Music Player

Introduction

Suppose you are a backend developer for a software company developing a backend for a music application. You are tasked to simulate the **add_to_playlist, play_song, next_song, and prev_song** functionalities.

You have decided to implement a queue for the playlist, and a stack for storing the played songs. You'll be writing custom implementations of both data structures using a python list as the base.

A queue is a FIFO data structure, implementing first-in first-out semantics. A stack is a LIFO data structure, implementing last-in first-out semantics.

Housekeeping points

- This is a minimal example and may not follow some standard practices.
- We focus on the main flow, and not much error handling.

Program Organization

The program is structured in the following manner. You are provided with a project which has three files:

1. **Driver.py:** Actual functionality of the player is available here to use. Users will access the driver.py to play the song or navigate around the playlist.
2. **Queue.py:** Backend implementation of Queue to enable the queue like feature in the playlist.
3. **Stack.py**: Backend implementation of Stack to enable the stack like feature in the playlist.

You are provided with the complete implementation of the driver and the scaffolding for the Queue and Stack modules. All the class methods to be implemented are marked as such and the comments in the code explain what you should implement for correct behaviour.

You can see the final expected output of the program in the **'expected_output.txt'** file. This should be what you get, once you have implemented the methods correctly.

Problem Statement

In this assignment we are going to implement two tasks that consists implementation of various stack and queue functions. Please make changes into stack.py and queue.py file only. The scaffolding code is already given in these files. Look at the code and implement those methods in respective classes.

1. Add code to implement the following methods in stack:
    a. empty()
    b. size()
    c. push()
    d. pop()
    e. The peek() method is already implemented as an example.

2. Add code to implement the following methods in queue:
    a. empty()
    b. size()
    c. enqueue()
    d. dequeue()
    e. rear()
    f. The front() method is already implemented as an example.

Evaluation Rubric

Total Project Points: 100

- Basic compilation without errors (10%) : 10 Points
- Correctness: (10 marks for each method implementation)
  Correctness of implementation
    - Problem statement - point 1 (40%) : 40 Points
    - Problem statement - point 2 (50%) : 50 Points

Program Instructions

1. Download the zipped folder named **M02-W06-Music-Player.zip**, unzip it on your local machine, and save it. Go into the directory named **M02-W06-Music-Player.**
2. Make sure you have Python 3.6 or higher installed. At your command prompt, run:

```
$ python --version
Python 3.7.3
```

   If not installed, install the latest available version of Python 3.

3. Open the unzipped folder, and make sure that you have some code available in the stack.py and queue.py package and driver.py. Once you have added the code based on the given task, you can run the Driver.py to verify if the added code is working successfully or not. Please look at the expected_output.txt file to understand the output.

   ```
   $ python3 Driver.py (On many Linux platforms)
                         OR
   $ python Driver.py     (On Windows platforms)
   ```
   In any case, one of these two commands should work.

4. After running **Driver.py,** you can examine the result. This file will currently run various simple calls that will communicate with different classes and methods in those classes. As you solve the problems, you'll be frequently modifying and running this file. You can comment or modify the initial code as needed.

5. Alternatively, you could install a popular Python **IDE**, such as **PyCharm** or **Visual Studio Code**, and select a command to build the project from there.A helper file/document that has necessary information about how to use PyCharm for the first time will also be available.

6. Once the program is ready to submit, zip the parent folder **M02-W06-Music-Player**, and upload the new zip file as **M02-W06-Music-Player.zip**. It is now ready for evaluation.