**Day 5 - Waitlist**

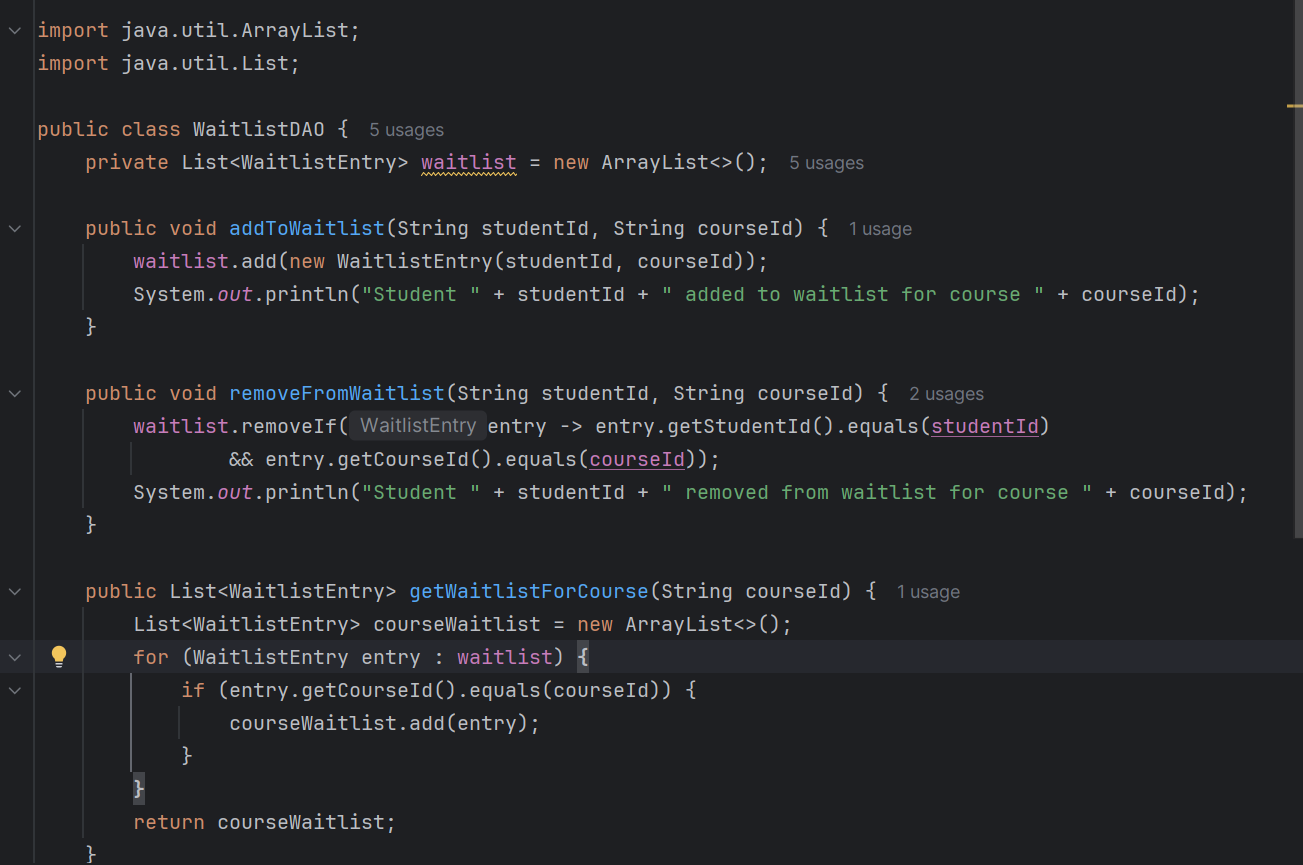
**WaitlistEntry.java**

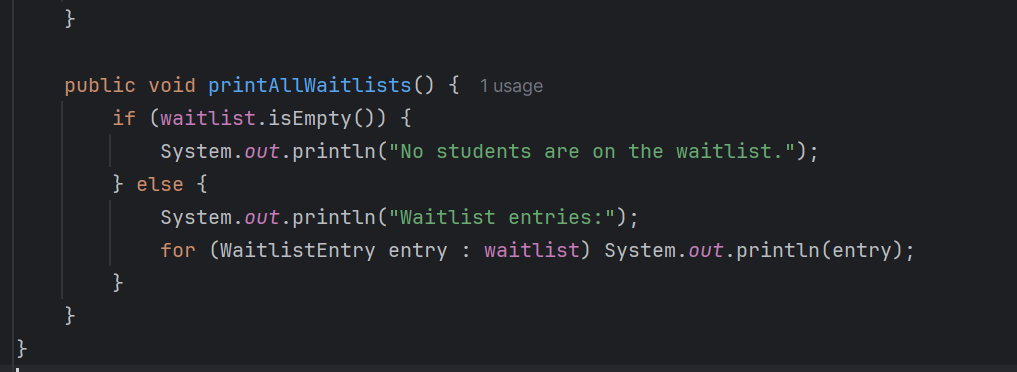
public class WaitlistEntry {  
 private String studentId;  
 private String courseId;  
  
 public WaitlistEntry(String studentId, String courseId) {  
 this.studentId = studentId;  
 this.courseId = courseId;  
 }  
  
