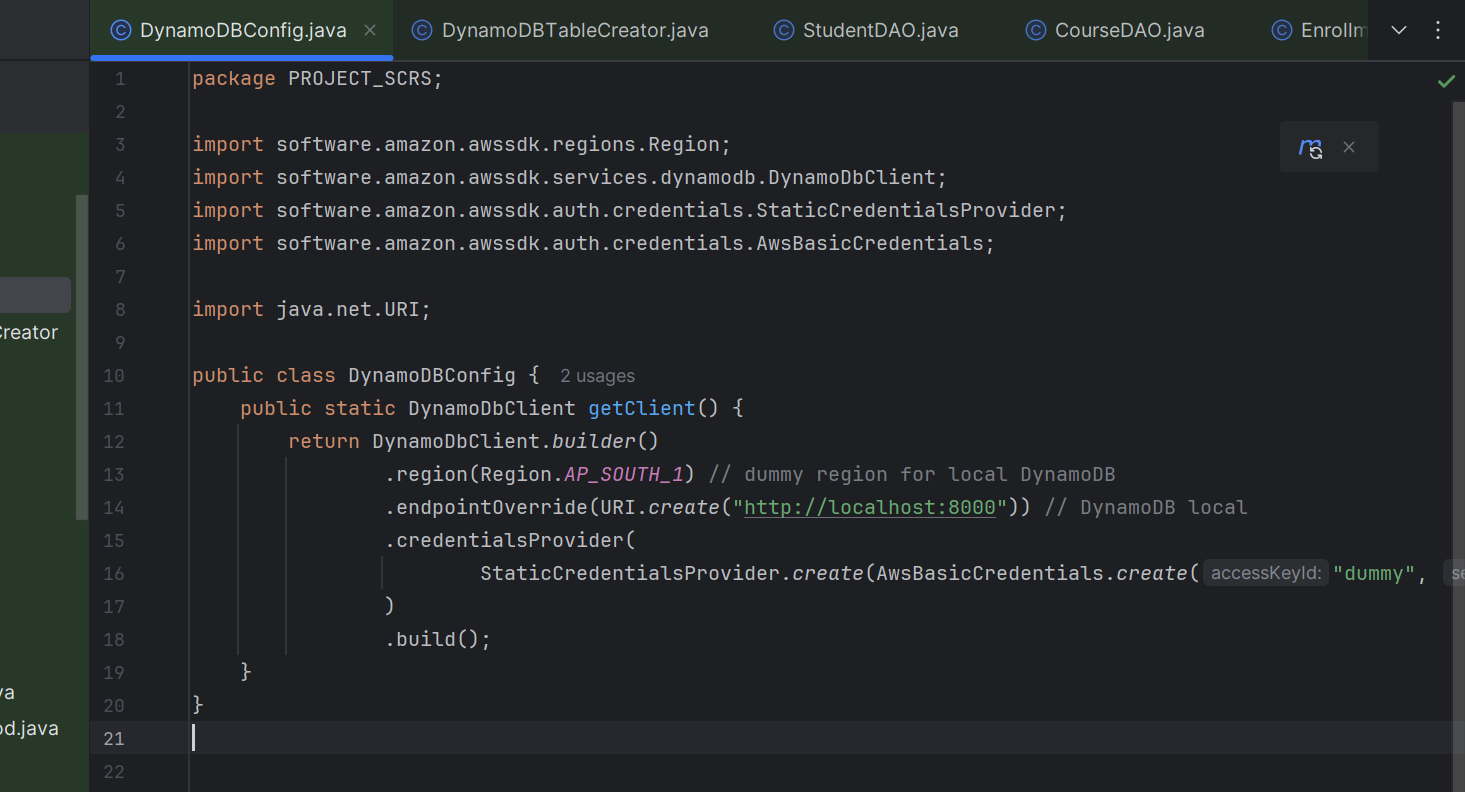
## **Day 4 - Check Seat Availability**

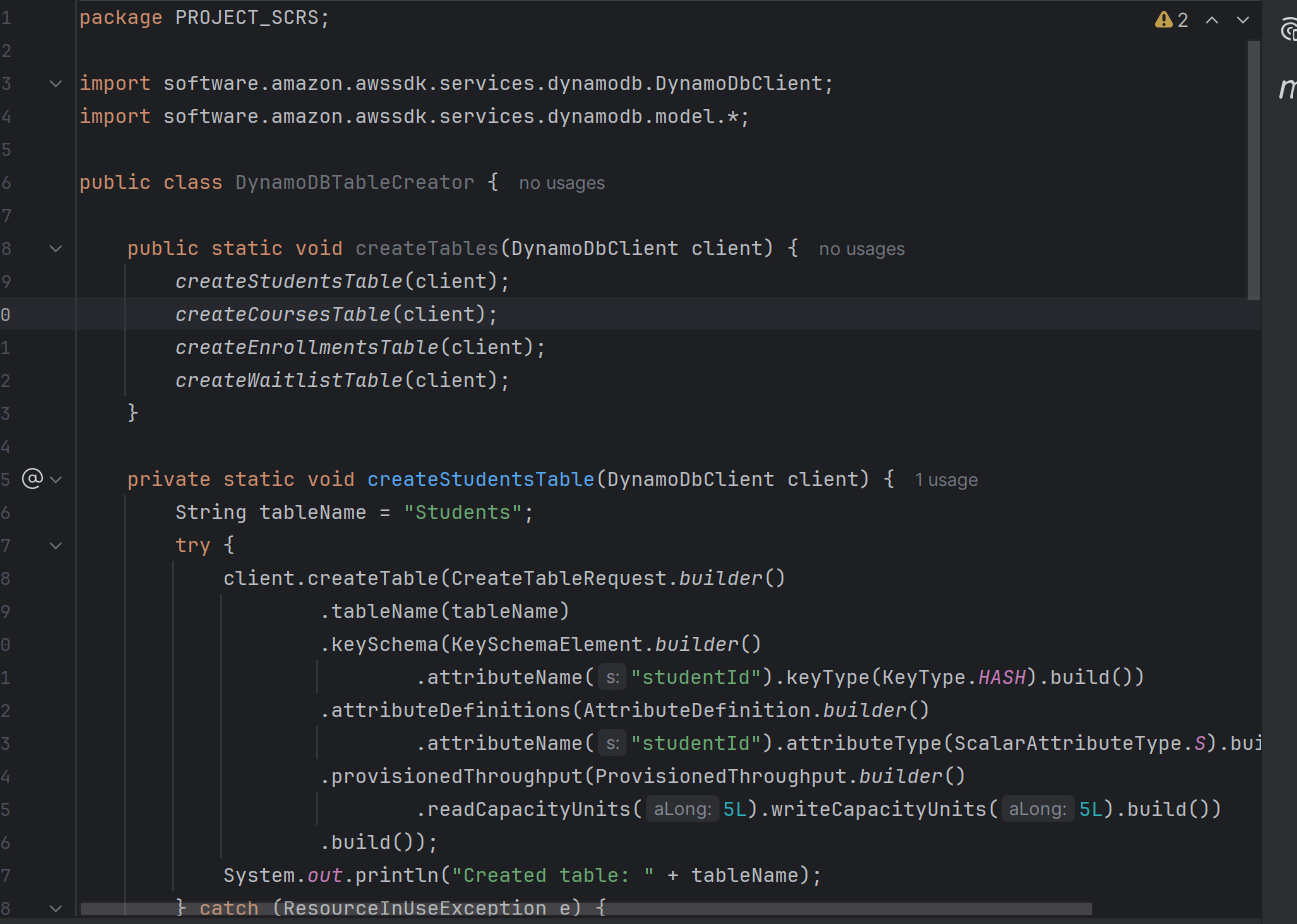
**DynamoDBConfig.Java**

import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;  
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;  
  
import java.net.URI;  
  
public class DynamoDBConfig {  
 public static DynamoDbClient getClient() {  
 return DynamoDbClient.*builder*()  
 .region(Region.*AP\_SOUTH\_1*) // dummy region for local DynamoDB  
 .endpointOverride(URI.*create*("http://localhost:8000")) // DynamoDB local  
 .credentialsProvider(  
 StaticCredentialsProvider.*create*(AwsBasicCredentials.*create*("dummy", "dummy"))  
 )  
 .build();  
 }  
}



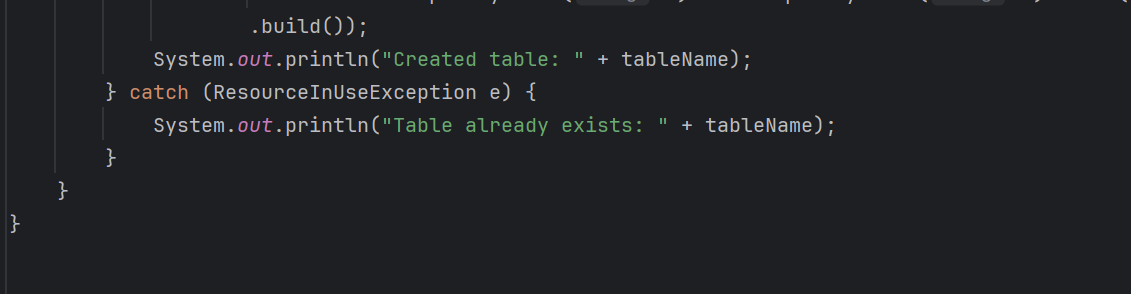
**DynamoDBTableCreator.java**

package PROJECT\_SCRS;  
  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.\*;  
  
public class DynamoDBTableCreator {  
  
 public static void createTables(DynamoDbClient client) {  
 *createStudentsTable*(client);  
 *createCoursesTable*(client);  
 *createEnrollmentsTable*(client);  
 *createWaitlistTable*(client);  
 }  
  
 private static void createStudentsTable(DynamoDbClient client) {  
 String tableName = "Students";  
 try {  
 client.createTable(CreateTableRequest.*builder*()  
 .tableName(tableName)  
 .keySchema(KeySchemaElement.*builder*()  
 .attributeName("studentId").keyType(KeyType.*HASH*).build())  
 .attributeDefinitions(AttributeDefinition.*builder*()  
 .attributeName("studentId").attributeType(ScalarAttributeType.*S*).build())  
 .provisionedThroughput(ProvisionedThroughput.*builder*()  
 .readCapacityUnits(5L).writeCapacityUnits(5L).build())  
 .build());  
 System.*out*.println("Created table: " + tableName);  
 } catch (ResourceInUseException e) {  
 System.*out*.println("Table already exists: " + tableName);  
 }  
 }  
  
 private static void createCoursesTable(DynamoDbClient client) {  
 String tableName = "Courses";  
 try {  
 client.createTable(CreateTableRequest.*builder*()  
 .tableName(tableName)  
