1 What **is a REST API, and why is it commonly used in web development?**

* What does ReST stand for?
  + Representational state transfer
* What are the main principles of RESTful architecture?
  + Stateless (It has all the info it needs)
  + Frontend and backend are separate and allows simultaneous development.
  + It’s uniform. Always uses GET, POST, DELETE, PUT, etc.
  + Everything is treated as a resource AKA an object basically.
* What are semantic URLs, and how do they improve the readability of RESTful APIs?
  + It’s readable URI, uses easy to read stuff instead of something that looks like a hashed code and is unreadable.

2 **Explain the purpose of using DTO (Data Transfer Object) classes in a java application:**

* Where are DTO classes typically used in the context of RESTful APIs?
  + DTOs are used mostly to obfuscate full entities. Entities carries everything we want them to have whereas DTOs are masks we apply and we expose the DTO to the endpoints, and controllers, so that we can selectively only show what we want to.
  + DTOs can also use nested helper classes to selectively only show the data we want to. We can make many and use these selectively for different endpoints if desired.
* How can DTO classes help in decoupling the domain model from the API response?
  + See above.
* Should DTO classes import domain model classes? Why or why not?
  + No, they should not import “Doman model” (Entities) because it breaks separation of concern and makes the DTO less flexible.
  + It creates dependencies. Changes to the DTO would make changes to entity and vice versa.
  + We can create separate mapper classes if we want to convert DTOs to entities.

3 **What is the difference between HTTP GET, POST, PUT, and DELETE methods? Provide examples for each:**

* How are these HTTP methods typically used in RESTful API design?
  + GET – Retrieve data
    - Example: GET /skilessons returns all ski lessons.
  + POST – Create a new resource
    - Example: POST /skilessons creates a new ski lesson based on data in the request body.
  + PUT – Update or replace a resource entirely
    - Example: PUT /skilessons/5 would update lesson ID 5.
  + DELETE – Remove a resource
    - Example: DELETE skilessons/5 would delete lesson ID 5.
* What are the main differences between PUT and POST methods in terms of idempotency and safety?
  + POST is not idempotent and will create 2 resources if send twice. PUT is idempotent and multiple requests will always end the same 1 result.
* How can the PATCH method be used to update resources in a RESTful API?
  + Patch can be used to update resources partially. PUT, per default, requires ALL the fields to be updated when we update. That being said, my update method in SkiLessonController uses PUT as if it was patch. Theoretically PUT but practically PATCH.
* What is the purpose of the OPTIONS method in HTTP?
  + It’s a request you can send to an API and figure out what you’re allowed and can do with that URI. The name of the request almost explains it. “What are my options here”

4 **What is error handling in REST APIs, and why is it important?**

* How can HTTP status codes be used to communicate errors in a RESTful API?
  + We can use status codes to communicate the category of the result and the specifics of it.
* What are some common error codes and their meanings in the context of RESTful APIs?
  + 2xx = Success (Example: 200 ok, 201 Created)
  + 4xx = Client error (Example 404 not found)
  + 5xx = Server error (Crashes or misfigurations)
* How can custom error messages be returned in a RESTful API response?
  + I use an ApiException class, ExceptionController and a Message class.  
    The message class can send back Json when the controller hits statuscode
  + Example: throw new ApiException(404, "Cannot delete: ski lesson not found");

5 **Describe how JSON is typically used in RESTful APIs for data exchange**

* What are the advantages of using JSON over other data formats like XML?
  + Simple syntax
  + Lightweight
  + Easy to read for people
  + Good for frontend because native support (It is called “JavaScript object Notation” after all)
  + Easy to map to object of us in java. We use the Jackson library for it.
* How can JSON be serialized and deserialized in Java using libraries like Jackson?
  + Serialization = Java object → JSON
  + Deserialization = JSON → Java object
  + My project uses ctx.bodyAsClass(SkiLesson.class). Here we are using Javalin but Javalin uses Jackson under the hood.
* What is the purpose of the Content-Type header in an HTTP request or response?
  + To tell the client and server what we are working with so that the payload can be interpreted correctly. Example: Content-Type: application/json