 public String getStudentId() { return studentId; }  
 public String getCourseId() { return courseId; }  
  
 @Override  
 public String toString() {  
 return "WaitlistEntry{" + "studentId='" + studentId + '\'' +  
 ", courseId='" + courseId + '\'' + '}';  
 }  
}



**WaitlistDAO.java**

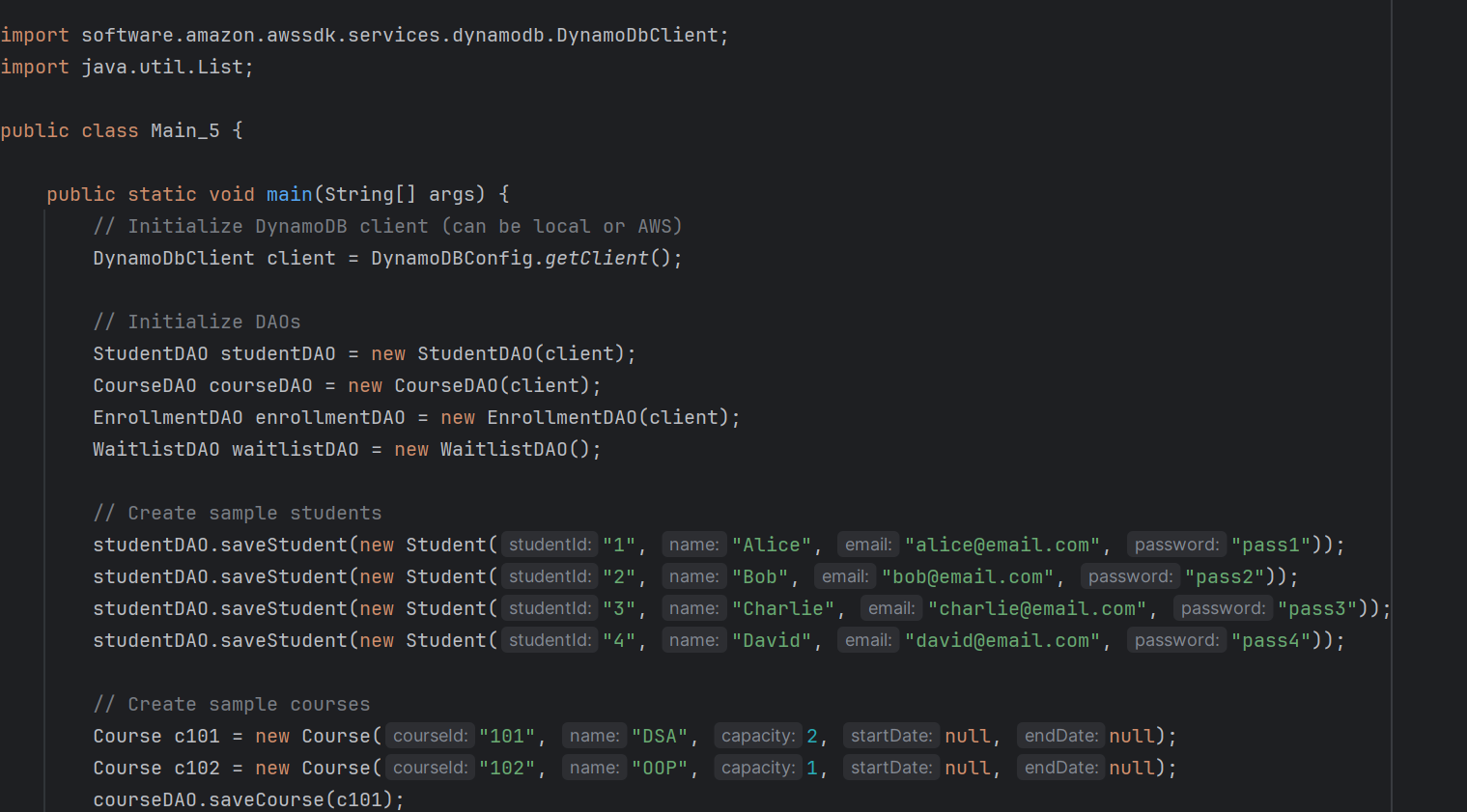
import java.util.ArrayList;  
import java.util.List;  
  