 .keySchema(KeySchemaElement.*builder*()  
 .attributeName("courseId").keyType(KeyType.*HASH*).build())  
 .attributeDefinitions(AttributeDefinition.*builder*()  
 .attributeName("courseId").attributeType(ScalarAttributeType.*S*).build())  
 .provisionedThroughput(ProvisionedThroughput.*builder*()  
 .readCapacityUnits(5L).writeCapacityUnits(5L).build())  
 .build());  
 System.*out*.println("Created table: " + tableName);  
 } catch (ResourceInUseException e) {  
 System.*out*.println("Table already exists: " + tableName);  
 }  
 }  
  
 private static void createEnrollmentsTable(DynamoDbClient client) {  
 String tableName = "Enrollments";  
 try {  
 client.createTable(CreateTableRequest.*builder*()  
 .tableName(tableName)  
 .keySchema(KeySchemaElement.*builder*()  
 .attributeName("enrollmentId").keyType(KeyType.*HASH*).build())  
 .attributeDefinitions(AttributeDefinition.*builder*()  
 .attributeName("enrollmentId").attributeType(ScalarAttributeType.*S*).build())  
 .provisionedThroughput(ProvisionedThroughput.*builder*()  
 .readCapacityUnits(5L).writeCapacityUnits(5L).build())  
 .build());  
 System.*out*.println("Created table: " + tableName);  
 } catch (ResourceInUseException e) {  
 System.*out*.println("Table already exists: " + tableName);  
 }  
 }  
  
 private static void createWaitlistTable(DynamoDbClient client) {  
 String tableName = "Waitlist";  
 try {  
 client.createTable(CreateTableRequest.*builder*()  
 .tableName(tableName)  
 .keySchema(KeySchemaElement.*builder*()  
 .attributeName("waitlistId").keyType(KeyType.*HASH*).build())  
 .attributeDefinitions(AttributeDefinition.*builder*()  
 .attributeName("waitlistId").attributeType(ScalarAttributeType.*S*).build())  
 .provisionedThroughput(ProvisionedThroughput.*builder*()  
 .readCapacityUnits(5L).writeCapacityUnits(5L).build())  
 .build());  
 System.*out*.println("Created table: " + tableName);  
 } catch (ResourceInUseException e) {  
 System.*out*.println("Table already exists: " + tableName);  
 }  
 }  
}



