Activity 6.5 - Writing Classes III (ToDo List with File I/O)

Overview

In this activity, you will add file I/O functionality to your TaskMaster driver class to allow ToDo lists to be accessed across sessions of TaskMaster. You will also practice using try-catch statements and exceptions.

Instructions

Getting Started

1. Open your `TaskMaster` class from your "Activity`6.4`" project in VS code. You may want to reference `Task`, but you will be working in `ToDoList` and `TaskMaster` in this activity. You will also create text files called test1.csv and test2.csv

`ToDoList` File I/O Requirements

You will add two methods to ToDoList:

- ToDoList(File file) This will be a new constructor in your `ToDoList` class that takes in a File object `file` and populates `taskArrayList` with the data contained in the file specified file format below (name of list, then all tasks as comma separated items).
    - Use a try-catch statement to exit cleanly with an appropriately thrown exception if the file is not readable (e.g. if it does not exist, if the user does not have appropriate permissions, etc)
    - After successfully opening the file, parse the contents of the file and populate `taskArrayList` with the tasks stored in the file.
- saveOut(File file) - This method will take a File object file. It will save the tasks in the arraylist (as a csv file).
    - Use a try-catch statement to exit cleanly and throw an appropriate exception if needed.
    - Remember to look at the file to be sure it is saving how you expect it to!

File Format Requirements

The first line of csv files for ToDoLists should contain the name of the ToDoList.
Each subsequent line of the csv files should have the following format to represent a single task:

```
description,complete,category,priority
```

For example, `test1.csv` could consist of the following:

```
Test List 1
Finish activity 6.5,false,SCHOOL,20
Eat a snack,false,PERSONAL,10
Take a nap,false,PERSONAL,15
```

`TaskMaster` Testing Requirements

In the last activity you tested your constructors and methods of your `ToDoList` class
In this activity, you will now test that functionality again when using the file I/O functionality.

- Create a new `ToDoList` instance using your constructor.
- Add `task1`, `task2`, and `task3` to your list using your `addTask(Task t)` method.
- Add another task to your list using your `addTask(String description)` method.
- Print your `ToDoList` and verify that your `toString` works as expected.
- Call the `saveOut` method and save your `ToDoList` to `test1.csv`. Does the file exist? Does it have the correct tasks in it?
- Write a new test file called `test2.csv` with three tasks in it. Format this file as shown above.
- Call the new `ToDoList` constructor method to open `test2.csv`.
- Print your `ToDoList` and verify that the tasks in `test2.csv` are the tasks in the current `ToDoList`.
- Add a task to `ToDoList` and call the `saveOut` method. Does `test2.csv` reflect the new state of `ToDoList`?

Terminology Identification

In your code add comments identifying examples of the following: attributes, methods, static method, pass-by-reference. These should be identified with an @keyterm tag within the comment.

Code Review

When you are finished with this activity, pair up with a classmate and review each other's code to make sure it meets all the requirements.

Submission

After completing the assignment, use the assignment link in Canvas and follow the submission instructions there. You will upload your .java files and put your reflection in the "Comments" box.

Reflection Requirements

Write a one paragraph reflection describing your experience with this activity. The reflection should also include the name of your code review partner AND something interesting you found in their code. Please review the activity rubric for more details.