util.ArrayList;
import java.util.Optional;

public class StudentRepository extends BaseRepository {

    public void add(Student s) {
        String sql = "INSERT INTO students(name, department) VALUES(?,?)";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setString(1, s.getName());
            ps.setString(2, s.getDepartment());
            ps.executeUpdate();
            System.out.println("Öğrenci eklendi.");
        } catch (SQLException e) {
            System.out.println("[StudentRepo] Ekleme hatası: " +
e.getMessage());
        }
    }

    public void update(Student s) {
        String sql = "UPDATE students SET name = ?, department = ? WHERE id
= ?";
        try (Connection conn = Database.connect(); PreparedStatement ps =
conn.prepareStatement(sql)) {
            ps.setString(1, s.getName());
            ps.setString(2, s.getDepartmen
```