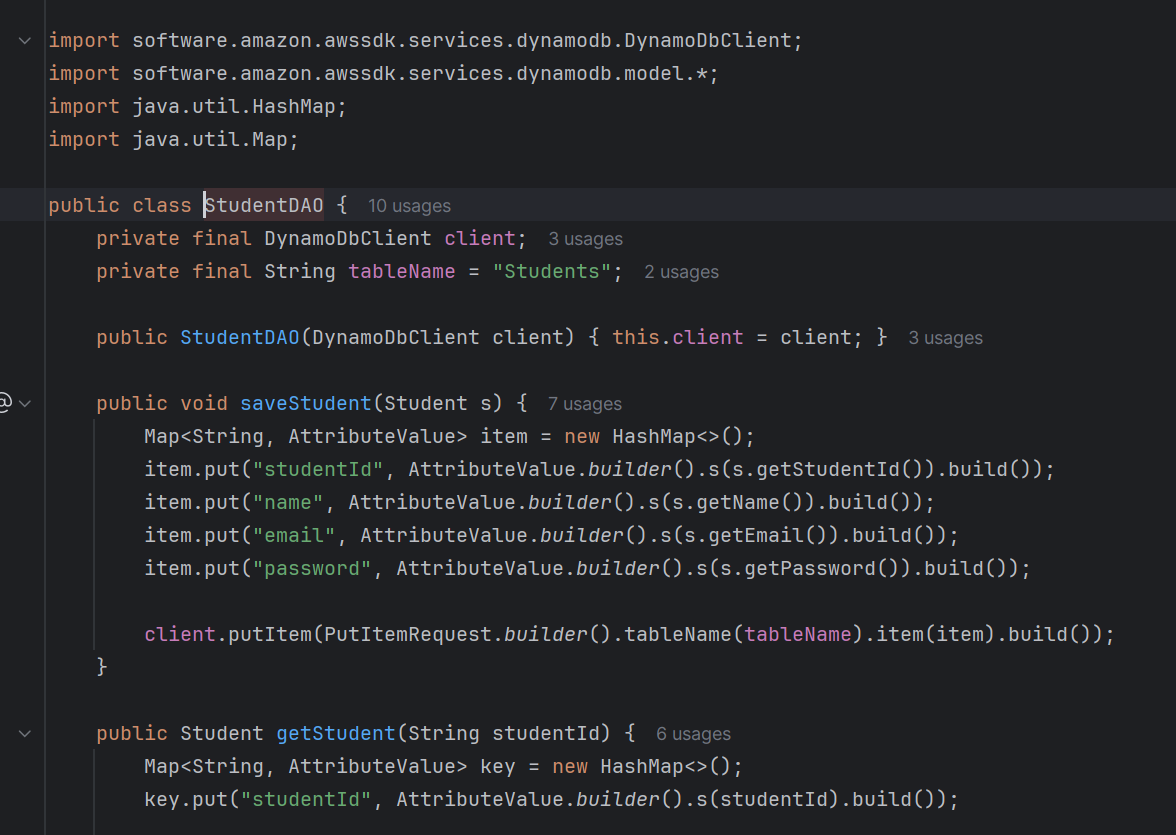


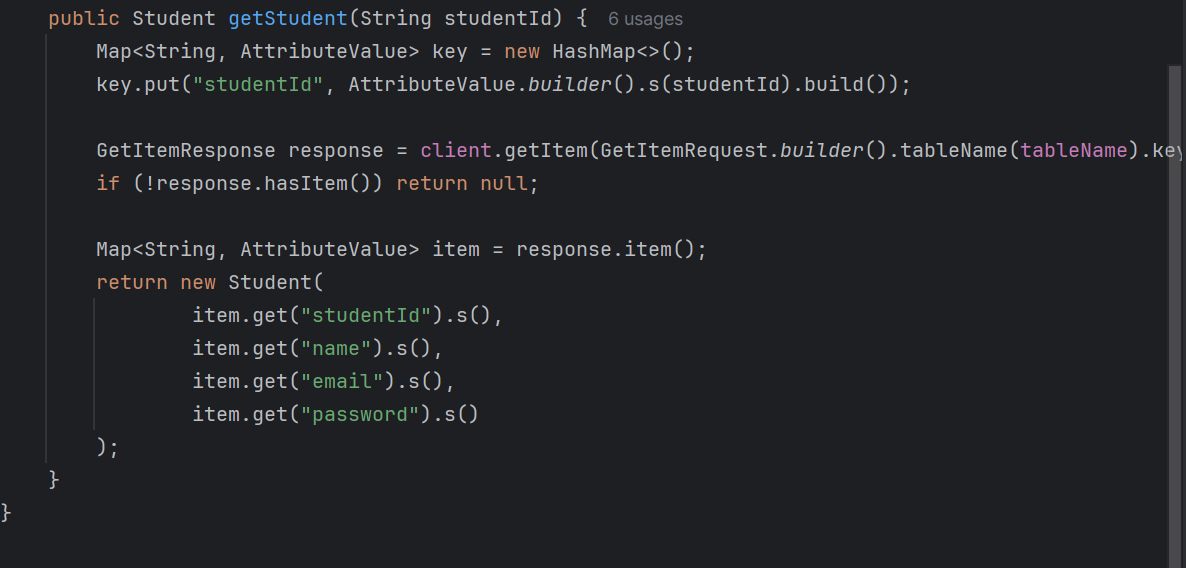




**StudentDA0.java**

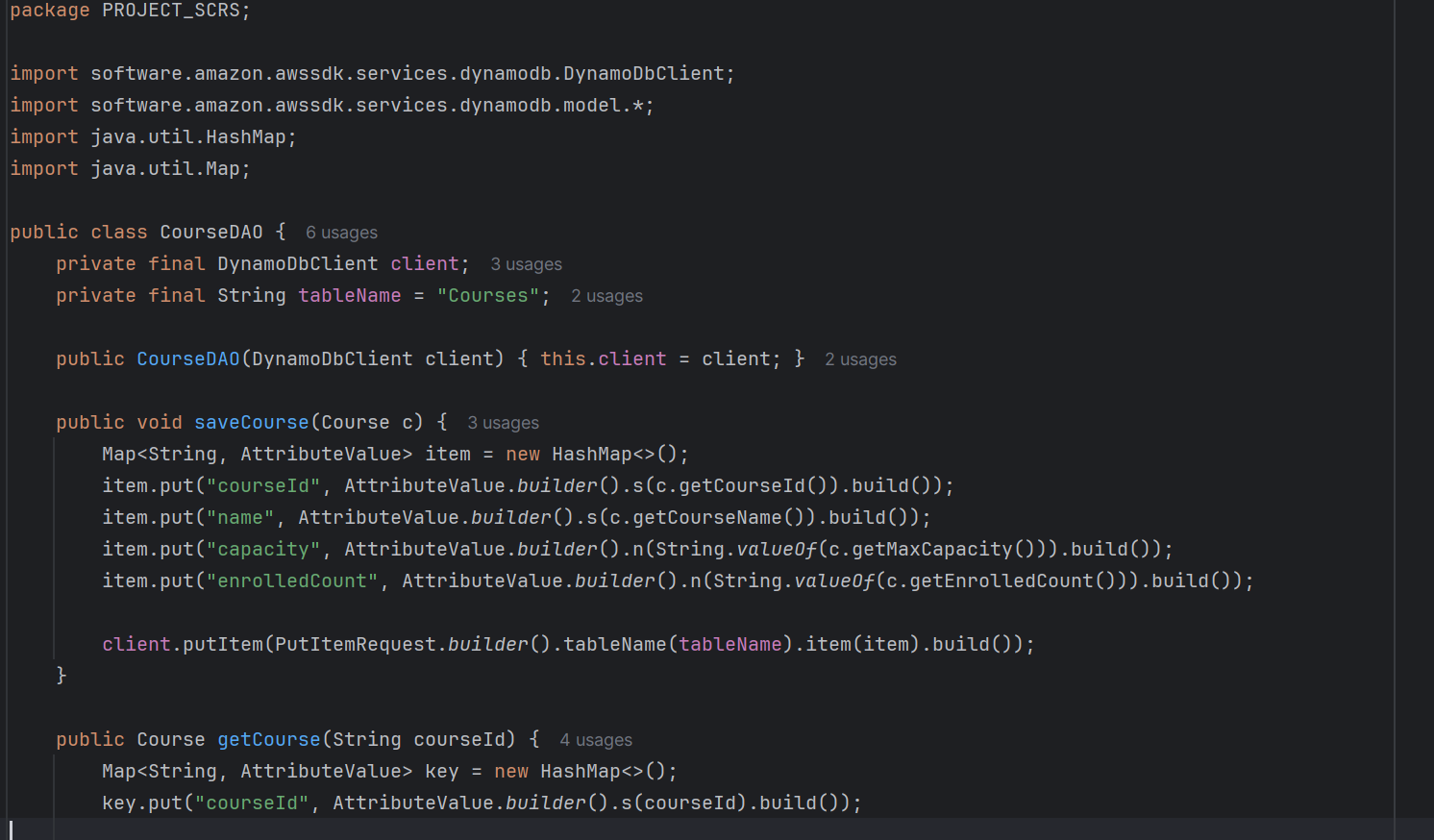
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
public class StudentDAO {  
 private final DynamoDbClient client;  
 private final String tableName = "Students";  
  
 public StudentDAO(DynamoDbClient client) { this.client = client; }  
  
 public void saveStudent(Student s) {  
 Map<String, AttributeValue> item = new HashMap<>();  
 item.put("studentId", AttributeValue.*builder*().s(s.getStudentId()).build());  
 item.put("name", AttributeValue.*builder*().s(s.getName()).build());  
 item.put("email", AttributeValue.*builder*().s(s.getEmail()).build());  
 item.put("password", AttributeValue.*builder*().s(s.getPassword()).build());  
  
 client.putItem(PutItemRequest.*builder*().tableName(tableName).item(item).build());  
 }  
  
 public Student getStudent(String studentId) {  
 Map<String, AttributeValue> key = new HashMap<>();  
 key.put("studentId", AttributeValue.*builder*().s(studentId).build());  
  
 GetItemResponse response = client.getItem(GetItemRequest.*builder*().tableName(tableName).key(key).build());  
 if (!response.hasItem()) return null;  
  
 Map<String, AttributeValue> item = response.item();  
 return new Student(  
 item.get("studentId").s(),  
 item.get("name").s(),  
 item.get("email").s(),  
 item.get("password").s()  
 );  
 }  
}

