SWE 501 – Introduction to Object-oriented Programming
Assignment 3 Report - ToDoList

Ali Şer Gök
ID: 2022719075
Date: 09.01.2023

ToDoList (class) is a class that allows you to manage a list of tasks, including adding new tasks, deleting tasks, searching for tasks by keyword, and listing all tasks. It has the following features and methods:

There is an ArrayList instance variable called "**tasks**" that stores the tasks in the to-do list.

There is a **ToDoList constructor** method to create a default to-do list. This method creates an empty to-do list.

"**setTasks**" method updates the to-do list. This method is used to update the to-do list and is called when the program is started, reading all items in the "input.txt" file and putting them in a new ArrayList called "newTasks". This ArrayList is passed as a parameter to the "setTasks" method and the "tasks" ArrayList used by the main program is updated.

"**addTask**" method adds a new task to the to-do list. This method writes the "Task" object created with user input in the main section to the "input.txt" file and adds it to the to-do list.

"**removeTask**" method finds and deletes the task the user wants to delete from the to-do list. This method performs the deletion based on the ID number of the task the user wants to delete. To perform this operation, it is necessary to first read all items in the "input.txt" file and add these items to an ArrayList< Task> because the deletion operation must be performed based on the ID number of the object to be deleted. Then, the new, reduced list is written back to the "input.txt" file. The following steps are followed to perform these operations:

1) First, the "input.txt" file is read and all items are added to a String ArrayList.
2) Then, for each item in this ArrayList<String>, the necessary information is extracted to create a Task object and a new object is created using this information.
3) The new object is then added to the "tasks" ArrayList<Task>.
4) The new list with the deleted task is written back to the "input.txt" file.

"**searchTasks**" method searches for tasks matching a keyword in the to-do list. This method searches the "tasks" ArrayList for tasks that contain the search keyword in the task description and returns a list of matching tasks. We use the toLowerCase method to lowercase all the letters in each word in the string arrays we are searching for the keyword and the letters in the keyword. If the words are the same, they will return equal values. This allows us to find the searched word regardless of capitalization.

"**listTasks**" method lists all tasks in the to-do list. This method lists all tasks in the "tasks" ArrayList<Task>.

Task (class) represents a task in a to-do list. Each task has a unique ID, a priority level, and a description.

“**taskCounter**” variable is used to automatically assign a unique ID to each new task. It is a static variable.

There are two constructors in this class. The first constructor creates an empty task.

The second **Task constructor** creates a Task object by taking the task description and task priority level of input. Additionally, within this constructive method, the “taskID” is equal to the static value of “taskCounter.” This ensures that each new task definition has a unique, previously unassigned “TaskID”.

“**setTaskCounter**” method updates the taskCounter value every time the list is read and written again. This way, when a new task is added, the task ID does not start from 100, but rather is updated based on the ID number at the end of the list.

“**setTaskID**” method allows each new object created when “input.txt” is read to take its original Task ID number.

“**getTaskID**” method is a getter for the task ID and returns the integer value of the task ID.

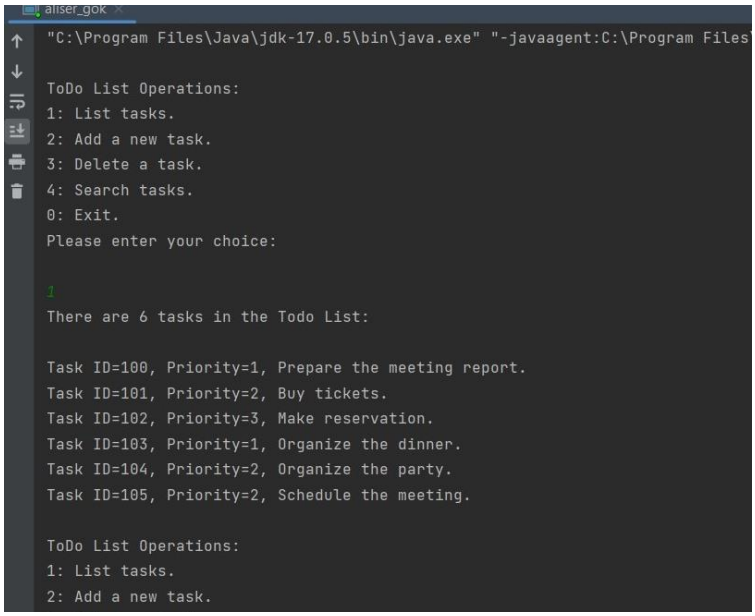
“**toString**” method returns a string representation of all information about a Task object, including the task ID, priority, and explanation.

How does your application remember the tasks between successive runs?

