Java Course Track Prequel

1. Your application is going to allow you to manage music Tracks. Do your work in **Labs/TrackStarter.**

   A. Your application is going to handle media Tracks. So the first thing you need to do is create a **Track** class. Your Track class should have the following properties:

      1. Id – should be unique
      2. Artist
      3. Album
      4. Duration
      5. Date
      6. Format – can have one of the following values: CD, MP3 or OGG.

      The Id should be an Int, the other properties can be Strings. Unless you can think of more appropriate types for them. Add whatever methods you feel are appropriate for such a class.

   B. You have to write a **service** and **dao** for your application. The **TrackService** class should allow you to

      1. Retrieve a Track using a Track id.
      2. Retrieve all tracks.
      3. Insert a track.
      4. Update a track.
      5. Delete a track

   C. Create a **TrackDAO** to be the keeper of the data. You will have to decide what Collection class you want to use to store your tracks.

   D. Write some code to test your classes. This can be either in the form of a main method or unit tests.

2. Add functionality to your TrackDAO to generate sequential id numbers for Tracks when they are created.
3. Make your Track class sortable by 'natural order'. You can decide what that order should be. Write code to create and sort a list of 5 Tracks.
4. Sort your list using some order other than the natural order. Use Lambda expressions.
5. Write a method called **findBy** that returns a list of Tracks based on search criteria that that you specify as an argument. Use the appropriate Java interface for filtering. Use Streams and Lambda expressions.

6. Springify your application.
   A. Make your TrackService and TrackDAO into beans. You can use either **@Service, @Repository, @Autowired,** etc, or create an **@Configuration** file with **@Bean** methods.
   B. Have your main class create an **AnnotationConfigApplicationContext**. Make sure it can find your configuration. (Note that there is already an application class called **PlaylistApp** in the *app* package which has been commented out. You can uncomment and use that one if you want, but you will have to have all the pieces in place before it will run)
   C. Get a reference to your TrackService bean from the context using **context.getBean(…)** and invoke operations on it.
7. Create a second DAO (Homework for Extra Credit)
   A. Create a copy of your current DAO with some appropriate name.
      1. Have the create method in your new DAO change the name of the customer in some way to mark that they come from the new DAO. This will useful for debugging.
   B. Put your two DAOs into seperate profiles
      1. Use the **@Profile** annotation.

C. Change your application code to set an active profile to choose one or the other of your DAOs.

1. AnntationConfigApplicationContext context = …
2. context.getEnvironment()::setActiveProfiles(“dev”)
3. context.scan(packages if any)
4. context.register(ConfigClasses if any)
5. context.refresh()

D. In Unit tests you can set the profile by using **@ActiveProfiles**