**TASK 5**

**Course Database: Store course information, including course code, title,**

**description, capacity, and schedule.**

**Student Database: Store student information, including student ID, name, and**

**registered courses.**

**Course Listing: Display available courses with details and available slots.**

**Student Registration: Allow students to register for courses from the available**

**options.**

**Course Removal: Enable students to drop courses they have registered for.**

CODE:

import java.util.ArrayList;

import java.util.Scanner;

class Course {

String courseCode;

String title;

String description;

int capacity;

String schedule;

Course(String courseCode, String title, String description, int capacity, String schedule) {

this.courseCode = courseCode;

this.title = title;

this.description = description;

this.capacity = capacity;

this.schedule = schedule;

}

}

class Student {

int studentID;

String name;

ArrayList<Course> registeredCourses = new ArrayList<>();

Student(int studentID, String name) {

this.studentID = studentID;

this.name = name;

}

}

public class CourseRegistrationSystem {

static ArrayList<Course> courseDatabase = new ArrayList<>();

static ArrayList<Student> studentDatabase = new ArrayList<>();

static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

initializeCourses();

while (true) {

displayMenu();

int choice = scanner.nextInt();

switch (choice) {

case 1:

displayCourseListing();

break;

case 2:

registerStudent();

break;

case 3:

removeCourse();

break;

case 4:

System.out.println("Exiting the program. Goodbye!");

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

static void initializeCourses() {

Course course1 = new Course("CSC101", "Introduction to Programming", "Basic programming concepts", 30, "MWF 10:00 AM");

Course course2 = new Course("MAT201", "Calculus I", "Limits, derivatives, integrals", 25, "TTH 2:00 PM");

Course course3 = new Course("ENG301", "English Literature", "Classic literature analysis", 20, "MWF 1:00 PM");

courseDatabase.add(course1);

courseDatabase.add(course2);

courseDatabase.add(course3);

}

static void displayMenu() {

System.out.println("\nCourse Registration System Menu:");

System.out.println("1. Display Course Listing");

System.out.println("2. Register for a Course");

System.out.println("3. Remove a Course");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

}

static void displayCourseListing() {

System.out.println("\nAvailable Courses:");

for (Course course : courseDatabase) {

System.out.println("Course Code: " + course.courseCode);

System.out.println("Title: " + course.title);

System.out.println("Description: " + course.description);

System.out.println("Capacity: " + course.capacity + " | Schedule: " + course.schedule);

System.out.println("------------------------------");

}

}

static void registerStudent() {

System.out.print("Enter student ID: ");

int studentID = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

System.out.print("Enter student name: ");

String studentName = scanner.nextLine();

Student student = new Student(studentID, studentName);

displayCourseListing();

System.out.print("Enter the course code to register: ");

String courseCode = scanner.nextLine();

Course selectedCourse = findCourse(courseCode);

if (selectedCourse != null && selectedCourse.capacity > 0) {

student.registeredCourses.add(selectedCourse);

selectedCourse.capacity--;

studentDatabase.add(student);

System.out.println("Registration successful!");

} else if (selectedCourse == null) {

System.out.println("Invalid course code. Please try again.");

} else {

System.out.println("Course is full. Cannot register for this course.");

}

}

static void removeCourse() {

System.out.print("Enter student ID for course removal: ");

int studentID = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

System.out.print("Enter the course code to remove: ");

String courseCode = scanner.nextLine();

Student student = findStudent(studentID);

Course selectedCourse = findCourse(courseCode);

if (student != null && selectedCourse != null && student.registeredCourses.contains(selectedCourse)) {

student.registeredCourses.remove(selectedCourse);

selectedCourse.capacity++;

System.out.println("Course removal successful!");

} else if (student == null) {

System.out.println("Student not found. Please try again.");

} else if (selectedCourse == null) {

System.out.println("Invalid course code. Please try again.");

} else {

System.out.println("Student is not registered for this course.");

}

}

static Course findCourse(String courseCode) {

for (Course course : courseDatabase) {

if (course.courseCode.equals(courseCode)) {

return course;

}

}

return null;

}

static Student findStudent(int studentID) {

for (Student student : studentDatabase) {

if (student.studentID == studentID) {

return student;

}

}

return null;

}

}