

Zadanie projektowe: Zaawansowany system zarządzania studentami

Cel

Opracowanie kompleksowego systemu zarządzania studentami (SMS) w języku Java, który pozwala użytkownikom na zarządzanie danymi studentów za pomocą graficznego interfejsu użytkownika (GUI) oraz łączy się z bazą danych w celu trwałego przechowywania informacji o studentach. System umożliwi operacje, takie jak dodawanie, usuwanie, aktualizowanie i wyświetlanie danych studentów, wykorzystując różne koncepcje programowania, które zostały omówione podczas kursu.

Wymagania projektowe

1. Struktura klas

- **Klasa `Student`:**
 - **Atrybuty:**
 - `name (String)`: Imię studenta.
 - `age (int)`: Wiek studenta (musi być liczbą całkowitą dodatnią).
 - `grade (double)`: Ocena studenta (musi być w zakresie od 0,0 do 100,0).
 - `studentID (String)`: Unikalny identyfikator studenta.
 - **Metody:**
 - Konstruktor do inicjalizacji atrybutów.
 - Metody gettery i settery dla każdego atrybutu.
 - Metoda `displayInfo()`, aby wydrukować szczegóły studenta.
- **Interfejs `StudentManager`:**
 - **Metody:**
 - `void addStudent(Student student)`: Dodaje nowego studenta do bazy danych.
 - `void removeStudent(String studentID)`: Usuwa studenta z bazy danych za pomocą unikalnego ID.
 - `void updateStudent(String studentID)`: Aktualizuje dane istniejącego studenta.
 - `ArrayList<Student> displayAllStudents()`: Pobiera i zwraca listę wszystkich studentów.
 - `double calculateAverageGrade()`: Oblicza i zwraca średnią ocen wszystkich studentów.
- **Klasa `StudentManagerImpl`:**
 - Implementuje interfejs `StudentManager`.
 - Zawiera metody do interakcji z bazą danych SQLite (lub inną relacyjną bazą danych) za pomocą JDBC.
 - Zapewnia, że wszystkie operacje na bazie danych są prawidłowo obsługiwane, w tym tworzenie niezbędnych tabel, jeśli nie istnieją.

2. Łączność z bazą danych

- Użyj **JDBC** (Java Database Connectivity), aby połączyć się z relacyjną bazą danych (zaleca się użycie SQLite dla uproszczenia).
- Utwórz tabelę bazy danych o nazwie `students` z następującymi kolumnami:

- name TEXT
 - age INTEGER
 - grade REAL
 - studentID TEXT PRIMARY KEY
- Wprowadź metody w klasie `StudentManagerImpl` do:
 - Nawiązania połączenia z bazą danych.
 - Wykonywania poleceń SQL dla operacji CRUD (tworzenie, odczyt, aktualizacja, usunięcie).
 - Prawidłowego zamykania połączenia z bazą danych, aby zapobiec wyciekom zasobów.

3. Graficzny interfejs użytkownika (GUI)

- Zaimplementuj **interfejs GUI** oparty na Swing, który pozwoli użytkownikom na efektywną interakcję z systemem zarządzania studentami.
- **Układ głównego okna:**
 - **Panel wejściowy:**
 - Etykiety i pola tekstowe dla:
 - Student ID
 - Name
 - Age
 - Grade
 - Upewnij się, że pola wejściowe walidują dane wprowadzone przez użytkownika (np. wiek musi być dodatni, ocena musi być w zakresie od 0,0 do 100,0).
 - **Przyciski:**
 - "Add Student": Wywołuje dodanie nowego studenta.
 - "Remove Student": Wywołuje usunięcie studenta za pomocą jego ID.
 - "Update Student": Umożliwia aktualizację szczegółów istniejącego studenta.
 - "Display All Students": Pobiera i wyświetla wszystkie rekordy studentów w obszarze tekstowym.
 - "Calculate Average": Oblicza i wyświetla średnią ocen wszystkich studentów.
 - **Panel wyjściowy:**
 - Obszar tekstowy wielowierszowy do wyświetlania:
 - Komunikatów potwierdzających (np. "Student added successfully").
 - Komunikatów o błędach (np. "Student ID not found").
 - Listy wszystkich studentów lub średniej ocen.
- **Obsługa zdarzeń:**
 - Zaimplementuj nasłuchiwanie akcji dla przycisków, aby obsługiwać interakcje użytkowników.
 - Upewnij się, że wszystkie działania użytkownika wywołują odpowiednie metody w klasie `StudentManagerImpl`.
 - Zapewnij zwrotną informację dla użytkownika przez panel wyjściowy dla każdej wykonanej akcji.

4. Obsługa wyjątków

- Wprowadź solidną obsługę wyjątków w całej aplikacji, aby zarządzać:

- Wyjątkami SQL podczas interakcji z bazą danych (np. błędy połączenia, błędy składni SQL).
- Błędami walidacji wejścia, zapewniając, że użytkownicy nie mogą wprowadzać niewłaściwych danych (np. wartości nienumeryczne dla wieku lub oceny).
- Błędami logicznymi, takimi jak próba aktualizacji lub usunięcia studenta, który nie istnieje w bazie danych.

Dodatkowe uwagi

- **Organizacja kodu:**
 - Upewnij się, że kod jest modularny, z wyraźnym podziałem na GUI, logikę biznesową i warstwę dostępu do danych.
- **Dokumentacja:**
 - Zawierać komentarze w kodzie, aby wyjaśnić kluczowe sekcje i logikę.
 - Utwórz plik README zawierający:
 - Instrukcje dotyczące kompilacji i uruchamiania aplikacji.
 - Przegląd funkcjonalności oferowanej przez system zarządzania studentami.
 - Instrukcje dotyczące konfiguracji bazy danych (np. polecenia SQL do stworzenia tabeli `students`, jeśli to konieczne).

Ostateczne rezultaty

- W pełni funkcjonalna aplikacja Java, która spełnia określone wymagania.
- Kod źródłowy z wyraźną organizacją i dokumentacją.
- Działająca baza danych SQLite (lub inna wybrana baza danych) z rekordami studentów.