Mastery Checkpoint: Database Integration

Tid: 3 timer

Verktøy: Java + Spring Boot med JPA/Hibernate + JUnit
eller: C# + Entity Framework Core + NUnit

Hjelpemidler: Google, Sana og eget materiell, men *IKKE* lov å snakke sammen/hjelpe hverandre.

Rammer for oppgaven

- Du skal lage en backend kodeløsning m/database for et selskap som leier ut el-sparkesykler.
- Del prosjektet på GitHub. Gjør det public, evt. legg til stianl (Java) eller tomas-sandnes (C#) som collaborators.
- Det skal **ikke** lages noe konsoll- eller webgrensesnitt. Du skal i stedet demonstrere løsningens funksjonalitet gjennom **enhetstester** (JUnit/NUnit).
- Husk også temaene vi har hatt om tidligere, som lagdeling og SOLID.
- Når du er ferdig eller rett før tiden går ut, gjør en siste commit og push til GitHub.

Domene

- **Scooter**
  - Id (unik)
  - Brand (f.eks. "Voi", "Tier")
  - Battery capacity (%)
  - Status (Available, In Use, Out of Order, …)
- **User**
  - Id (unik)
  - Name
  - Phone number
- **Trip**
  - Id (unik)
  - Start time
  - End time (null hvis ikke avsluttet enda)
  - Distance (km)
  - Cost (NOK)
  - Koblinger til Scooter og User

Del 1

Modellering

- Opprett entitetsklasser og passende relasjoner gjennom din ORM (JPA eller EF Core):
- En bruker kan kjøre mange turer.
- En elsparkesykkel («scooter») kan benyttes til mange turer.

Datalagring

- Bruk code-first (top-down) for generering av DB (i PostgreSQL).
- *For Java:* H2 in-memory DB er greit alternativ til Postgres for testingen.

Spørringer

Implementer metoder som kan:

- Finne alle scootere som er **ledige** og har batteriprosent over 20.

- Hente alle turer for en bestemt bruker, sortert etter starttidspunkt.
- Regne ut **gjennomsnittlig pris per km** for alle turer.
- Finne hvilken bruker som har kjørt **flest turer**.

Tester
Gjør initialisering og tester som:

- Oppretter et knippe brukere, scootere og turer.
- Kjører spørringene over og verifiserer at resultatene stemmer.
- Viser at domenelogikk for elsparkesykler opprettholdes, f.eks. at en scooter ikke kan ha to aktive turer samtidig.

Del 2 (hvis tid til overs)

Oppført i foreslått rekkefølge, men det er lov å løse de i den rekkefølgen dere ønsker.

a) Legg til noen tester som benytter mocking (om du ikke allerede har gjort det).
b) Opprett et DTO objekt som inneholder info om Turer. Konkret: Antall turer, lengste tur, korteste tur og gjennomsnittlig lengde på en tur. Skriv en metode som returnerer data på formatet til dette objektet samt enhetstest(er) for å sjekke at det virker.
c) Bruk testcontainers til testing i stedet for lokal Postgres/H2.
d) Legg til entiteten **LocationEvent** som lagrer posisjonen til en scooter (tidspunkt & GPS-koordinater).
e) Lag en metode som tar brukerens GPS-koordinat inn (f.eks. 59.93, 10.80) og finner ut hvor langt unna nærmeste 3 scootere er basert på siste LocationEvent per scooter. Det er ok å mocke/fake svaret for utregningen av avstand.

--- Slutt på oppgavesettet ---