N – Modularization and Utility Classes

Code Samples - Documenation

# Examples

The following examples are used to illustrate this topic.

1. **CD** – This class represents a CD of Songs. It supports a public method to get long songs (which are songs whose RunningTime is greater than the average running time). A private method called resizeArray() is created to “modularize” this public method’s behaviour.
2. **RunningTime** – This class represents an amount of time in minutes and seconds, and is used as the length of time for a Song. This class introduces the Scanner class for one of its constructors, which generates the running time from a string that is formatted as “minutes:seconds” (such as “5:39”, which would be 5 minutes and 30 seconds).  
   This class also uses an internal private method called paddZero() that ensures the seconds whose values are less than 10 have a leading zero when converted to a string.
3. **TextFormatter** – This class provides static methods for common string formatting needs, such as right and left justifying strings.
4. **MusicInfoFormatter** – This class provides static methods for formatting CD information, such as the CD’s title and the information for a given Song. This kind of formatting is commonly seen, for example, in CD players that display the currently playing song.

# CD

This class represents a CD of Songs. It supports a public method to get long songs (which are songs whose RunningTime is greater than the average running time). A private method called resizeArray() is created to “modularize” this public method’s behaviour.

* **getLongSongs()** – Refactor this method so that the code that resizes the array of songs is put into its own private method, called resizeArray.

# RunningTime

This class represents an amount of time in minutes and seconds, and is used as the length of time for a Song. This class introduces the Scanner class for one of its constructors, which generates the running time from a string that is formatted as “minutes:seconds” (such as “5:39”, which would be 5 minutes and 30 seconds).  
This class also uses an internal private method called paddZero() that ensures the seconds whose values are less than 10 have a leading zero when converted to a string.

* **toString()** – Refactor this method to use a new private method called paddZero. The paddZero method will return a string version of the integer value passed in and will ensure that the string has a leading zero character if the supplied value is less than ten.

# TextFormatter

This class provides static methods for common string formatting.

* **leftJustify()** – This static method will append spaces after the supplied text until the supplied text is at least the required length. Note that this method will not trim the supplied text if it is longer than the required length.
* **rightJustify()** – This static method will prepend spaces before the supplied text until the supplied text is at least the required length. Note that this method will not trim the supplied text if it is longer than the required length.
* **shortenWithEllipsis()** – This static method will check the length of the supplied text against the desired length. If the supplied text is longer, it will trim off trailing characters and append an ellipsis (…) , while ensuring that the resulting text is the desired length.

# MusicInfoFormatter

This class provides static methods for formatting CD information, such as the CD’s title and the information for a given Song. This kind of formatting is commonly seen, for example, in CD players that display the currently playing song.

* **formatTitleInfo()** – This method generates a string that has title information for a CD that includes the CDs title and the name of the performing artist. The title name must be shortened with an ellipsis to a maximum length of 30 characters. The generated string must use the following overall format:  
  Title: 'CD Title' (Artist: Performing Artist Name)
* **formatSongInfo()** – This method generates a string that follows specific formatting for a given song. It must use the following format:  
   Track Number: Song Title [Running Time] (by Song Writer)  
  The running time must be in the format of MM:SS.