import java.util.\*;

class Course {

private String code;

private String title;

private String description;

private int capacity;

private String schedule;

private int enrolled;

public Course(String code, String title, String description, int capacity, String schedule) {

this.code = code;

this.title = title;

this.description = description;

this.capacity = capacity;

this.schedule = schedule;

this.enrolled = 0;

}

public String getCode() {

return code;

}

public String getTitle() {

return title;

}

public String getDescription() {

return description;

}

public int getCapacity() {

return capacity;

}

public String getSchedule() {

return schedule;

}

public int getAvailableSlots() {

return capacity - enrolled;

}

public boolean register() {

if (enrolled < capacity) {

enrolled++;

return true;

}

return false;

}

public boolean unregister() {

if (enrolled > 0) {

enrolled--;

return true;

}

return false;

}

}

class Student {

private String studentID;

private String name;

private List<Course> registeredCourses;

public Student(String studentID, String name) {

this.studentID = studentID.toLowerCase();

this.name = name;

this.registeredCourses = new ArrayList<>();

}

public String getStudentID() {

return studentID;

}

public String getName() {

return name;

}

public List<Course> getRegisteredCourses() {

return registeredCourses;

}

public boolean registerCourse(Course course) {

if (registeredCourses.contains(course) || !course.register()) {

return false;

}

registeredCourses.add(course);

return true;

}

public boolean dropCourse(Course course) {

if (registeredCourses.remove(course)) {

course.unregister();

return true;

}

return false;

}

}

public class CourseDatabaseApp {

private static Map<String, Course> courses = new HashMap<>();

private static Map<String, Student> students = new HashMap<>();

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

public static void main(String[] args) {

initializeCourses();

while (true) {

System.out.println("\n1. Display Courses");

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

System.out.println("3. Drop Course for Student");

System.out.println("4. Display Student Information");

System.out.println("5. Exit");

System.out.print("Choose an option: ");

int choice = getValidIntInput();

switch (choice) {

case 1:

displayCourses();

break;

case 2:

registerStudentForCourse();

break;

case 3:

dropStudentCourse();

break;

case 4:

displayStudentInfo();

break;

case 5:

System.out.println("Goodbye!");

return;

default:

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

}

}

}

private static void initializeCourses() {

courses.put("CS101", new Course("CS101", "Intro to Computer Science", "Basics of computer science", 30, "MWF 10-11AM"));

courses.put("MATH101", new Course("MATH101", "Calculus I", "Introduction to calculus", 25, "TTh 9-10:30AM"));

courses.put("ENG101", new Course("ENG101", "English Literature", "Overview of English literature", 20, "MW 1-2:30PM"));

}

private static void displayCourses() {

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

for (Course course : courses.values()) {

System.out.printf("Code: %s, Title: %s, Description: %s, Capacity: %d, Schedule: %s, Available Slots: %d%n",

course.getCode(), course.getTitle(), course.getDescription(), course.getCapacity(), course.getSchedule(), course.getAvailableSlots());

}

}

private static void registerStudentForCourse() {

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

String studentID = scanner.nextLine().trim().toLowerCase();

Student student = students.get(studentID);

if (student == null) {

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

String name = scanner.nextLine().trim();

student = new Student(studentID, name);

students.put(studentID, student);

}

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

String courseCode = scanner.nextLine().trim().toUpperCase();

Course course = courses.get(courseCode);

if (course == null) {

System.out.println("Course not found.");

return;

}

if (student.registerCourse(course)) {

System.out.println("Successfully registered for the course.");

} else {

System.out.println("Failed to register for the course. It might be full or you are already registered.");

}

}

private static void dropStudentCourse() {

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

String studentID = scanner.nextLine().trim().toLowerCase();

Student student = students.get(studentID);

if (student == null) {

System.out.println("Student not found.");

return;

}

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

String courseCode = scanner.nextLine().trim().toUpperCase();

Course course = courses.get(courseCode);

if (course == null) {

System.out.println("Course not found.");

return;

}

if (student.dropCourse(course)) {

System.out.println("Successfully dropped the course.");

} else {

System.out.println("Failed to drop the course. You might not be registered for it.");

}

}

private static void displayStudentInfo() {

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

String studentID = scanner.nextLine().trim().toLowerCase();

Student student = students.get(studentID);

if (student == null) {

System.out.println("Student not found.");

return;

}

System.out.printf("Student ID: %s, Name: %s%n", student.getStudentID(), student.getName());

System.out.println("Registered Courses:");

for (Course course : student.getRegisteredCourses()) {

System.out.printf("Code: %s, Title: %s, Schedule: %s%n", course.getCode(), course.getTitle(), course.getSchedule());

}

}

private static int getValidIntInput() {

while (true) {

try {

int input = Integer.parseInt(scanner.nextLine().trim());

return input;

} catch (NumberFormatException e) {

System.out.print("Invalid input. Please enter a number: ");

}

}

}

}