Extra Assignment: To-do List Application

Due: January 16th, 2023 (6am) **SWE 501**

In this assignment, you implement a simple to-do manager application. User can be able to list tasks in his/her to-do list, add a new task, remove an existing task and search for a keyword in task descriptions. A task has a unique task ID, a priority value, and a task description. When a new task is added to the to-do list, its unique ID should be automatically generated. Task IDs start with 100.

When the program starts, you should display a menu to the user to choose possible actions, as shown below:

```
ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 1
```

User can be able to perform these operations using the numbered menu options. See the sample program output for a typical execution of the to-do list application. Initally, add few tasks programmatically with your Java code before the user starts to add new tasks, i.e., program should not start with zero tasks. See the sample program for few task examples. Feel free to choose your own tasks.

Sample Program Usage

```
ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 1

There are 6 tasks in the Todo List:
Task ID=100, Priority=1, Prepare the meeting report.
Task ID=101, Priority=2, Buy tickets.
Task ID=102, Priority=3, Make reservation.
Task ID=103, Priority=1, Organize the dinner.
Task ID=104, Priority=2, Organize the party.
Task ID=105, Priority=2, Schedule the meeting.


ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 2

Add a new task:
Enter task priority (1: Low, 2: Medium, 3:High):
3
Enter task description:
Buy a coat.
Task added to the Todo List.

ToDo List Operations:
1: List tasks.
```

```
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 3

Delete a task:
Enter a Task ID to be deleted: 99
Task with ID: 99 is not found in the Todo List


ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 3

Delete a task:
Enter a Task ID to be deleted: 101
Task with ID: 101 is removed.


ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 1

There are 6 tasks in the Todo List:
Task ID=100, Priority=1, Prepare the meeting report.
Task ID=102, Priority=3, Make reservation.
Task ID=103, Priority=1, Organize the dinner.
Task ID=104, Priority=2, Organize the party.
Task ID=105, Priority=2, Schedule the meeting.
Task ID=106, Priority=3, Buy a coat.


ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 4

Enter the search keyword: Plan
Search results for the keyword: Plan
Found 0 tasks in the Todo List


ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 4

Enter the search keyword: organize
Search results for the keyword: organize
Task ID=103, Priority=1, Organize the dinner.
Task ID=104, Priority=2, Organize the party.
Found 2 tasks in the Todo List
```

```
ToDo List Operations:
1: List tasks.
2: Add a new task.
3: Delete a task.
4: Search tasks.
0: Exit.
Please enter your choice: 0

Bye.
```

Below, you are given UML class diagrams for the Task and ToDoList classes. Your implementation should use these classes to perform the required operations.

Task	
-taskCounter: int	Task counter starts with 100 and is incremented automatically to generate a unique task ID for new tasks.
-taskID: int	Unique task ID for a task.
-priority: int	Priority level of a task: 1: Low, 2:Medium, 3:High.
-explanation: String	Description of a task.
Task(priority: int, explanation: String)	Constructor
toString(): String	Generates a string representation of a task
Getter and setter methods (if needed)	

ToDoList	
-taskList: ArrayList<Task>	Array list to store tasks in a to-do list.
ToDoList()	Default constructor
addTask(newTask: Task)	Adds a new task to the to-do list.
removeTask(taskID: int)	Removes the task with a given task ID. If a task with the given task ID is not available in the to-do list, outputs a warning message.
listTasks(): void	Displays all tasks in the to-do list.
search(keyword: String): void	Searches the task descriptions for a keyword. If the keyword is contained in a task description, it displays the task information. Search is case-insensitive. See the sample program output for more details about the output format.

Bonus [20 pts]
In its current form, application forgets any modifications made to the tasks if run multiple times. When you execute the program again, it starts with the same tasks since it does not store the newly added tasks (or removes the deleted tasks). As a bonus, modify the to-do list manager application so that it stores the modified to-do list after exiting the application. So, when you execute the program again, it remembers added/removed tasks from the previous run. Hint: Use text files ☺

Report Contents
In your report, provide a sample program output which contains all the cases given below:

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

Additionally, briefly explain the algorithms for the following cases:

1. How do you automatically determine a new task IDs?
2. How do you search for a (case-insensitive) keyword?
3. How do you remove a task?
4. If you attempt the bonus, explain your solution for the bonus: How does your application remember the tasks between successive runs?

Evaluation Criteria

Code (Grade weight: 80%)

Correctness of the solution
Compliance to submission rules and programming style, e.g., naming conventions, indentation, comments.

Report (Grade weight: 20%)

Completeness of the report, compliance to the report format, correctness of the content and language.

Submission Guide

Submission Files

Submit a single compressed (.zip) file to the Moodle.
Name your zip file as name_surname.zip.
Zip file should contain all source codes (under the \code directory), and report (in PDF format, under the \report directory).
Name the main code which is used to run your assignment as name_surname.java.
Name your report as name_surname.pdf.
Contents of each Java file should start with your name, student ID, date, and a brief code summary in a Javadoc style comment block.

Mandatory Submission

Submission of assignments is mandatory. If you do not submit an assignment, you will fail the course.

Late Submission Policy

Maximum submission delay is two days. Late submission will be graded on a scale of 50% of the original grade.
Submission is mandatory even if you submit your assignment late.

Plagiarism

Plagiarism leads to grade F and YÖK regulations will be applied