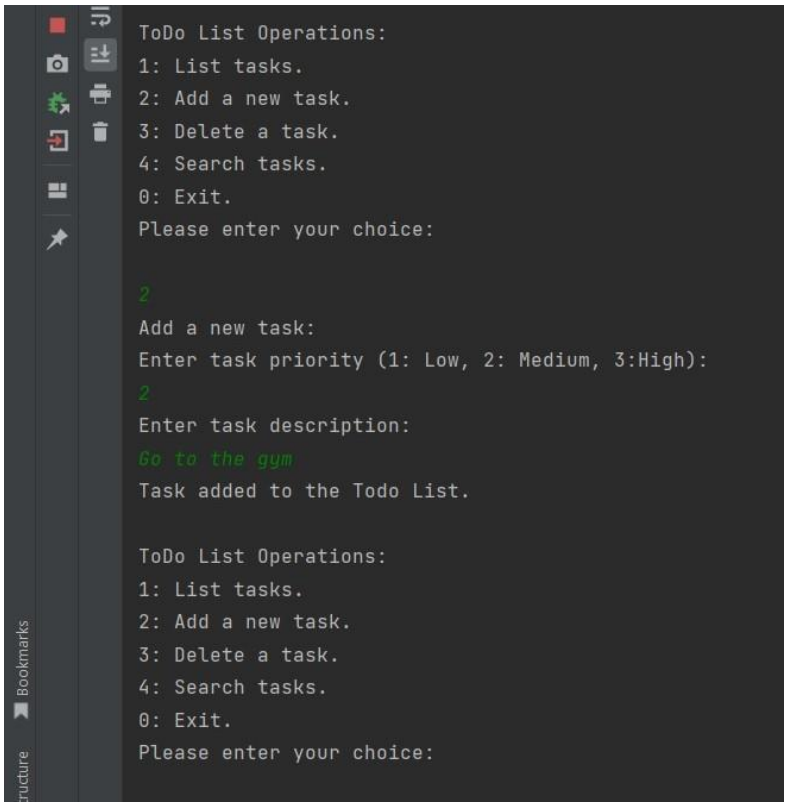
Main (class) reads task data from an "input.txt" file and converts it into Task objects. It also has a menu-based user interface that allows the user to list their tasks, add new tasks, delete existing tasks, search tasks, and exit the program. The program first reads all elements in the "input.txt" file and stores them in an ArrayList<String> called "list1". It then creates a Task object for each element in "list1" and adds these Task objects to another ArrayList<Task> called "newTasks". The ToDoList class is then used to set the tasks in "newTasks" and perform various operations on them, such as adding, deleting, searching, and listing tasks. If the "input.txt" file is empty, the program skips the process of creating Task objects and goes straight to the menu-based user interface. The user can then perform various operations on the tasks, and these changes will be recorded in the "input.txt" file when the program is closed.

Program Outputs

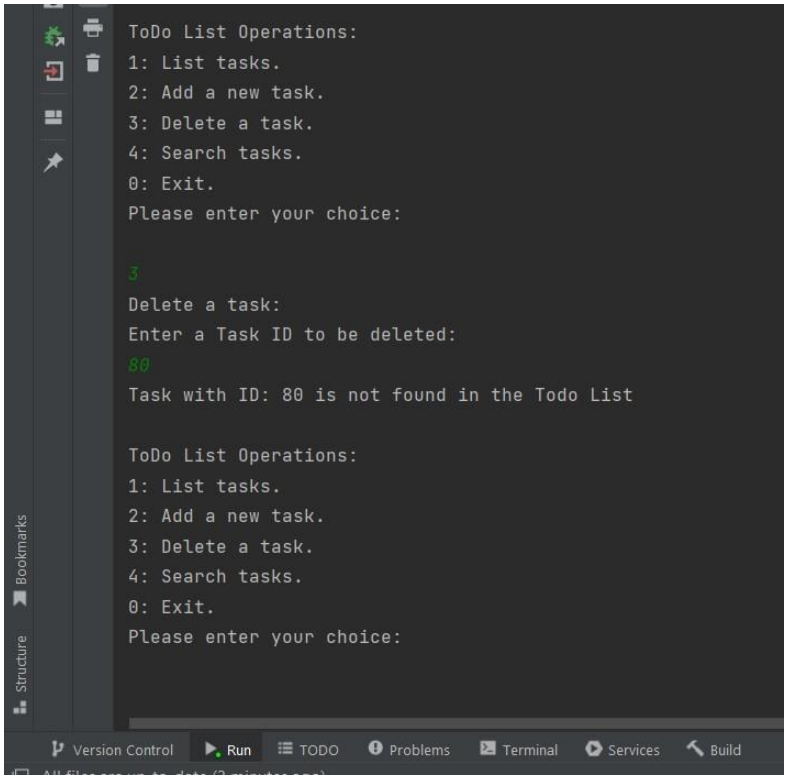
1. List tasks.