public class WaitlistDAO {  
 private List<WaitlistEntry> waitlist = new ArrayList<>();  
  
 public void addToWaitlist(String studentId, String courseId) {  
 waitlist.add(new WaitlistEntry(studentId, courseId));  
 System.*out*.println("Student " + studentId + " added to waitlist for course " + courseId);  
 }  
  
 public void removeFromWaitlist(String studentId, String courseId) {  
 waitlist.removeIf(entry -> entry.getStudentId().equals(studentId)  
 && entry.getCourseId().equals(courseId));  
 System.*out*.println("Student " + studentId + " removed from waitlist for course " + courseId);  
 }  
  
 public List<WaitlistEntry> getWaitlistForCourse(String courseId) {  
 List<WaitlistEntry> courseWaitlist = new ArrayList<>();  
 for (WaitlistEntry entry : waitlist) {  
 if (entry.getCourseId().equals(courseId)) {  
 courseWaitlist.add(entry);  
 }  
 }  
 return courseWaitlist;  
 }  
  
 public void printAllWaitlists() {  
 if (waitlist.isEmpty()) {  
 System.*out*.println("No students are on the waitlist.");  
 } else {  
 System.*out*.println("Waitlist entries:");  
 for (WaitlistEntry entry : waitlist) System.*out*.println(entry);  
 }  
 }  
}

