Now that you've completed the section, let's see if you can answer these questions. The answer sheet is available as a PDF in the "Support Files" folder.

Section 3: Creating an API

1) If you want to have an immutable entity, what is required to be able to use it in a rest controller?
   a. You can't have immutable entities. The mapper needs a setter to fill in the data, but the setter can be package-private.
   b. You need a constructor that gets all field values as parameters.
   c. The Constructor needs the **@JsonCreator** annotation.
   d. You need a factory method instead of an all-fields constructor.
   e. The parameters need the **@JsonProperty** annotation.

2) When you add hypermedia links to an entity, what is an important criteria to take care of?
   a. To have a link called **id** because every rest entity has an ID that points to the entity
   b. To have a link called **self** because every rest entity should have a self link that points to the entity
   c. The HTTP method should be stated in the link, so clients know which HTTP methods are allowed
   d. Links should point to actions such as add and delete
   e. Links should point to resources
   f. All links should also be present in the service document

3) When your controller returns a generic class like **List<ENTITY>**, which among the following functions will you use to create the correct class from the JSON?
   a. mapper.readValue(result, List<ENTITY>.class);
   b. mapper.readValue(result, new List<ENTITY>(){});
   c. mapper.readValue(result, new ArrayList<ENTITY>());
   d. mapper.readValue(result, new TypeReference<List<ENTITY>>() {})
   e. This is not possible; you have to create a concrete class that is fully typed.

4) When you use the **linkTo(methodOn(…))** link builder, which among the following criteria is important when dealing with method parameters?
   a. You can set all to null
   b. You might have to provide bogus values when the method is overloaded with multiple implementations
   c. You need to provide values for all fields
   d. Values that are fetched from the URL path need to be matching the desired entity
   e. Values that are fetched from **RequestBody** need to be matching the desired entity
   f. Values that are fetched from the URL parameters need to be matching the desired entity

5) Which among the following ways to control the HTTP status are valid?
   a. Use **ResponseEntity<>** as the return value and provide the status here
   b. Use **HttpEntity<>** as the return value and provide the status here
   c. Throw an Exception and catch it in a method annotated with **@ExceptionHandler**