Audio Player - Adapter Pattern

Renamed MusicPlayer to MediaPlayer for a broader name, so it can inherit more meaning. Wrote a function in AudioPlayer to execute commands with only one parameter delegating to the right object and method. AudioPlayer now can play every kind of adapted music. Changed name of “MusicPlayer” to “MediaPlayer”, same reason as in MusicPlayer to MediaPlayer, metioned earlier.

Now the most important part is the MediaAdapter in this case. It helps us create an object for the fitting function so that we don’t have a type mismatch when delegating calls. The adaptee, AudioPlayer, calls and creates an object of the Adapter class. On creation of the object puts in the type of object it delivers and the adapter will then go on to use the right methods for the delivered object.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Only the driver class is not tested, as it is only used for executing the existing code and has no further functionality or methods. Although, if we run the Driver-class under JUnit coverage, we can achieve 100% also in the Driver class.

Ein Bild, das Text, Screenshot, Anzeigetafel, schwarz enthält.

Automatisch generierte Beschreibung

The line coverage in MediaAdapter is not 100% since the else clauses are already getting covered by AudioPlayer which renders them basically redundant, so the could be removed.

If we did that, the result would look like this:

Ein Bild, das Text, Screenshot, Anzeigetafel enthält.

Automatisch generierte Beschreibung

The only thing remaining uncovered are the empty methods that have to be implemented because of the interface.

Mp4Player and VlcPlayer both have one empty method that they implemented from the interface “AdvancedMediaPlayer”. There is no use in testing an empty method, therefore only 50% of both of these classes are tested.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung