Problem Description: To-Do List Manager

You are required to design and implement an object-oriented To-Do List Manager that helps users manage their tasks effectively. The system should allow users to perform various operations on tasks, such as adding new tasks, updating existing ones, removing tasks, marking tasks as completed, and viewing tasks based on different criteria. The design should adhere to object-oriented programming (OOP) principles and be implemented in a language of your choice.

Core Components

1. **Task**
    - **Attributes:**
        - **Task ID:** A unique identifier for each task.
        - **Title:** A short description or name of the task.
        - **Description:** Detailed information about the task.
        - **Due Date:** The deadline by which the task should be completed.
        - **Priority:** The importance level of the task (for example, "High", "Medium", "Low").
        - **Status:** A flag indicating whether the task is completed or still pending.
    - **Behaviors/Methods:**
        - **Mark as Completed:** Change the status of a task to indicate it has been finished.
        - **Update Task:** Modify the title, description, due date, and priority of a task.
        - **Display Task:** Return or print a formatted representation of the task details (e.g., showing Task ID, title, due date, priority, and status).
2. **To-Do List**
    - **Attributes:**
        - **Tasks:** A collection (such as a list or array) that holds multiple tasks.
    - **Behaviors/Methods:**
        - **Add Task:** Create a new task with the provided details and add it to the task collection.
        - **Remove Task:** Delete a task from the collection based on its Task ID.
        - **Mark Task as Completed:** Update the status of a specific task (identified by Task ID) to completed.
        - **Update Task:** Modify the details of an existing task using its Task ID.

- **Retrieve All Tasks:** Display all tasks in the collection.
- **Retrieve Completed Tasks:** Display only the tasks that have been marked as completed.
- **Retrieve Pending Tasks:** Display only the tasks that are not yet completed.
- **Search Tasks:** Find and display tasks that match a given keyword (for example, by searching within the task titles).
- **Sort Tasks:** Sort the tasks based on certain criteria:
  - **By Due Date:** Sort tasks in order of their due dates.
  - **By Priority:** Sort tasks based on their priority (you can define a mapping for priority levels, e.g., "High" as highest priority).

Functional Requirements

1. **Task Management:**
   - **Add a Task:** The system should allow adding a new task with all its attributes.
   - **Remove a Task:** The system should allow removal of a task using its unique Task ID.
   - **Update a Task:** The system should allow updating the title, description, due date, and priority of an existing task.
   - **Mark as Completed:** The system should allow marking a specific task as completed.
2. **Task Retrieval:**
   - **Get All Tasks:** The system should display all tasks.
   - **Get Completed Tasks:** The system should display only tasks that are marked as completed.
   - **Get Pending Tasks:** The system should display only tasks that are still pending.
3. **Additional Operations:**
   - **Search Tasks:** Allow users to search for tasks by providing a keyword. The search should match the keyword against the task titles.
   - **Sort Tasks:** Allow users to sort tasks by either due date or priority.

Input Format

The system will receive commands (or method calls) that instruct it to perform the above operations. Each command will include the necessary parameters. The commands may include (but are not limited to):

- **add_task(taskId, title, description, dueDate, priority)**
- **remove_task(taskId)**
- **mark_completed(taskId)**
- **update_task(taskId, title, description, dueDate, priority)**
- **get_all_tasks()**
- **get_completed_tasks()**
- **get_pending_tasks()**
- **search_tasks(keyword)**
- **sort_tasks(criteria)**
  (where criteria can be "due_date" or "priority")

Note: The exact format of input commands can be adapted based on your implementation environment. The primary focus is on designing appropriate classes, methods, and interactions between them.

Output Format

For each command that retrieves or displays tasks (such as `get_all_tasks`, `get_completed_tasks`, `get_pending_tasks`, `search_tasks`, and `sort_tasks`), output the list of tasks in a human-readable format. Each task should be formatted to include its ID, status (e.g., "Completed" or "Pending"), title, due date, and priority.

Example Scenario

1. **Adding Tasks:**
   - Add a task with ID `1`, title `"Submit Assignment"`, description `"Submit the final assignment for OOP course"`, due date `"2025-03-01"`, and priority `"High"`.
   - Add another task with ID `2`, title `"Grocery Shopping"`, description `"Buy vegetables and fruits"`, due date `"2025-02-05"`, and priority `"Medium"`.
2. **Updating a Task:**
   - Update task `2` to change its title to `"Grocery and Supplies Shopping"`.
3. **Marking a Task as Completed:**
   - Mark task `1` as completed.
4. **Retrieving Tasks:**

- Retrieve and display all tasks.
- Retrieve and display only completed tasks.
- Retrieve and display only pending tasks.

5. **Searching and Sorting:**
   - Search for tasks with the keyword `"shop"`.
   - Sort tasks by due date and then by priority.