6 **What is logging in the context of application development, and how can it improve error management and debugging?**

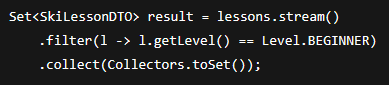
* How can logging be implemented in a Java application using libraries like Logback?
  + In Java, logging is typically done using SLF4J
  + We add org.slf4j in the POM file.
* What are the different log levels, and when should they be used in an application?
  + There are 6 levels
  + We have most commonly used, debug, info and error
* How can log messages be formatted to include relevant information like timestamps and log levels?
  + Usually done though an .xml file
  + Here, with certain syntax you can produce log messages that are easy to read for people.

7 **Explain the concept of dependency injection and its benefits in creating scalable applications.**

* Where do you use dependency injection in your application?
  + In our POM file by using maven
* What is the alternative to dependency injection?
  + Alternative is to download libraries manually and adding them to the project.
  + Kind terrible, doesn’t scale, is error prone and other annoying problems.
* How does dependency injection help in your code?
  + Makes it fast and easy
  + Keeps project consistent across environments like if I wanted to run the project on a different PC or server.
  + Reduces a lot of manual labor.

**8. How do streams and lambda expressions work in Java, and how can they help in processing collections?**

**9 Show an example of using streams and lambda expressions to filter, map, and reduce elements in a collection.**

* **What are the advantages of using streams and lambda expressions over traditional iteration methods?**
* **Show examples of grouping elements in a collection using streams and lambda expressions.**
  + Streams are a sequence of methods that are applied to an object and only returns a value of once the entire stream is done
  + We can skip writing customs loops and instead use streams to get the same result.
  + It’s much more readable than loops.
  + Example:  
    
  + This filters a list of lessons to only include beginner ones — without writing loops.
  + Lambda helps write functional and local code.
  + No need to create additional classes with additional methods.
  + Can write the code in-line.
  + Example is in above picture where it’s annotated by “ l -> “

10 **What are Java Generics, and why are they used? Provide an example of a generic class or interface.**

* How can generics help in creating reusable and type-safe code?
  + By not being specific with something like what object to use but rather just that is had to be any kind of object.
  + Example would be the IDAO<T> class
* Why do we want to use interface data types over concrete data types?
  + Easier to work with. It’s more of a plug and play than concrete data types. If we wanted to swap it out for something else in the future then it’s much easier.
* How can you restrict the types that can be used with a generic class or method?
  + We can do something like IDAO<T extends class>
  + But to be fair. I have yet to understand the appeal of interfaces yet. It has been explained to me but I’ve yet to feel any real world proof.

11 **Explain the role of JPA (Java Persistence API) in Java applications and how it manages data persistence.**

* + We use hibernate which implements JPA.
  + Its role is to map java objects to entities instead of using raw SQL.
  + We do this by specifying which classes are entities with @Entity and also adding them to our Hibernate config file.

12 **What are Hibernate and EntityManagerFactory, and how do they facilitate ORM (Object-Relational Mapping) in Java?**

* What is the purpose of the HibernateConfig file in your application?
  + Sets up the Hibernate configuration programmatically
  + Registers entity classes
* What is a DAO (Data Access Object), and how does it fit into the design of a backend service?
  + Separation of concern
  + Tells the program how to work with the datatypes in handles.
  + Example in SkiLessonDAO handles all DB operations for ski lessons and InstructorDAO does the same for instructor entities.

