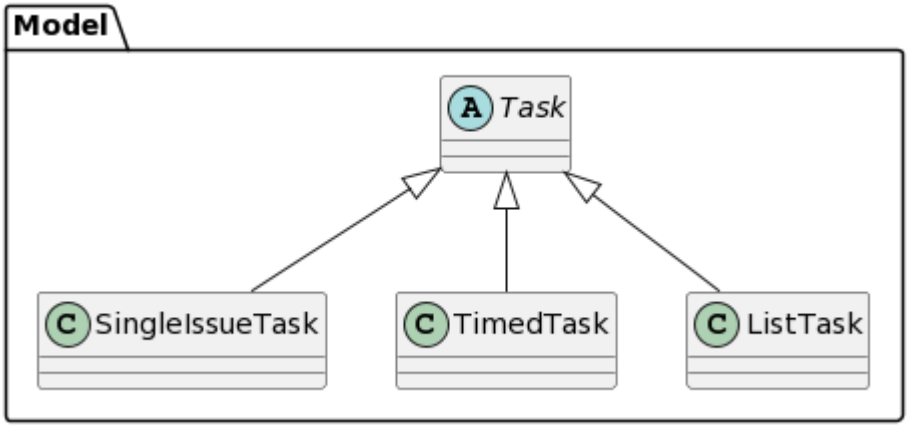
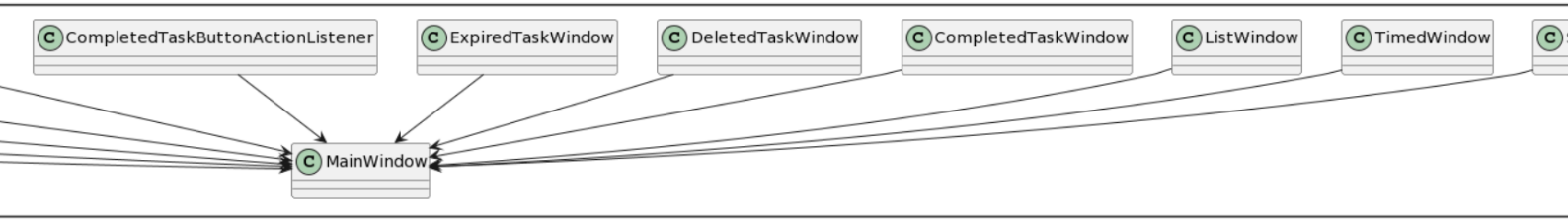