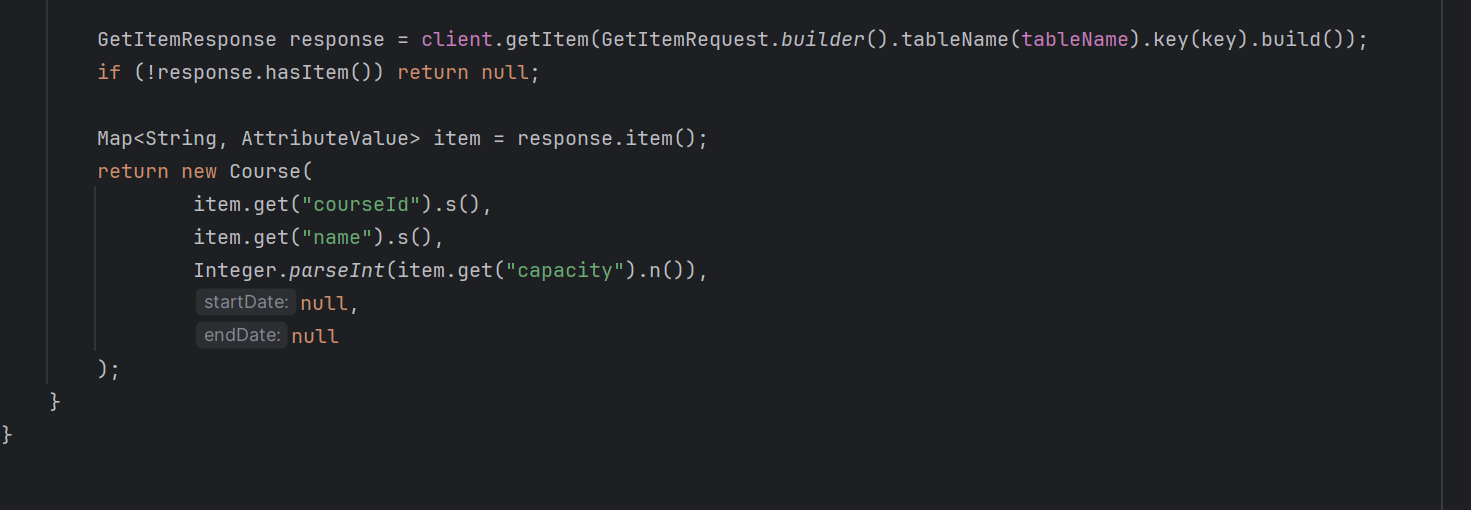




**CourseDA0.java**

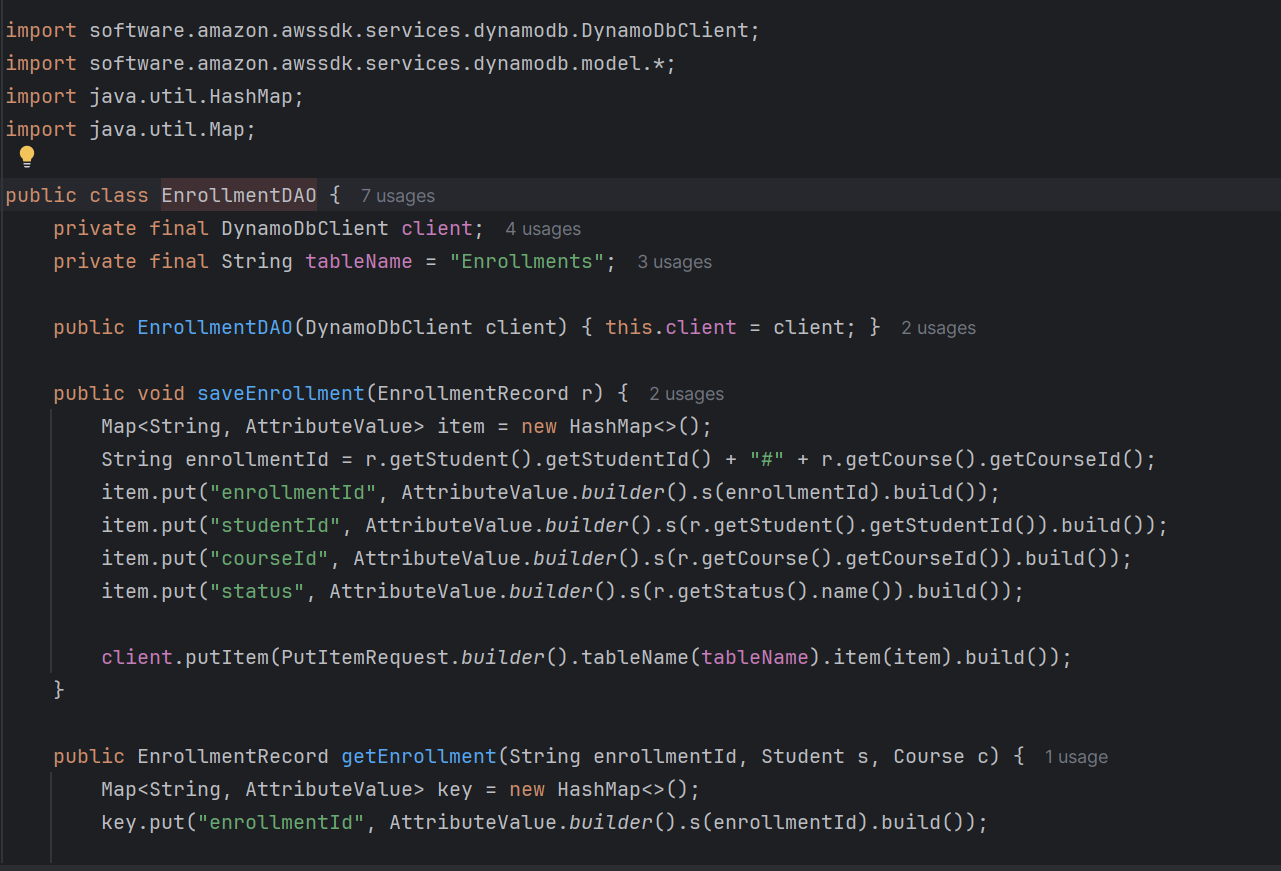
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
public class CourseDAO {  
 private final DynamoDbClient client;  
 private final String tableName = "Courses";  
  
 public CourseDAO(DynamoDbClient client) { this.client = client; }  
  
 public void saveCourse(Course c) {  
 Map<String, AttributeValue> item = new HashMap<>();  
 item.put("courseId", AttributeValue.*builder*().s(c.getCourseId()).build());  
 item.put("name", AttributeValue.*builder*().s(c.getCourseName()).build());  
 item.put("capacity", AttributeValue.*builder*().n(String.*valueOf*(c.getMaxCapacity())).build());  
 item.put("enrolledCount", AttributeValue.*builder*().n(String.*valueOf*(c.getEnrolledCount())).build());  
  
 client.putItem(PutItemRequest.*builder*().tableName(tableName).item(item).build());  
 }  
  
 public Course getCourse(String courseId) {  
 Map<String, AttributeValue> key = new HashMap<>();  
 key.put("courseId", AttributeValue.*builder*().s(courseId).build());  
  
 GetItemResponse response = client.getItem(GetItemRequest.*builder*().tableName(tableName).key(key).build());  
 if (!response.hasItem()) return null;  
  
