Adham Niazi ID:1765

Introduction:

The following write up is a technical breakdown of the code for the java program TM.Java. This program will be shown heavily in detail through code and explanation. This program is made of a single file java program. Such that it will create a log file tracking the details about tasks specified by the user

Theory Of Operation:

The objective of TM.Java is to take in input through the command line containing a limited selection of operations, then output a text file containing the commands that were inputted along with a date and timestamp of it. This is to create a time tracking software that can easily be switched between for managing, well, time. In order to achieve this the program uses command line arguments, stored in an array of String type args. TM.Java will then try to open the log file, and if successfully opened, the program reads the file line by line and processes the commands it sees with the date and time. It then stores the string name of each instruction as well as the instruction information in a Hashmap of type <String, Task>. In order to do this, the program loops until the input file has been completely read, and on each loop it will store each line in several String arrays, breaking them down into each component of a Task. These components are Name, Description, Start Time, End Time, Time Worked, and a boolean to track if it's currently being worked on. Once in string array form, the program will check the input to see if the name of the tasks in the log correspond to tasks that are in the hashmap. In the case of no log existing of the task, it will be created on certain input. The program will determine the instruction type and how it's meant to handle it through a chain of classes. Once the file is completely read and the task objects are created, it reads in the command line prompt. If the prompt is of valid form, it will be either added to an already existing Task object, or it will be created in the TM or TaskManager classes. The program is made up of three classes, these are TM, TaskManager, and Task. Task is the lowest class in hierarchy. It contains the structure of a Task object, and is used to run tasks in Task Manager class. TM class is the one that contains the main function. Each one of these will be gone into more detail throughout the rest of this technical breakdown.

Discussion:

TM.java:

TM.java is a comprehensive single file java program that uses code to update a log file and keep track of the time spent on a task. The program is able to keep track of multiple tasks at the same time. This is through the Task class.

```
Class Task {
      private String name;
      private String description;
      private String size;
      private long startTime;
      private long endTime;
      private long timeWorked;
      private boolean inUse;

      ...
```

The task class encapsulates the attributes associated with individual tasks. The Task class is a blueprint for the Task object. This will be the backbone of the program. Each task as shown above has the attributes ‘name’, ‘description’, ‘size’, ‘startTime’, ‘endTime’, ’timeWorked’, and ‘inUse’. Each of these attributes is crucial in maintaining the functionality of the task tracking application. The name attribute allows for users to refer to specific tasks by any name they desire, so long as it doesn't have spaces in it. This is set with the setName function. The description attribute provides a blank space for users to input information that will compile about the task at hand. This is set with the setDescription function. The size attribute is a T-shirt based simple explanation of the challenge the task will be. The startTime attribute is store as a long in milliseconds in the backend so that calculations of time will be simpler. It is also important to know when the task was started. The endTime is just as important for the same reasons except needing to track when a task has finished to calculate the amount of timeWorked. This attribute is used the least, except it needs to be stored so that it can accumulate when a user starts and stops a task multiple times in succession. Finally the boolean inUse is important to set a flag as to when the task is being worked on or not for error handling.

The start of the program starts by reading in the information from the log file, task_log,txt. This file is where all the tasks that are being worked on will be logged for future reference. As the program reads entries from the log file, it populates a hashmap data structure of type Map<String, Task>, where each task is associated with its unique name. This allows for easier task management during the program's execution. As the program goes on, The log file becomes a repository of actions, such as starting, stopping, describing, and renaming tasks, each started with a timestamp and separated with commas. This allows for simple .split parsing.

Ultimately the important parts of the code deal with TaskManager. The Task class is the backbone of this calls upon. When a task is run through the command line and the java file interprets it.The TaskManager class leverages the Task class which as stated before is the building block for managing individual tasks. TaskManager is an attempt at encapsulating the functionality of the tasks themselves. This includes changing the name, starting the task, stopping the task, describing the task and completing the summaries based upon the input to the summary option type.

I ultimately did not plan this out very well, and it is missing a lot. Unfortunately i ran out of time.