Listing 1: TodoManager.java

```java
import java.time.LocalDate;

public class TodoManager {
    private TodoNode head;
    private int size = 0;

    public TodoManager() {
        this.head = null;
    }

    public int getSize() {
        return size;
    }

    public void addTodo(String description, LocalDate dueDate) {
        Todo newTodo = new Todo(description, dueDate);
        if (head == null || head.todo.getDueDate().isAfter(dueDate)) {
            TodoNode newNode = new TodoNode(newTodo);
            newNode.next = head;
            head = newNode;
        } else {
            TodoNode current = head;
            while (current.next != null &&
                current.next.todo.getDueDate().isBefore(dueDate)) {
                current = current.next;
            }
            TodoNode newNode = new TodoNode(newTodo);
            newNode.next = current.next;
            current.next = newNode;
        }
    }

    public Todo findTodoById(int id) {
        TodoNode current = head;
        while (current != null) {
            if(current.todo.getId() == id) {
                return current.todo;
            }
            current = current.next;
        }
        return null;
    }
    public int countTodos(LocalDate until, Status status) {
        int count = 0;
        boolean meetsCondition = false;
        TodoNode current = head;
        while (current != null) {
            if((status == null || current.todo.getStatus() == status) && (until ==
                null || !current.todo.getDueDate().isAfter(until))){

            }
            if (meetsCondition) {
                count++;
            }
            current = current.next;
        }
        return count;
    }

    public Todo[] getAllTodos() {
        int size = countTodos(null, null);
        Todo[] todos = new Todo[size];
        int i = 0;
```

```
        TodoNode current = head;
        while (current != null) {
            todos[i] = current.todo;
            i++;
            current = current.next;
        }
        return todos;
    }

    public Todo[] getTodosUntil(LocalDate until) {
        int size = countTodos(until, null);
        Todo[] todos = new Todo[size];
        int i = 0;
        TodoNode current = head;
        while (current != null) {
            if (!current.todo.getDueDate().isAfter(until)) {
                todos[i] = current.todo;
                i++;
            }
            current = current.next;
        }
        return todos;
    }

    public Todo[] getOpenTodos() {
        int size = countTodos(null, Status.OPEN);
        Todo[] todos = new Todo[size];
        int i = 0;
        TodoNode current = head;
        while (current != null) {
            if (current.todo.getStatus() == Status.OPEN) {
                todos[i++] = current.todo;
            }
            current = current.next;
        }
        return todos;
    }

    public Todo[] getOpenTodosUntil(LocalDate until) {
        int size = countTodos(until, Status.OPEN);
        Todo[] result = new Todo[size];
        int i = 0;
        TodoNode current = head;
        while (current != null) {
            if (current.todo.getStatus() == Status.OPEN &&
                !current.todo.getDueDate().isAfter(until)) {
                result[i] = current.todo;
                i++;
            }
            current = current.next;
        }
        return result;
    }

    public Todo[] getDoneTodos() {
        int size = countTodos(null, Status.DONE);
        Todo[] todos = new Todo[size];
        int i = 0;
        TodoNode current = head;
        while (current != null) {
            if (current.todo.getStatus() == Status.DONE) {
                todos[i++] = current.todo;
            }
            current = current.next;
```

```
        }
        return todos;
    }

    public Todo[] getDoneTodosUntil(LocalDate until) {
        int size = countTodos(until, Status.DONE);
        Todo[] result = new Todo[size];
        int i = 0;
        TodoNode current = head;
        while (current != null) {
            if (current.todo.getStatus() == Status.DONE &&
                !current.todo.getDueDate().isAfter(until)) {
                result[i] = current.todo;
                i++;
            }
            current = current.next;
        }
        return result;
    }









}
```