 Map<String, AttributeValue> item = response.item();  
 return new Course(  
 item.get("courseId").s(),  
 item.get("name").s(),  
 Integer.*parseInt*(item.get("capacity").n()),  
 null,  
 null  
 );  
 }  
}



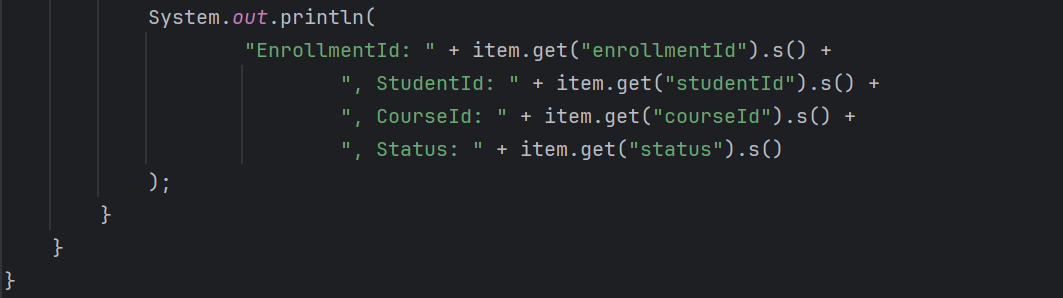


**EnrollmentDA0.java**

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
public class EnrollmentDAO {  
 private final DynamoDbClient client;  
 private final String tableName = "Enrollments";  
  
 public EnrollmentDAO(DynamoDbClient client) { this.client = client; }  
  
 public void saveEnrollment(EnrollmentRecord r) {  
 Map<String, AttributeValue> item = new HashMap<>();  
 String enrollmentId = r.getStudent().getStudentId() + "#" + r.getCourse().getCourseId();  
 item.put("enrollmentId", AttributeValue.*builder*().s(enrollmentId).build());  
 item.put("studentId", AttributeValue.*builder*().s(r.getStudent().getStudentId()).build());  
 item.put("courseId", AttributeValue.*builder*().s(r.getCourse().getCourseId()).build());  
 item.put("status", AttributeValue.*builder*().s(r.getStatus().name()).build());  
  
 client.putItem(PutItemRequest.*builder*().tableName(tableName).item(item).build());  
 }  
  
 public EnrollmentRecord getEnrollment(String enrollmentId, Student s, Course c) {  
 Map<String, AttributeValue> key = new HashMap<>();  
 key.put("enrollmentId", AttributeValue.*builder*().s(enrollmentId).build());  
  
 GetItemResponse response = client.getItem(GetItemRequest.*builder*().tableName(tableName).key(key).build());  
 if (!response.hasItem()) return null;  
  
 Map<String, AttributeValue> item = response.item();  
 return new EnrollmentRecord(s, c, enrollmentStatus.*valueOf*(item.get("status").s()));  
 }  
  
 // ------------------ NEW: Print all enrollments ------------------  
 public void printAllEnrollments() {  
 ScanRequest scanRequest = ScanRequest.*builder*()  
 .tableName(tableName)  
 .build();  
  
 ScanResponse scanResponse = client.scan(scanRequest);  
  
 if (scanResponse.count() == 0) {  
 System.*out*.println("No enrollments found.");  
 return;  
 }  
  
 System.*out*.println("Enrollments:");  
 for (Map<String, AttributeValue> item : scanResponse.items()) {  
 System.*out*.println(  
 "EnrollmentId: " + item.get("enrollmentId").s() +  
 ", StudentId: " + item.get("studentId").s() +  
 ", CourseId: " + item.get("courseId").s() +  
 ", Status: " + item.get("status").s()  
 );  
 }  
 }  
}