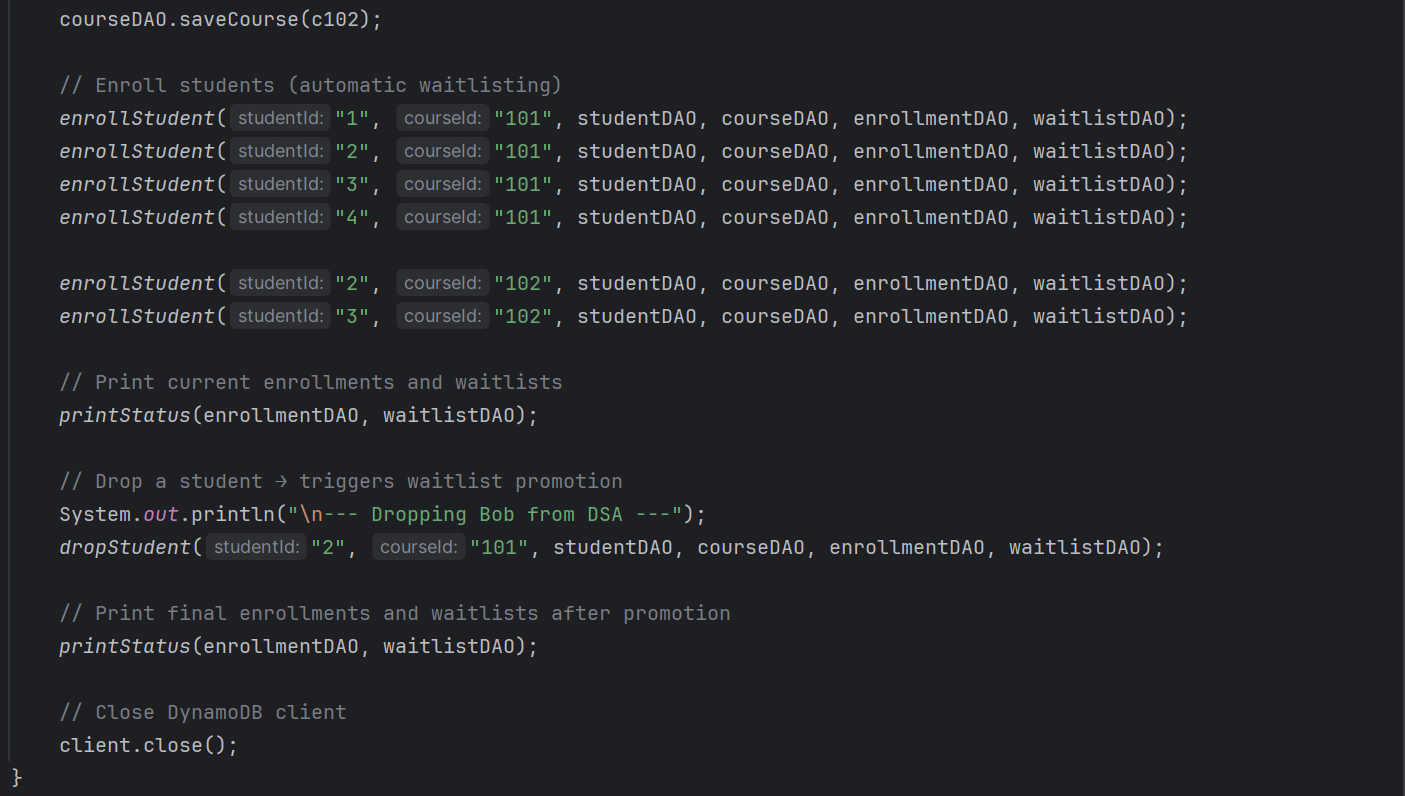


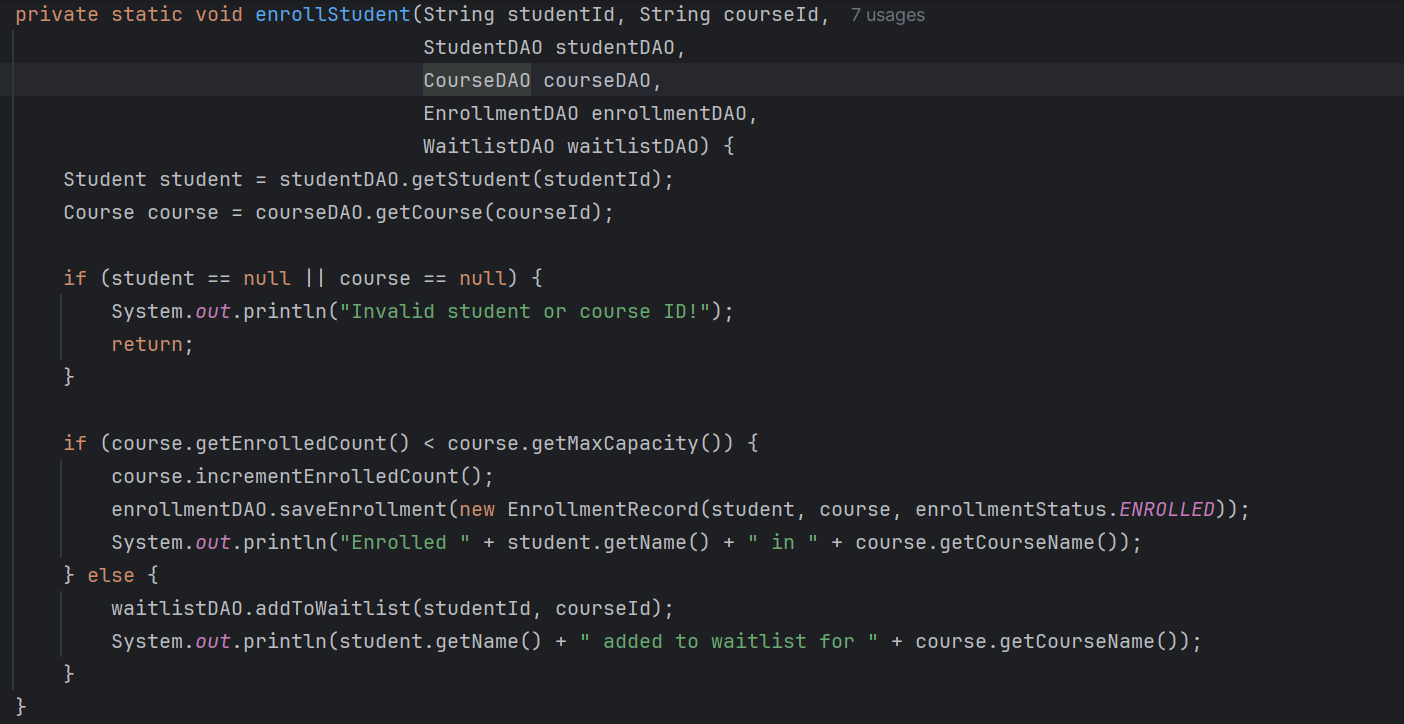


