ExerciseSheet05

After you have implemented all that stuff from ExerciseSheet04, you think about it for a while (and learn some cool new Stuff about Generics). You recognize that you have implemented nearly the same thing for Tracks and Artists when it comes to Formatters and Comparators.

To improve your code, you decide to change these using the concept of Generics.

For the comparators, you additionally decide to use the already existing `Comparator` interface located in `java.util` package.

After that, the manager comes back to you again and asks for a possibilty to store all the tracks of his Artists in a list. (Really, he didn't ask for that before, although all requests had this aim :-)) Here we start with a newly created class called `MyTrackContainer`.

He would love to sort the lists within this container, but additionally, he wants to be able to filter tracks by the duration of a Track or the title of a Track. For this task, you decide to implement a so called "Matcher", using again the concepts of abstract classes, inheritance and generics.

> Be aware that it may happen that you may have questions during implementation. Don't hesitate to ask the "managers" (your lecturers and colleagues) by posting your question to the forum!

Tasks

- Implement the Comparators `DurationComparator` `TitleComparator` by using the Generic Interface `Comparator<T>` from the `java.util` package for `Track`. Refer to https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html for more information about this interface.
- Implement the Comparator `WriterComparator` like the other two Comparators for `Track`. Be aware that you should use the natural ordering of the class Artist. This means you should implement the interface `Comparable` located in the `java.lang` package in the `Artist` class first. Refer to https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html for more information about this interface.
- Next let's come to the Formatters. In this exercise you need to implement only one Formatter, called `ShortTrackFormatter`. This Formatter should implement the generic interface `MyFormatter<T>`. It's recommended to implement the method `format(T t)`before the other ones, as you understand the needed format for the other ones better after this one.
- Last before we come to the "big picture", we implement some Matchers. Matchers are used to check if a given Object matches a predefined pattern. Implement the two concrete classes `DurationMatcher`and `TitleMatcher` for `Track` based on the abstract generic class `MyMatcher<T>`.
- Implement a Container called `MyTrackContainer`. This is the heart of our exercise. Within the container, you should heavily use various collections and their methods.
- Implement a class `DemoApp` with a main method. In the main method, create an Object of Type MyTrackContainer and add Tracks to this one. Try to use all methods you have implemented to add, remove, filter and sort your container. Don't forget to print your Tracklists using your newly created Formatter. In the best case you add a text menu for ease-of-use. (recommended)

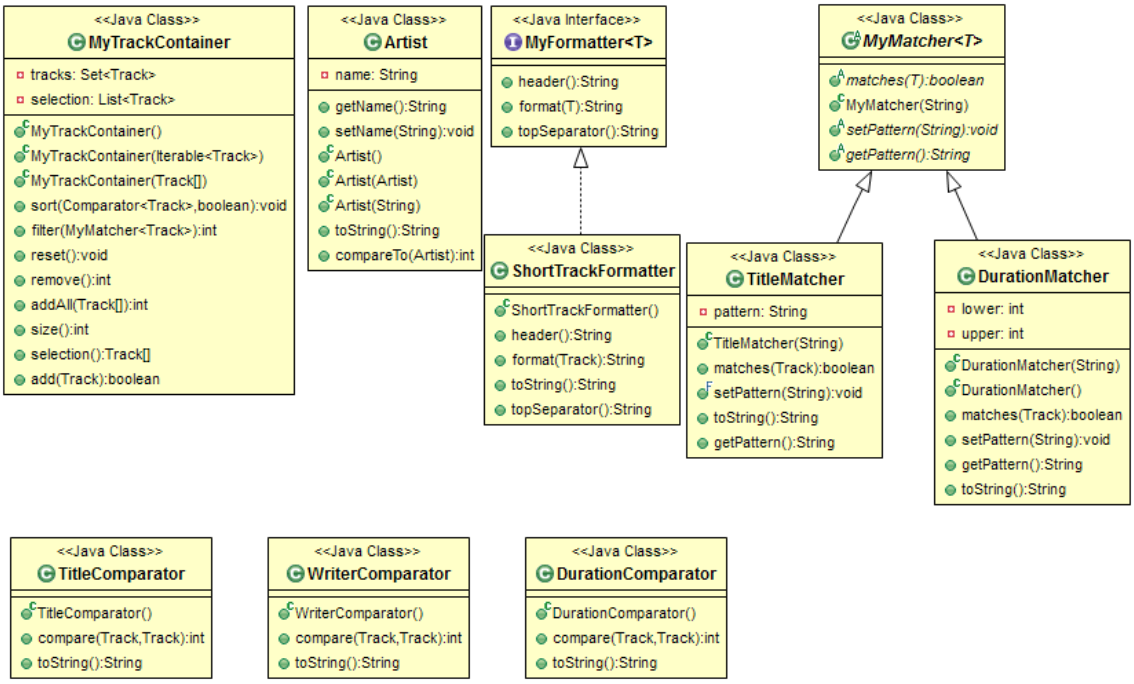

Figure 1: Classdiagram ES05 (only new/ to be changed classes)

> For a detailed description of the fields and methods, please refer to the respective javadoc. Remember: You find new Methods introduced in the actual ExerciseSheet by checking the *Introduced in:* Tag.

Hint

Again, we provide some implemented code as a starting point and support.

The provided finished classes can be found in the ES05_provided.zip and should be imported to your ExerciseSheet.

Information in javadoc

You are asked to implement the Formatters, Comparators, Matchers and Interfaces, the `Comparable<T>` interface in the class `Artist` as well as the `MyTrackContainer`

For some methods/fields/classes, you may find a headline **Hint** which gives you implementation hints.

##Tests## The following amount of tested methods should be shown in your testng overview:

Classes	Methods
Artist	4
DurationComparator	5
DurationMatcher	11
MyTrackContainer	13
ShortTrackFormatter	6
TitleComparator	5
TitleMatcher	8
WriterComparator	5

Classes	Methods
sum	**57**

##ATTENTION##

- Make sure you have implemented all methods and fields with the predefined names!
- If you are not able to implement some of them because of the requirements, generate at least the methods themselves, or else the tests won't compile and therefore the ExerciseSheet won't be graded.
- Never change the testfiles themselves (except the neccessary changes for package name and imports), adjustments in testfiles lead to 0 points of the exercise (as grading will be done with the original ones).
- Please note that 100% passed tests don't mean 100% correct implementation