Lesson 1: Spring framework and Dependency injection

- Spring context
- Setter injection
- Constructor injection
- Init() method
- Autowiring
- Field injection
- @Autowired
- @qualifier
- XML configuration
- Classpath scanning + Autowiring (@Service, @Repository, @Component)
- Java configuration

You do **not** need to write Spring XML configuration, but you need to understand given XML configuration for setter and constructor injection
You should be able to configure dependency injection with annotations for

- setter injection
- constructor injection
- field injection by type
- field injection by name

You should be able to write simple Java configuration for both setter and constructor injection

Lesson 2: Spring boot

- @SpringBootApplication
- run() method
- Application.properties
- Layers and responsibilities
- Service class
- @Profile and activate a profile

You should be able to write simple Spring Boot application, including the Application class and dependency injection.
You should be able to write code using profiles

- AOP
- Crosscutting concern
- @Aspect
- @Before, @After, @AfterReturning, @Afterthrowing, @Around
- Getting the return value, exception, parameters, target class.
- Advantages and disadvantages

You do not need to write AOP code, but you need to understand given AOP code.

Lesson 3:

- Different options to go to the database
- JDBC
- Advantages and disadvantages
- NamedParameterJdbcTemplate

You do not need to write Spring JDBC code, but you need to understand given Spring JDBC code.

- JPA
- Advantages and disadvantages
- Spring repository interface
- @Entity, @Id, @GeneratedValue
- Id generation: identity column, sequence

You should be able to write entity classes including the mapping and you should be able to write repository interfaces.

Lesson 4: JPA mapping

- Entity class mapping (@Transient, @Lob, @Column, @Temporal)
- Association mapping
- ManyToOne
- OneToMany
- ManyToMany
- mappedBy
- @JoinColumn
- @JoinTable
- Cascading
- Fetching (eager and lazy)
- @OrderColumn (List)
- @OrderBy (List)
- @MapValue (Map)

You should be able to map a domain model on a database.
You should be able to draw the database tables and their columns based on a JPA mapping.

Lesson 5: JPA mapping

- Inheritance
  - Single table
  - Joined tables
  - Table per class
- 1 class on 2 tables
- 2 classes on 1 table
- Composite keys/ids
- DTO classes

You should be able to map a domain model on a database.
You should be able to draw the database tables and their columns based on a JPA mapping.
You should be able to write DTO classes

Lesson 6: queries

- Queries defined with method names
- @Query
- Named queries
- Native queries
- Named parameters
- Join on objects (ToOne): use . notation
- Join on collections (ToMany): use join
- Make ManyToOne lazy
- Fetch join for ManyToOne
- Distinct … fetch join for OneToMany
- Optimization

You should be able write queries in a repository using method names and @Query
You do not need to write queries using named queries or native queries.

Lesson 7: Transactions

- Local/global transactions
- 2 phase commit
- Transaction propagation (required, requires_new, supports, not_supported)
- Isolation level
- Dirty read
- Non repeatable read
- Phantom read
- @Transactional
- Lost update problem
- First commit wins vs. last commit wins
- @Version

You should be able configure transactions the right way on methods.

Lesson 8: REST

- GET, PUT, POST, DELETE http requests
- Idempotent
- @RequestMapping, @GetMapping, etc.
- PathVariable
- Query Parameter
- ResponseEntity
- RestTemplate
- REST API design

You should be able to write a REST controller according API design best practices
You do not need to write a RestTemplate