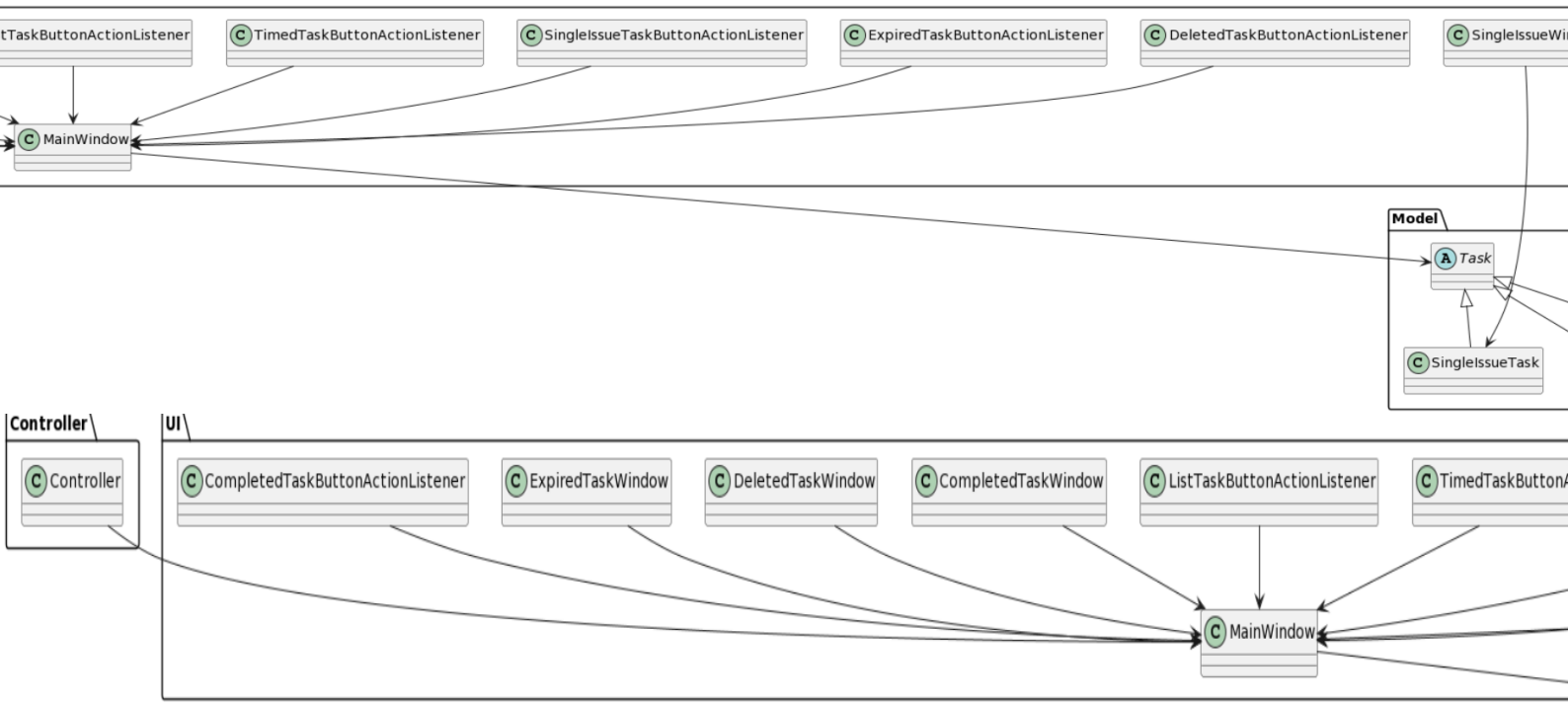
Class Diagrams



