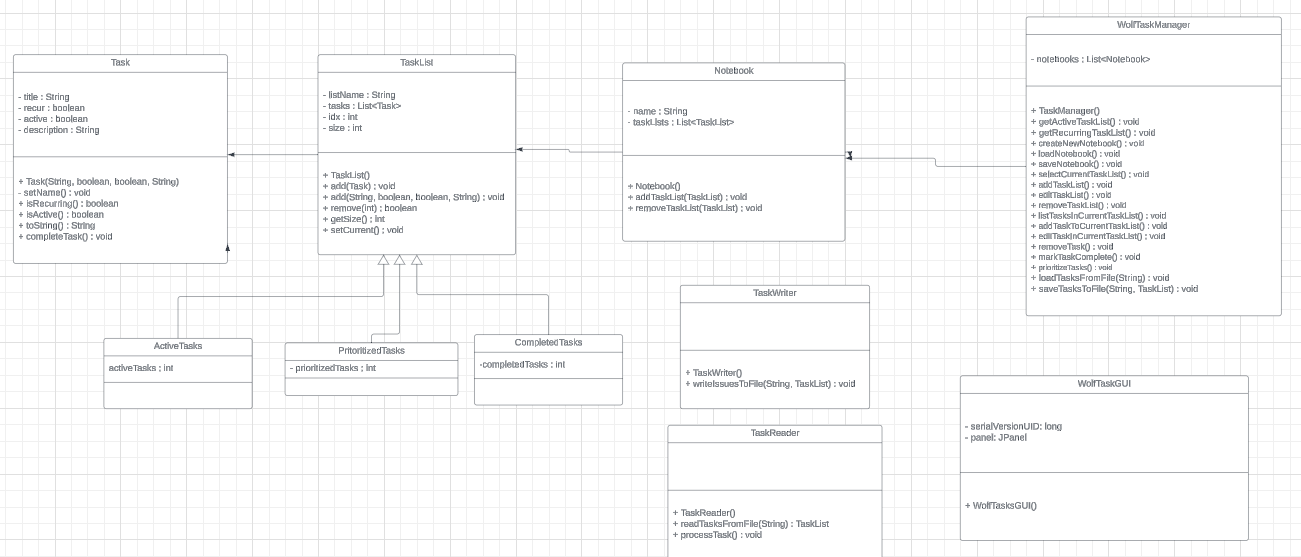
WolfTasks Design Document

Name: Brendan Viscount

Date: 3/18/2024

**UML Diagram:**

**­­­­**

**\*Zoom 200% for UML, some Getters and Setters were excluded for brevity\***

**Design Rationale:**

1. **Importance of Objects**:
   * The key objects in the system's implementation include TaskList, Task, Notebook, and WolfTasks. TaskList represents a collection of tasks, Task represents an individual task, Notebook manages multiple TaskLists, and WolfTasks serves as the main controller for interactions between the user interface and the underlying data. These objects are important as they encapsulate the core pieces and functionalities required for managing tasks and notebooks with WolfTasks.
2. **Required Data**:
   * The data required to implement the system include task names, descriptions, statuses and priority. These data elements are necessary for representing tasks effectively and tracking their state and progress. Additionally, notebook names and collections of task lists are needed to organize and manage tasks.
3. **Appropriate Responsibilities**:
   * the TaskList object is responsible for managing a collection of tasks, while the Task object is responsible for representing individual tasks with properties and behaviors. Similarly, the Notebook object is responsible for managing collections of task lists and facilitating operations on them.
4. **Relationships Between Objects**:
   * Inheritance and composition relationships are utilized to model the relationships between objects effectively. For instance, TaskList and Notebook classes demonstrate composition relationships, where a Notebook contains multiple TaskLists, and each TaskList contains multiple Tasks. These relationships are crucial for organizing and structuring the system's components in a hierarchy, facilitating code reuse and maintainability.
5. **Design Patterns**:
   * The Model-View-Controller design pattern is employed to separate data management, user interface, and the application logic. The WolfTasks class acts as the controller, managing interactions between the user interface and the underlying data model, represented by TaskList, Task, and Notebook classes.
6. **Limitations and Future Additions**:
   * One limitation of the design may be its representation of task lists and tasks, which may not accommodate dynamic changes in structures or new features in the future. Additionally, the design may need to support collaborative management across multiple users.