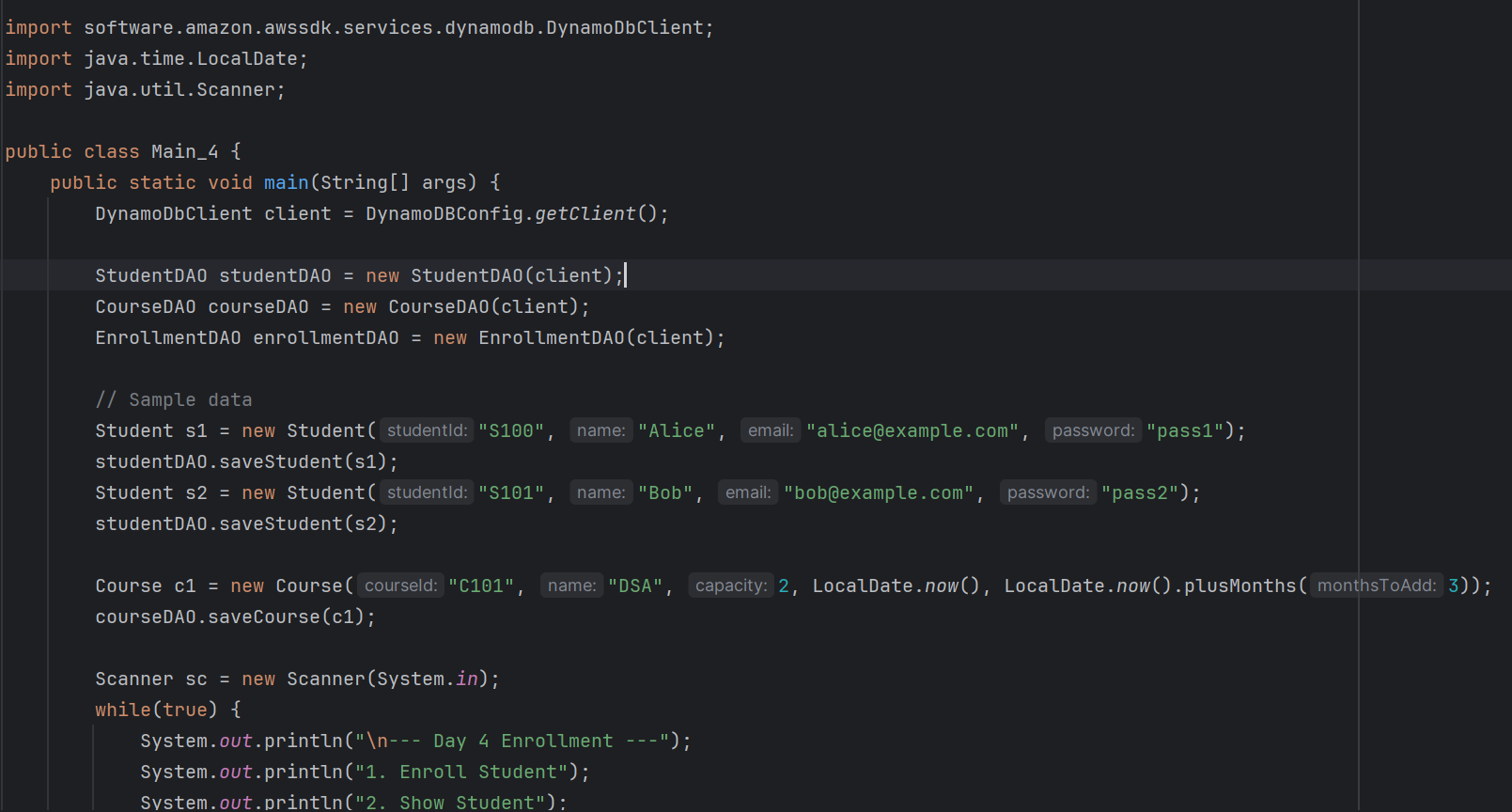


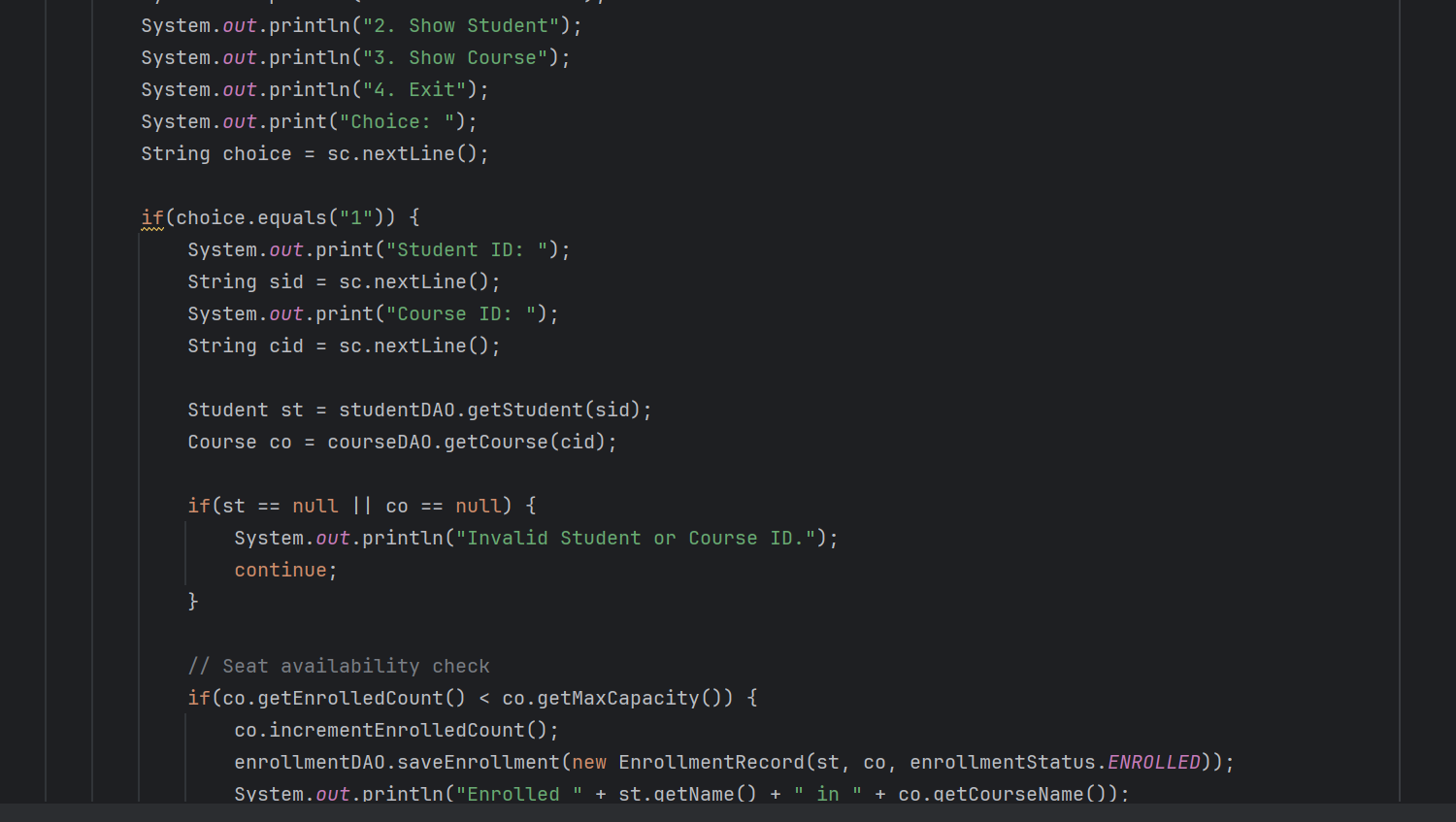


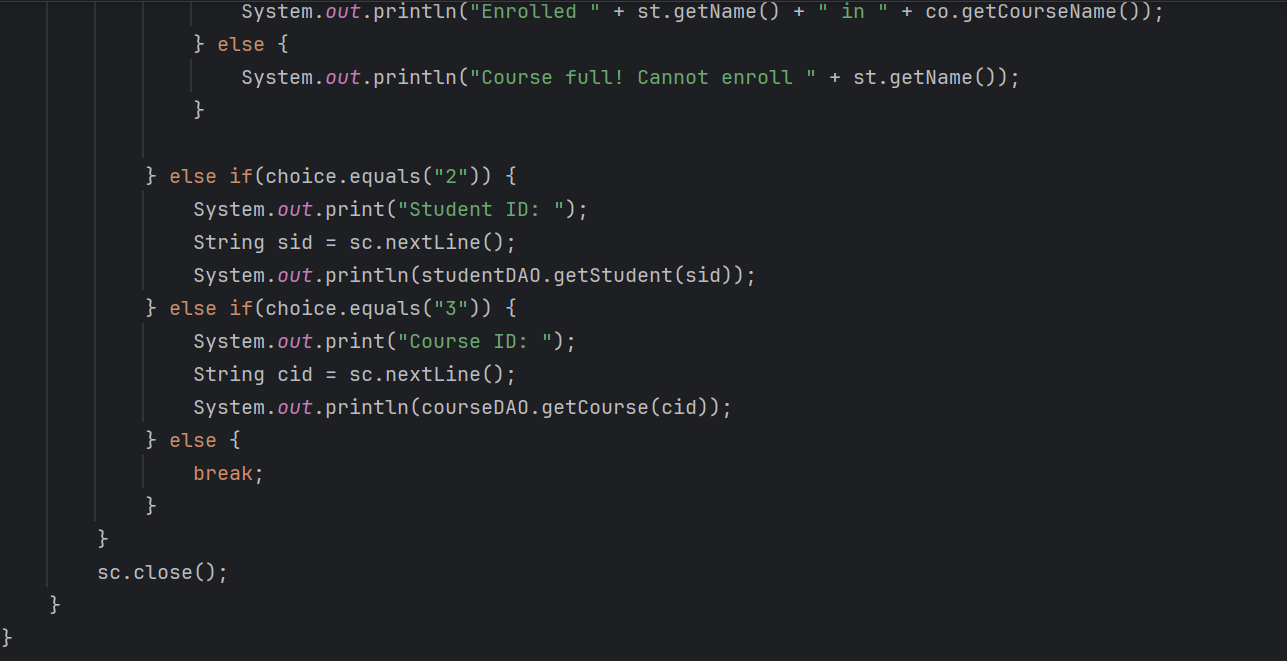
**Main\_4.java**

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import java.time.LocalDate;  
import java.util.Scanner;  
  
public class Main\_4

{  
 public static void main(String[] args) {  
 DynamoDbClient client = DynamoDBConfig.*getClient*();  
  
 StudentDAO studentDAO = new StudentDAO(client);  
 CourseDAO courseDAO = new CourseDAO(client);  
 EnrollmentDAO enrollmentDAO = new EnrollmentDAO(client);  
  
 // Sample data  
 Student s1 = new Student("S100", "Alice", "alice@example.com", "pass1");  
 studentDAO.saveStudent(s1);  
 Student s2 = new Student("S101", "Bob", "bob@example.com", "pass2");  
 studentDAO.saveStudent(s2);  
  
 Course c1 = new Course("C101", "DSA", 2, LocalDate.*now*(), LocalDate.*now*().plusMonths(3));  
 courseDAO.saveCourse(c1);  
  
 Scanner sc = new Scanner(System.*in*);  
 while(true) {  
 System.*out*.println("\n--- Day 4 Enrollment ---");  
 System.*out*.println("1. Enroll Student");  
 System.*out*.println("2. Show Student");  
 System.*out*.println("3. Show Course");  
 System.*out*.println("4. Exit");  
 System.*out*.print("Choice: ");  
 String choice = sc.nextLine();  
  
 if(choice.equals("1")) {  
 System.*out*.print("Student ID: ");  
 String sid = sc.nextLine();  
 System.*out*.print("Course ID: ");  
 String cid = sc.nextLine();  
  
 Student st = studentDAO.getStudent(sid);  
 Course co = courseDAO.getCourse(cid);  
  
 if(st == null || co == null) {  
 System.*out*.println("Invalid Student or Course ID.");  
 continue;  
 }  
  
 // Seat availability check  
 if(co.getEnrolledCount() < co.getMaxCapacity()) {  
 co.incrementEnrolledCount();  
 enrollmentDAO.saveEnrollment(new EnrollmentRecord(st, co, enrollmentStatus.*ENROLLED*));  
 System.*out*.println("Enrolled " + st.getName() + " in " + co.getCourseName());  
 } else {  
 System.*out*.println("Course full! Cannot enroll " + st.getName());  
 }  
  
 } else if(choice.equals("2")) {  
 System.*out*.print("Student ID: ");  
