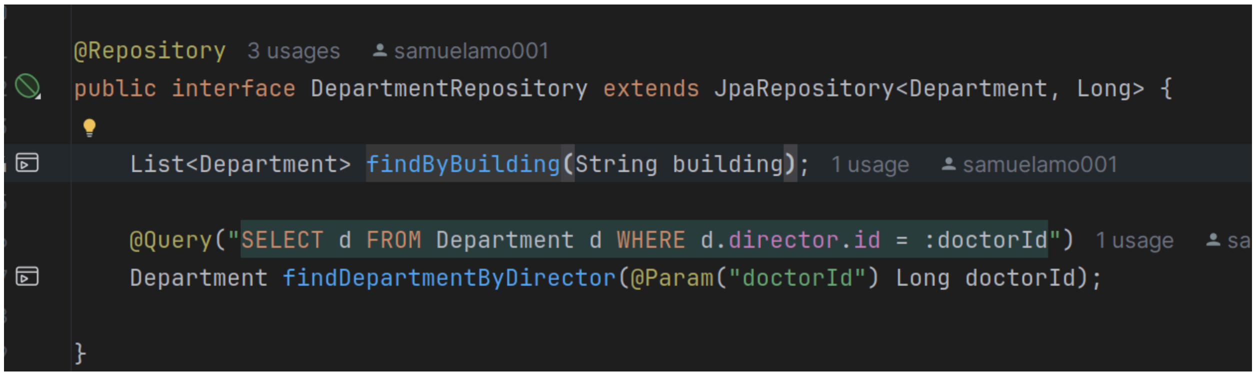
An explanation of repository pattern and query methods

Implementing Repositories in My Project

In my project, I used the Repository Pattern to manage entities related to the hospital management system. Below are the repositories I created for Department, Doctor, Nurse, Patient, and Ward entities.

DepartmentRepository

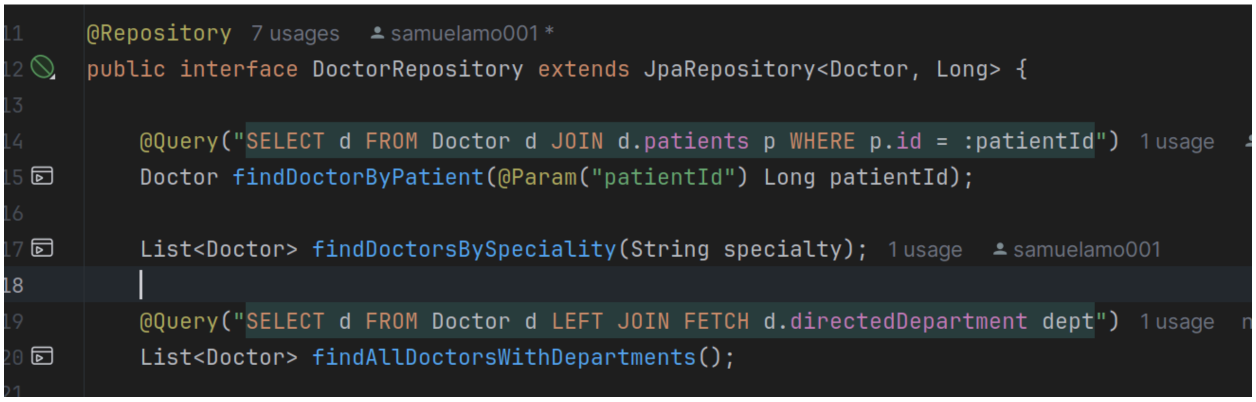
The DepartmentRepository is responsible for data access operations related to the Department entity. It extends **JpaRepository<Department, Long>** to inherit basic CRUD operations. I also defined custom query methods, such as finding departments by building and by director.



```
@Repository  3 usages   samuelamo001
public interface DepartmentRepository extends JpaRepository<Department, Long> {

    List<Department> findByBuilding(String building);  1 usage   samuelamo001

    @Query("SELECT d FROM Department d WHERE d.director.id = :doctorId")  1 usage   sa
    Department findDepartmentByDirector(@Param("doctorId") Long doctorId);

}
```