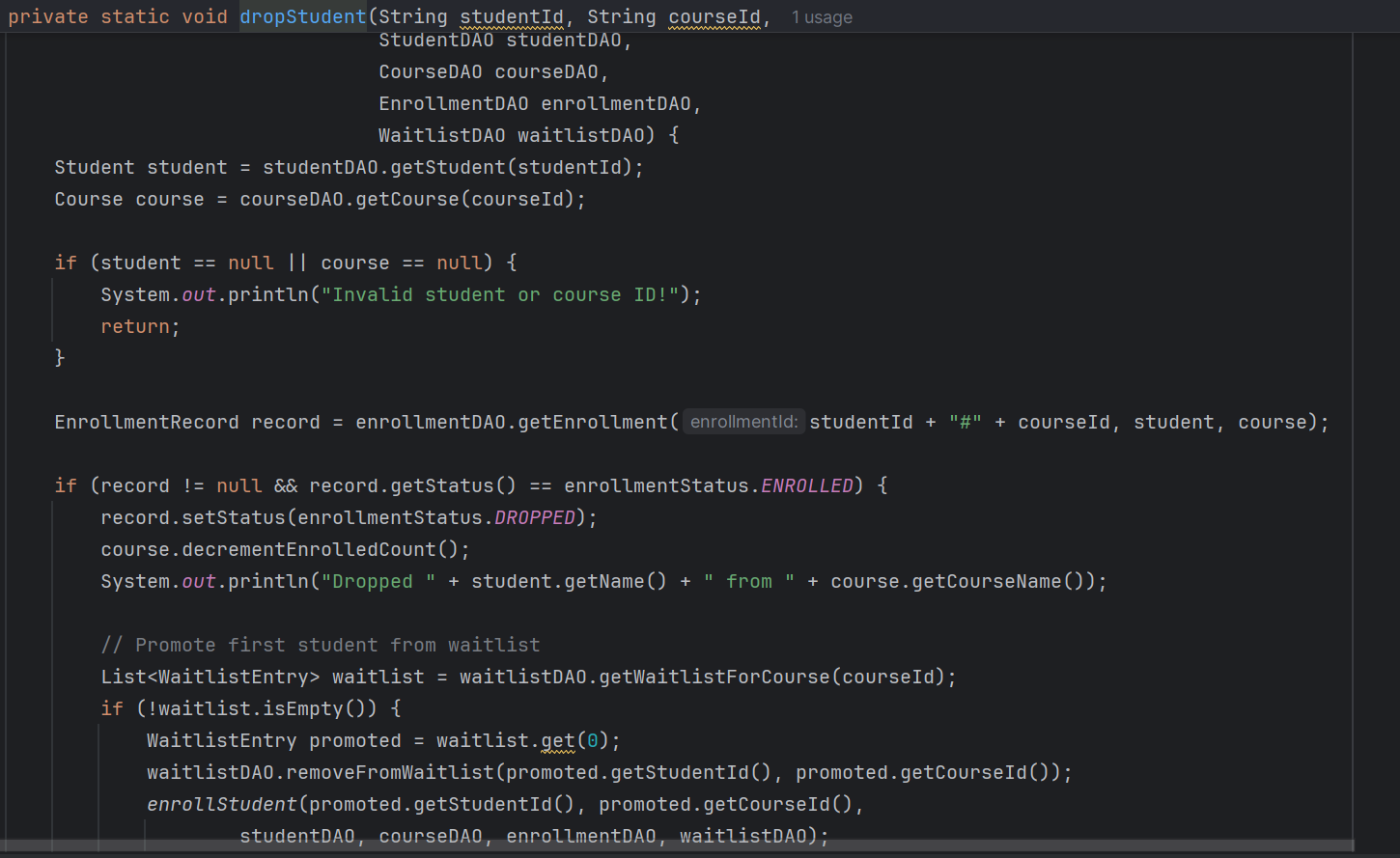
**Main\_5.java**

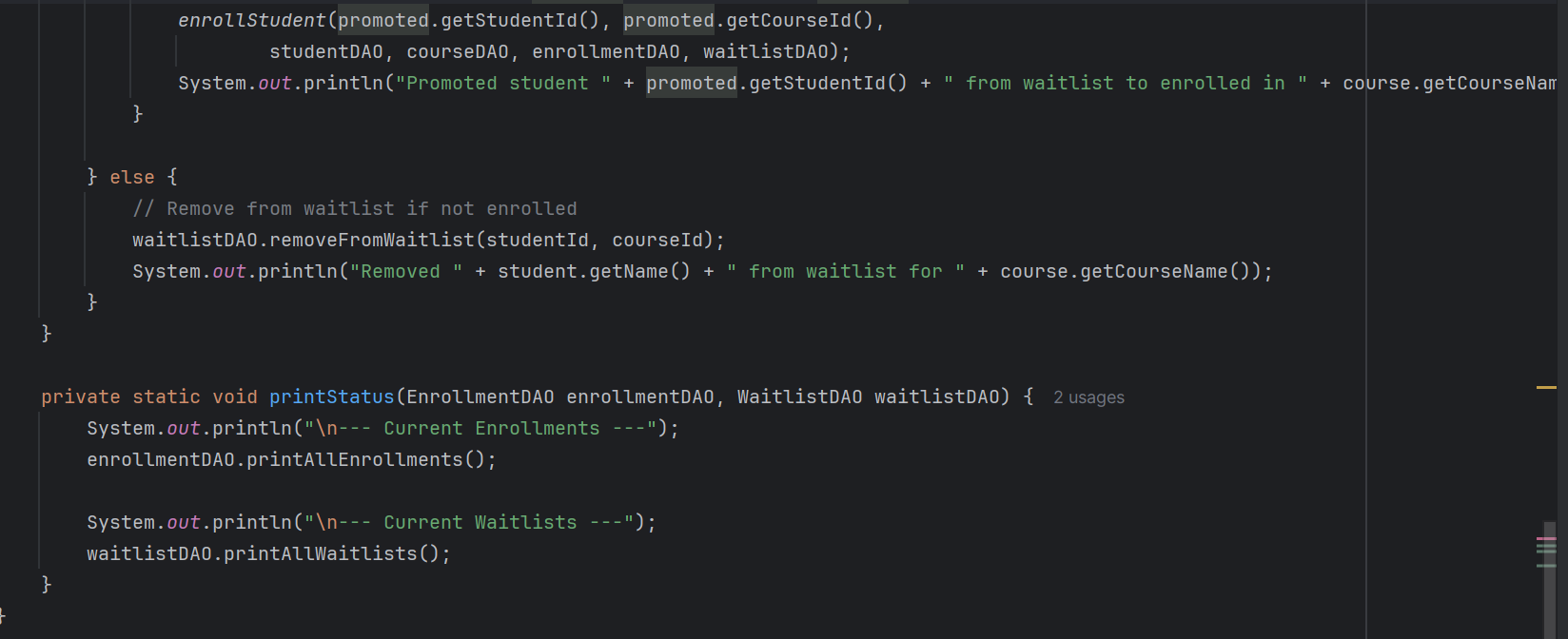
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
 import java.util.List;  
  