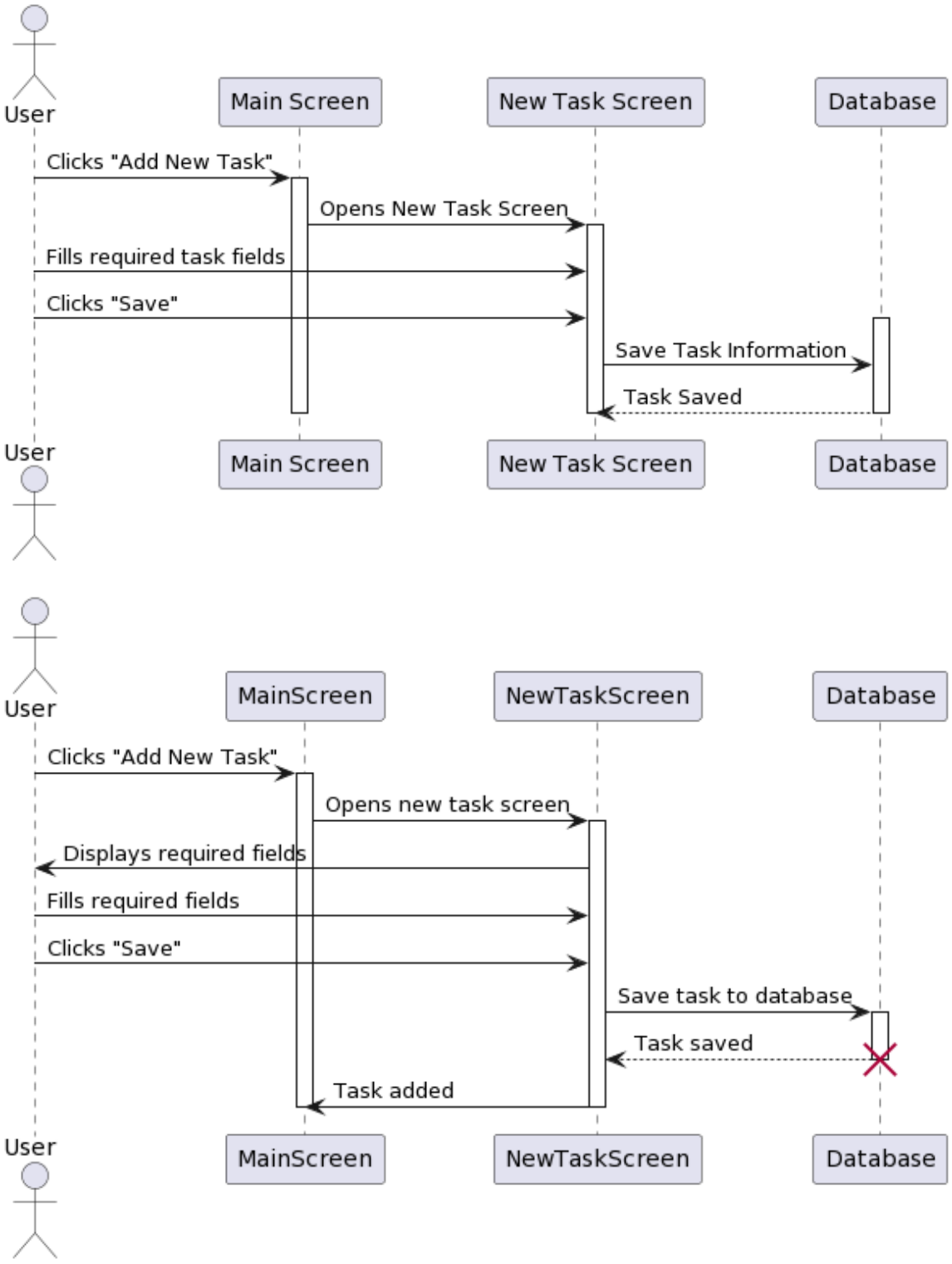




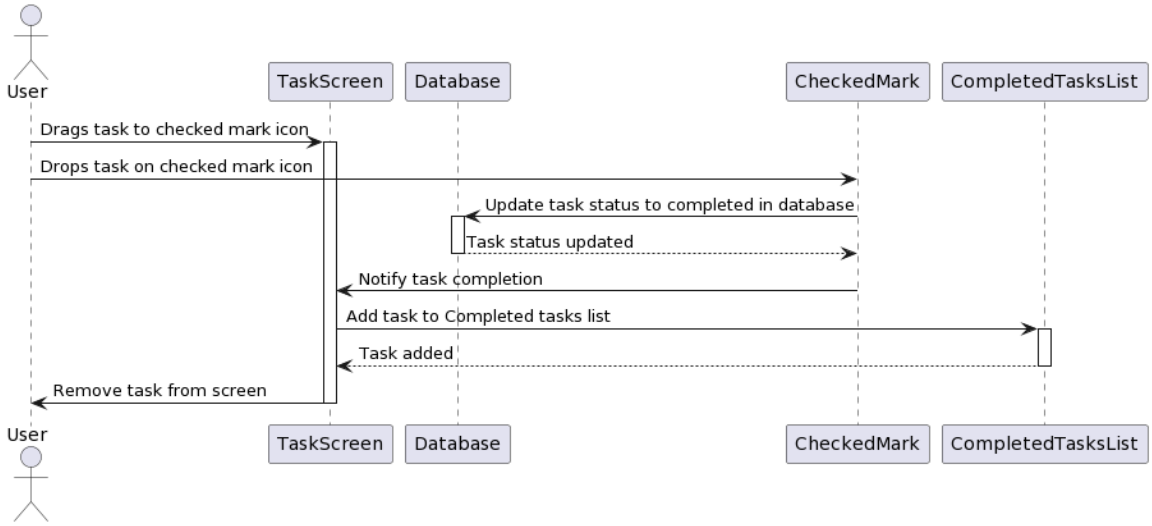
https://www.plantuml.com/plantuml/png/XPB1QWCX48RlFiMymEO9IacRzX1eJatfsUuEGU9gq5bQIFlkPHKoef6vxVx_f_-e_pQI3hJCLbtqUDOdX8EVqC9L0G3ehqX1ZmIZrJ72KSTpqlFlkt4dY_iO5woSevbnghGt4ob9V-gMzR67Qu4Sj76Vnat--tvAwN5EeN9I8-xyVB58UQp6VS6-z_fpCQ7rsbsV5oBldaOotgqpeCDGPOWa9qfaVJa2pSyYSwlKnOhdstb7mLkBeNn6bXES-hHPjuE7ts60_5yDtg6uH_VzMwSKLmU6uH4OLqLI1RHJf3occYdMFBQR8VZW6_xC93TRYPsfLrJ5aV2gFH8kLwXPKlH8XljbakucY-rj202VlmDiqKtBRFy1