 String sid = sc.nextLine();  
 System.*out*.println(studentDAO.getStudent(sid));  
 } else if(choice.equals("3")) {  
 System.*out*.print("Course ID: ");  
 String cid = sc.nextLine();  
 System.*out*.println(courseDAO.getCourse(cid));  
 } else {  
 break;  
 }  
 }  
 sc.close();  
 }  
}







**Changed EnrollmentService.java for Checking seat Availability**

if (!c.isFull()) {

c.enrollStudent(s);

record = new EnrollmentRecord(s, c, enrollmentStatus.ENROLLED);

} else {

// Add to waitlist queue

record = new EnrollmentRecord(s, c, enrollmentStatus.WAITLISTED);

waitlistManager.addToWaitlist(courseId, studentId);

}

**Including these lines in EnrollmentSevice.java**

**Day 4 – Waitlist using List**

**Purpose:** Introduce a **waitlist system** for courses when full.  
**Main points:**

1. **WaitlistEntry class**
   * Stores studentId and courseId.
   * Simple POJO representing a waitlist record.
2. **WaitlistDAO**
   * Stores waitlist as a List<WaitlistEntry>.
   * Methods:
     + addToWaitlist(studentId, courseId) → adds student to waitlist for a course.
     + removeFromWaitlist(studentId, courseId) → removes student from waitlist.
     + getWaitlistForCourse(courseId) → returns all students on the waitlist for a course.
     + printAllWaitlists() → prints the current waitlist.
3. **Integration with Enrollment**
   * During enrollment:
     + If course is **full**, student is added to waitlist (WAITLISTED status).
     + Otherwise, student is enrolled normally (ENROLLED status).
   * During drop:
     + If a student drops and there’s someone on the waitlist, the **first waitlisted student is promoted** to enrolled.
4. **Main4 (Day 4 Main)**
   * Demonstrates enrollment beyond capacity.
   * Shows waitlist management:
     + Students beyond capacity are waitlisted.
     + Dropping a student promotes the first waitlisted student.
   * Prints course and waitlist status before and after drop.

**Key Concept Learned:**

* Basic waitlist functionality using ArrayList.
* Promotion of students from waitlist when seats become available.