public class Main\_5 {  
  
 public static void main(String[] args) {  
 // Initialize DynamoDB client (can be local or AWS)  
 DynamoDbClient client = DynamoDBConfig.*getClient*();  
  
 // Initialize DAOs  
 StudentDAO studentDAO = new StudentDAO(client);  
 CourseDAO courseDAO = new CourseDAO(client);  
 EnrollmentDAO enrollmentDAO = new EnrollmentDAO(client);  
 WaitlistDAO waitlistDAO = new WaitlistDAO();  
  
 // Create sample students  
 studentDAO.saveStudent(new Student("1", "Alice", "alice@email.com", "pass1"));  
 studentDAO.saveStudent(new Student("2", "Bob", "bob@email.com", "pass2"));  
 studentDAO.saveStudent(new Student("3", "Charlie", "charlie@email.com", "pass3"));  
 studentDAO.saveStudent(new Student("4", "David", "david@email.com", "pass4"));  
  
 // Create sample courses  
 Course c101 = new Course("101", "DSA", 2, null, null);  
 Course c102 = new Course("102", "OOP", 1, null, null);  
 courseDAO.saveCourse(c101);  
 courseDAO.saveCourse(c102);  
  
 // Enroll students (automatic waitlisting)  
 *enrollStudent*("1", "101", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
 *enrollStudent*("2", "101", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
 *enrollStudent*("3", "101", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
 *enrollStudent*("4", "101", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
  
 *enrollStudent*("2", "102", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
 *enrollStudent*("3", "102", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
  
 // Print current enrollments and waitlists  
 *printStatus*(enrollmentDAO, waitlistDAO);  
  
 // Drop a student → triggers waitlist promotion  
 System.*out*.println("\n--- Dropping Bob from DSA ---");  
 *dropStudent*("2", "101", studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
  
 // Print final enrollments and waitlists after promotion  
 *printStatus*(enrollmentDAO, waitlistDAO);  
  