13 **What is the purpose of some common JPA annotations**

* What is the purpose of using annotations like @Entity, @Table, and @Column in JPA entities?
  + @Entity marks the class as an entity so that JPA knows how to work with it
  + @Table is an optional annotation for setting a name of the table.
  + Main purpose is to mark different variables for JPA to work with.
* What is the difference between FetchType.LAZY and FetchType.EAGER in JPA entity relationships?
  + Lazy only loads when accessed which improves performance.
  + Eager I used when we always want to related data. Example @ManyToOne.
* What is the purpose of the @Id and @GeneratedValue annotations in JPA entity classes?
  + Id marks the primary key and generatedValue lets the database create unique constraint ID automatically.

14 **Describe the One-to-Many relationship in a database and how it can be represented using JPA annotations.**

* What is the difference between unidirectional and bidirectional relationships in JPA entities?
  + Uni is 1 when entity knows of the other but not the other way
  + Bi is when both knows each other.
    - Example: Instructor has a Set<SkiLesson>, and SkiLesson has a reference to its Instructor.
  + My project uses Bidirectional relationships.
* How can cascading operations be used to maintain consistency in a One-to-Many relationship?
  + Personally, I used it in my project for when we remove an instructor. By specifying the cascade type, I can avoid removing a ski lesson just because I remove an instructor that is associated with it.
* What are some common pitfalls to avoid when working with cascading operations in JPA entities?
  + As specified above, we want to avoid unintentional deletion
  + Circular references. We can get stuck in loops when serializations and deserializations happen.
  + Missing mappedBy which creates separate tables and is not what we want.

15 **Explain how external APIs can be integrated into an application and some considerations when calling these APIs.**

* What are RESTful APIs, and how can they be consumed in a Java application?
  + Restful APIs exposes endpoints that we can access through HTTP
  + We can consume them in our application by parsing the JSON responses to java objects.
* How can you handle authentication when calling external APIs?
  + API keys either as header or query parameter
  + Bearer tokens
* How can we best consume JSON data from an external API in a Java application?
  + Usually with a library like Jackson. Then if we want, we can use DTOs to handle the data.

16 **What is JWT (JSON Web Token), and how does it enhance security in RESTful applications?**

* How can JWT be used for authentication and authorization in a RESTful API?
  + User can login, they’re then verified by the server and sent a JWT token, the token is then send with future requests. Usually the token will expire after some time
* What are the main components of a JWT token, and how are they used for secure communication?
  + Header, payload and signature.
  + Header specifies what the payload is signed with
  + Payload is the actual data
  + Signature verifies that token is valid and untampered.
* How can JWT tokens be validated and decoded in a Java application?
  + Usually done by a library. It’s something that can collect the signature, parse the token and compare it to the secret key we’ve set and verify its validity. Secret key can be set with environment variables to hide it.

17 **How does role-based access control work, and how can it be applied to secure REST API endpoints?**

* Show in your code how you can restrict access to certain endpoints based on user roles.
  + I have not done this but you would make a JWT middleware class that can check for valid tokens. Then you slap the method onto the endpoints you want to protect.
* How are the user roles from the token used to determine access rights in your application?
  + In a JWT based system, the role could be read from the payload. Depending on what role is inside, the controllers, routes or services could have different behaviour like only allowing instructors to create lessons or admins deleting them.

18 **Describe the differences between integration testing and unit testing in the context of REST API endpoints.**

* What is RestAssured, and how can it be used to test RESTful APIs in Java?
  + It’s a library that allows for easy integration testing. It lets you send HTTP requests and checks if the status code and Json is correct.
* How can you test access control and error handling in your RESTful API using integration tests?
  + I’m not entirely sure but I think it’s with using it’s methods like .header and .statuscode
* How does Hamcrest help in writing expressive assertions in your test cases?
  + I don’t think I actively used it but it’s supposed to be an easy to read way of asserting the desired results of a test and figuring out if it’s correct or the test fails.

19 **Give an overview of how we deploy our backend applications to the cloud**

* What is CI/CD?
* How are we using GitHub Actions to automate the deployment of your application?
* What is Docker Hub?
* How can we use Docker Compose to manage multi-container applications?
* How do we re-deploy our application when we have made changes to our code?

I have so catching up to do on this one… sorry