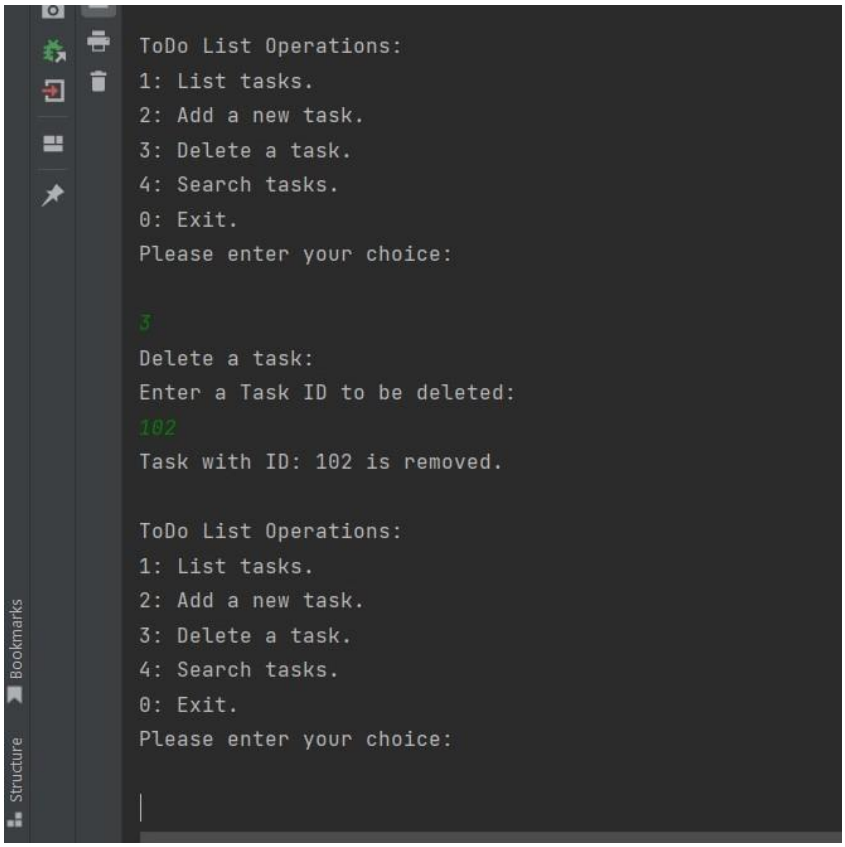
2. Add a new task.



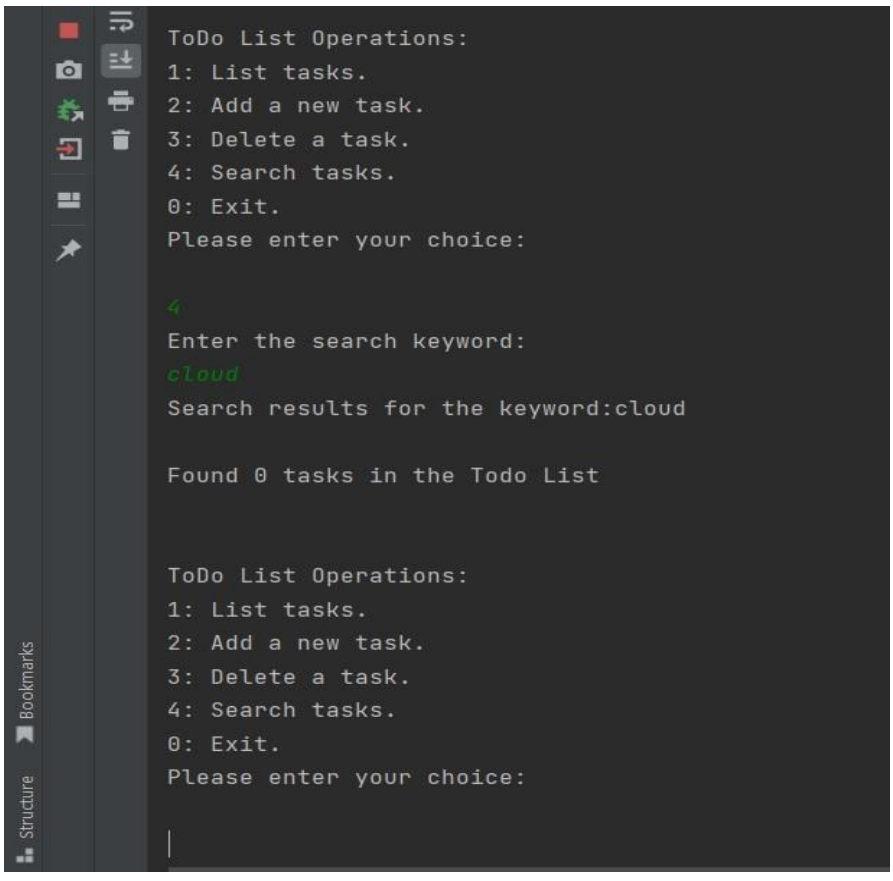
3. Attempt to remove a task with an invalid task ID.



4. Attempt to remove an existing task.



5. Search for a non-existing keyword



6. Search for an existing keyword