Sequence Diagrams:

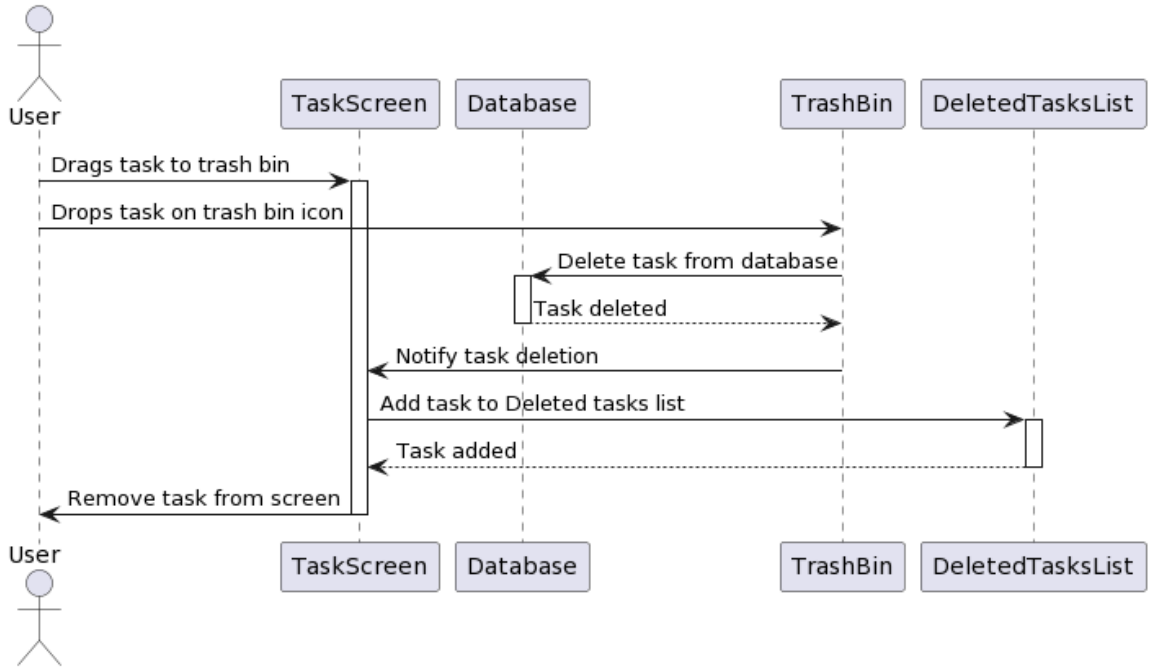
Adding a task



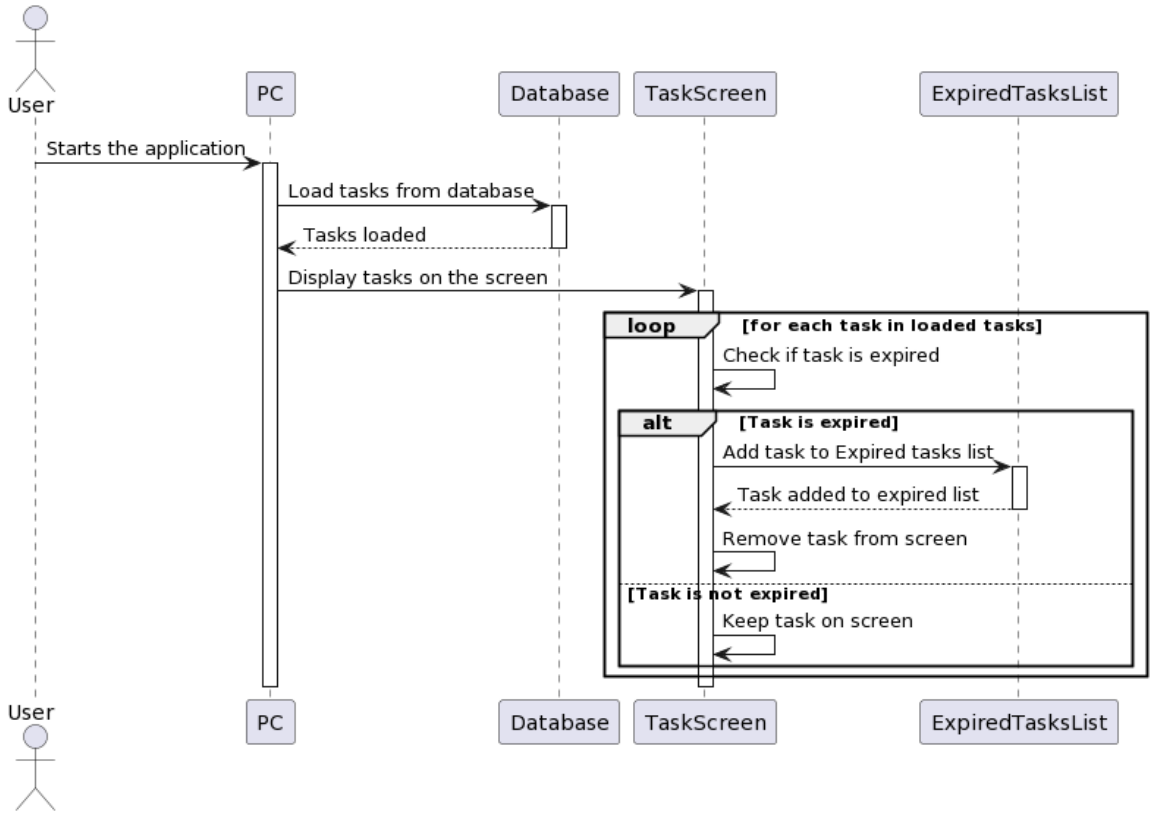
Mark task as completed by dragging it and dropping it to the check mark