DoctorRepository

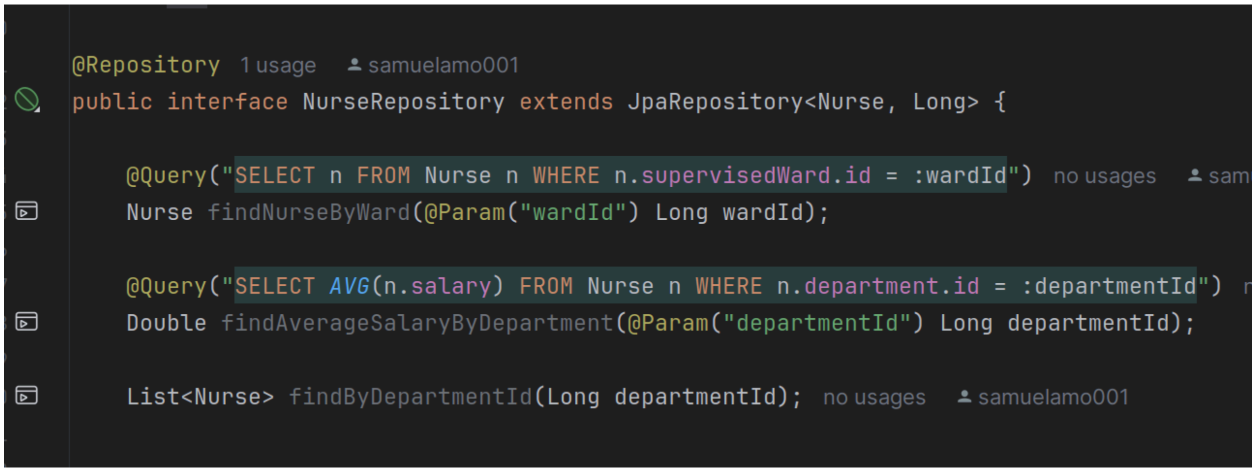
The DoctorRepository manages the Doctor entity. It includes methods to find doctors based on their specialty and patients they are treating. Additionally, I created a query to fetch all doctors along with their respective departments.



```
@Repository  7 usages   samuelamo001 *
public interface DoctorRepository extends JpaRepository<Doctor, Long> {

    @Query("SELECT d FROM Doctor d JOIN d.patients p WHERE p.id = :patientId")  1 usage
    Doctor findDoctorByPatient(@Param("patientId") Long patientId);

    List<Doctor> findDoctorsBySpeciality(String specialty);  1 usage   samuelamo001

    @Query("SELECT d FROM Doctor d LEFT JOIN FETCH d.directedDepartment dept")  1 usage
    List<Doctor> findAllDoctorsWithDepartments();
```

NurseRepository

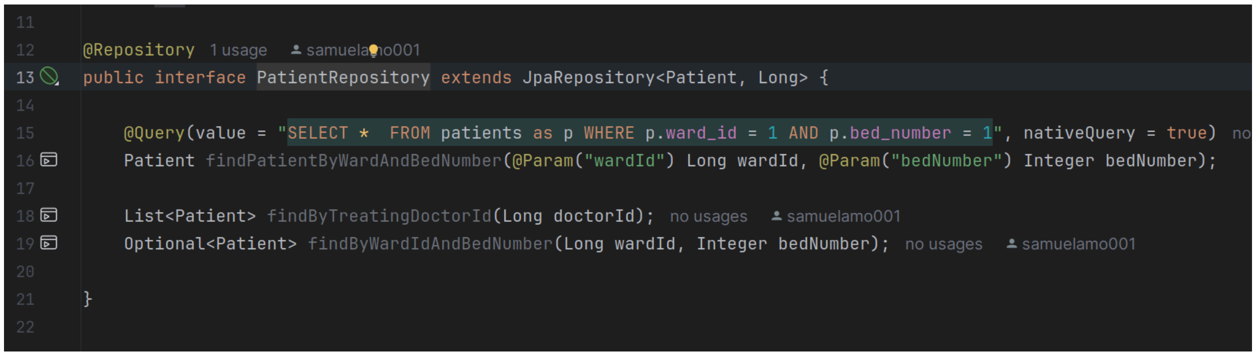
For the Nurse entity, the NurseRepository contains methods to find nurses by their supervised ward or department. It also calculates the average salary of nurses within a department.



```
@Repository  1 usage   samuelamo001
public interface NurseRepository extends JpaRepository<Nurse, Long> {

    @Query("SELECT n FROM Nurse n WHERE n.supervisedWard.id = :wardId")  no usages   sam
    Nurse findNurseByWard(@Param("wardId") Long wardId);

    @Query("SELECT AVG(n.salary) FROM Nurse n WHERE n.department.id = :departmentId")  r
    Double findAverageSalaryByDepartment(@Param("departmentId") Long departmentId);

    List<Nurse> findByDepartmentId(Long departmentId);  no usages   samuelamo001
```

