Critter Chronologer Recreation Guide (Exam Reference Version)

This step-by-step guide helps you recreate your working project in exam conditions using the exact same flow you used originally.

Follow these steps in the given order. Make changes in files exactly as described. All tests will pass if done correctly.

STEP 1: START WITH BASIC ENTITY STRUCTURE

----

1 Create `Customer.java` in `com.udacity.jdnd.course3.critter.user`

- Fields: id, name, phoneNumber, notes, List<Pet> pets
- Annotations: @Entity, @OneToMany(mappedBy = "owner")

2 Create `Pet.java` in `com.udacity.jdnd.course3.critter.pet`

- Fields: id, PetType type, name, birthDate, notes, Customer owner
- Annotations: @Entity, @ManyToOne

STEP 2: SET UP REPOSITORIES

----

3 Create interfaces:

- `CustomerRepository.java` extends JpaRepository<Customer, Long>
- `PetRepository.java` extends JpaRepository<Pet, Long>, add:
  List<Pet> findAllByOwnerId(Long ownerId);

STEP 3: SET UP DTOs

----

4 Create `CustomerDTO.java`, `PetDTO.java` with fields matching the ones used in test class (id, name, ownerId, etc.)

STEP 4: BUILD USER SERVICE

----

5 Create `UserService.java`

- `saveCustomer(CustomerDTO)` create and save Customer
- `getAllCustomers()` return list of CustomerDTOs
- `getOwnerByPet(long petId)` return CustomerDTO of pets owner
- `saveEmployee(EmployeeDTO)`
- `getEmployee(long employeeId)`
- `setAvailability(Set<DayOfWeek>, long employeeId)`
- `findMatchingEmployees(EmployeeRequestDTO)`

STEP 5: USER CONTROLLER

----

6 Create `UserController.java`

- Add mappings:

  POST /user/customer

  GET /user/customer

  GET /user/customer/pet/{petId}

  POST /user/employee

  GET /user/employee/{id}

  PUT /user/employee/{id}

  GET /user/employee/availability

STEP 6: PET SERVICE

----

7 Create `PetService.java`

- `savePet(PetDTO)` set owner, save pet, update customer
- `getPetById(long id)`
- `getPetsByOwner(long ownerId)`

STEP 7: PET CONTROLLER

----

8 Create `PetController.java`

- POST /pet
- GET /pet/{id}
- GET /pet/owner/{ownerId}

STEP 8: SCHEDULE SERVICE & ENTITIES

----

9 Create `Schedule.java`

- Fields: id, List<Employee> employees, List<Pet> pets, LocalDate date, Set<EmployeeSkill> activities
- Annotations: @Entity, @ManyToMany

 Create `ScheduleRepository.java`

- findAllByPetsContains(Pet pet)
- findAllByEmployeesContains(Employee employee)

11 Create `ScheduleService.java`

- `createSchedule(ScheduleDTO)`
- `getAllSchedules()`
- `getScheduleForPet(long petId)`
- `getScheduleForEmployee(long empId)`
- `getScheduleForCustomer(long customerId)`

STEP 9: SCHEDULE CONTROLLER

----

12 Create `ScheduleController.java`

- POST /schedule
- GET /schedule
- GET /schedule/pet/{id}, /employee/{id}, /customer/{id}

STEP 10: RUN ALL TESTS

----

13 Run `CritterFunctionalTest.java` from IntelliJ

- All 9 tests should now pass
- If any test fails, check endpoint mappings and service logic

END OF GUIDE

Use this sequence to avoid confusion or panic during the exam. Youve got this!