 // Close DynamoDB client  
 client.close();  
 }  
  
 private static void enrollStudent(String studentId, String courseId,  
 StudentDAO studentDAO,  
 CourseDAO courseDAO,  
 EnrollmentDAO enrollmentDAO,  
 WaitlistDAO waitlistDAO) {  
 Student student = studentDAO.getStudent(studentId);  
 Course course = courseDAO.getCourse(courseId);  
  
 if (student == null || course == null) {  
 System.*out*.println("Invalid student or course ID!");  
 return;  
 }  
  
 if (course.getEnrolledCount() < course.getMaxCapacity()) {  
 course.incrementEnrolledCount();  
 enrollmentDAO.saveEnrollment(new EnrollmentRecord(student, course, enrollmentStatus.*ENROLLED*));  
 System.*out*.println("Enrolled " + student.getName() + " in " + course.getCourseName());  
 } else {  
 waitlistDAO.addToWaitlist(studentId, courseId);  
 System.*out*.println(student.getName() + " added to waitlist for " + course.getCourseName());  
 }  
 }  
  
 private static void dropStudent(String studentId, String courseId,  
 StudentDAO studentDAO,  
 CourseDAO courseDAO,  
 EnrollmentDAO enrollmentDAO,  
 WaitlistDAO waitlistDAO) {  
 Student student = studentDAO.getStudent(studentId);  
 Course course = courseDAO.getCourse(courseId);  
  
 if (student == null || course == null) {  
 System.*out*.println("Invalid student or course ID!");  
 return;  
 }  
  
 EnrollmentRecord record = enrollmentDAO.getEnrollment(studentId + "#" + courseId, student, course);  
  
 if (record != null && record.getStatus() == enrollmentStatus.*ENROLLED*) {  
 record.setStatus(enrollmentStatus.*DROPPED*);  
 course.decrementEnrolledCount();  
 System.*out*.println("Dropped " + student.getName() + " from " + course.getCourseName());  
  
 // Promote first student from waitlist  
 List<WaitlistEntry> waitlist = waitlistDAO.getWaitlistForCourse(courseId);  
 if (!waitlist.isEmpty()) {  
 WaitlistEntry promoted = waitlist.get(0);  
 waitlistDAO.removeFromWaitlist(promoted.getStudentId(), promoted.getCourseId());  
 *enrollStudent*(promoted.getStudentId(), promoted.getCourseId(),  
 studentDAO, courseDAO, enrollmentDAO, waitlistDAO);  
 System.*out*.println("Promoted student " + promoted.getStudentId() + " from waitlist to enrolled in " + course.getCourseName());  
 }  
  
 } else {  
 // Remove from waitlist if not enrolled  
 waitlistDAO.removeFromWaitlist(studentId, courseId);  
 System.*out*.println("Removed " + student.getName() + " from waitlist for " + course.getCourseName());  
 }  
 }  
  
 private static void printStatus(EnrollmentDAO enrollmentDAO, WaitlistDAO waitlistDAO) {  
 System.*out*.println("\n--- Current Enrollments ---");  
 enrollmentDAO.printAllEnrollments();  
  
 System.*out*.println("\n--- Current Waitlists ---");  
 waitlistDAO.printAllWaitlists();  
 }  
}











**WaitlistDAO / WaitlistEntry**

* Stores a list of students waiting for courses.
* Key methods:
  + addToWaitlist(studentId, courseId)
  + removeFromWaitlist(studentId, courseId)
  + getWaitlistForCourse(courseId)
  + printAllWaitlists()

**Enrollment Logic**

* enrollStudent():
  + Checks if course has available seats:
    - ✅ If yes → ENROLLED
    - ⚠ If full → added to waitlist
* dropStudent():
  + If ENROLLED → sets status to DROPPED, decrements seat
  + Promotes first student from waitlist (FIFO) automatically
  + If WAITLISTED → removes from waitlist

**Main\_5 Workflow**

1. Initialize DAOs (students, courses, enrollment, waitlist)
2. Create students and courses
3. Enroll students → automatically waitlist if full
4. Print **current enrollments and waitlists**
5. Drop a student → triggers **waitlist promotion**
6. Print **final enrollments and waitlists**

**Key Features**

* Waitlist ensures **first-come, first-served promotion**
* Supports **mixed status**: ENROLLED / WAITLISTED / DROPPED
* Integrates with **DynamoDB** for student/course persistence