PatientRepository

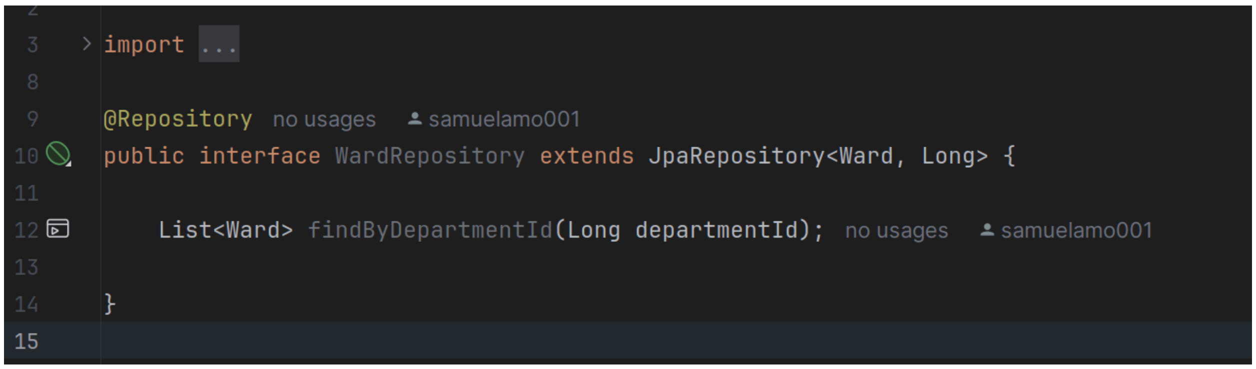
The PatientRepository manages the Patient entity, with methods to find patients by their ward and bed number, as well as those treated by a specific doctor. I used both JPQL and native SQL queries to accomplish this.



```

@Repository  1 usage   samuelamo001
public interface PatientRepository extends JpaRepository<Patient, Long> {

    @Query(value = "SELECT *  FROM patients as p WHERE p.ward_id = 1 AND p.bed_number = 1", nativeQuery = true)  no
    Patient findPatientByWardAndBedNumber(@Param("wardId") Long wardId, @Param("bedNumber") Integer bedNumber);

    List<Patient> findByTreatingDoctorId(Long doctorId);  no usages   samuelamo001
    Optional<Patient> findByWardIdAndBedNumber(Long wardId, Integer bedNumber);  no usages   samuelamo001

}

```

WardRepository

The WardRepository is responsible for operations related to the Ward entity. It includes a method to find wards by department.



```
import ...

@Repository  no usages  samuelamo001
public interface WardRepository extends JpaRepository<Ward, Long> {

    List<Ward> findByDepartmentId(Long departmentId);  no usages  samuelamo001

}

```

Conclusion

The Repository Pattern, combined with the power of Spring Data's query methods, offers a robust and efficient way to handle data access in applications. By abstracting the data access layer, the pattern promotes clean, maintainable, and testable code, which is essential for building scalable enterprise applications.

In my project, I've utilized these principles to manage the complexity of the hospital management system, ensuring that data access operations are both efficient and easy to maintain.