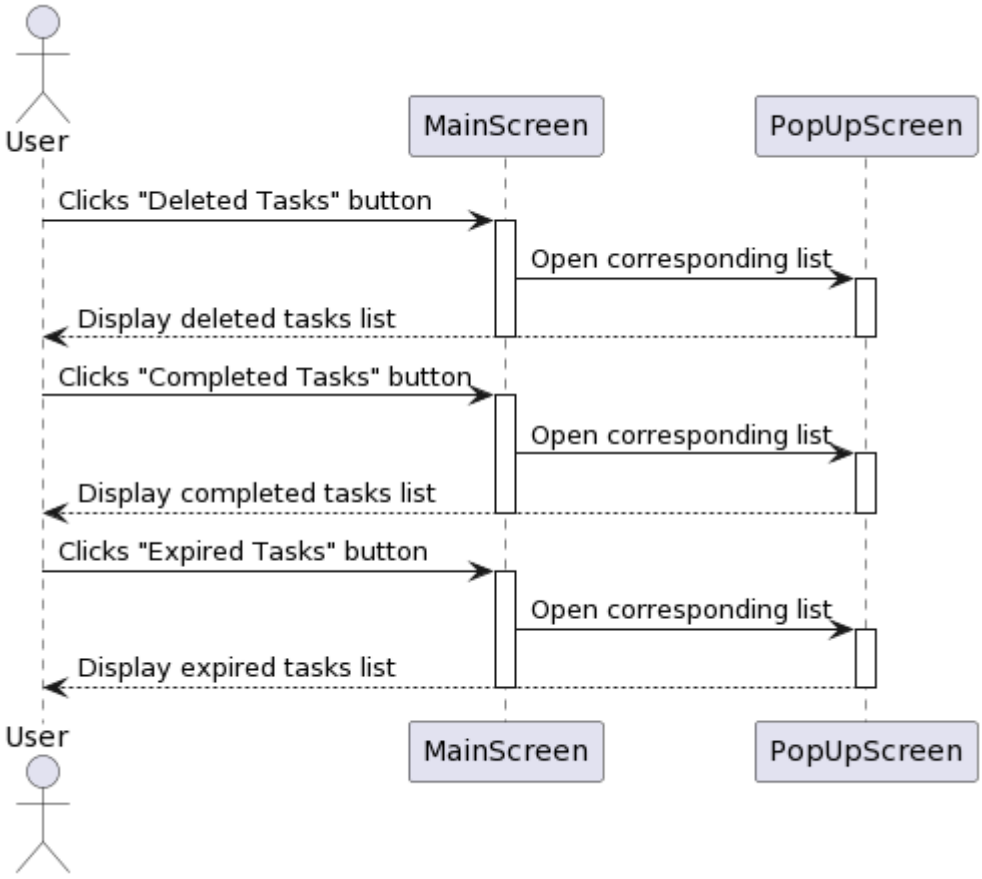
Deleting a task by dragging it and dropping it to the trash bin icon



at every start of the application, the system checks, filters the tasks and checks the expired ones. The expired tasks are deleted from the tasks and added to the expired tasks list.



The user can choose to view any of the lists of the deleted, expired, or completed tasks by clicking on the corresponding button



Test Cases:

TaskClass related test case:

`taskInitialization_ShouldCreateTaskWithGivenParameters()`
- **Objective**: This test verifies that a `Task` object can be properly initialized with the provided parameters.
- **Steps**:
  1. **Given**: Initialize parameters (title, description, dueDate, category).
  2. **When**: Create a new `Task` object using the initialized parameters.
  3. **Then**: Perform assertions to ensure that the created task has the expected values for its attributes (title, description, dueDate, category) and some initial states (not completed, non-null creation date).

`taskMarkAsCompleted_ShouldMarkTaskAsCompleted()`
- **Objective**: This test verifies the behavior of the `markAsCompleted()` method in the `Task` class.
- **Steps**:
  1. **Given**: Create a new `Task` object.
  2. **When**: Check if the task is initially not completed, then call `markAsCompleted()` method.
  3. **Then**: Assert that after calling `markAsCompleted()`, the task's completion status changes to `true`.

`taskToString_ShouldReturnStringRepresentation()`
- **Objective**: This test ensures that the `toString()` method of the `Task` class generates the expected string representation.
- **Steps**:
  1. **Given**: Create a new `Task` object with specific attributes.
  2. **When**: Retrieve the string representation by calling the `toString()` method.
  3. **Then**: Validate that the generated string contains the expected substrings corresponding to different attributes of the task (title, description, dueDate, category).

This test suite helps ensure the correctness of the `Task` class's behavior, aiding in identifying potential issues or regressions when changes are made to the class implementation.

Single Issue Task related test case::

`singleIssueTaskInitialization_ShouldCreateSingleIssueTaskWithGivenParameters()`
- **Objective**: This test verifies the proper initialization of a `SingleIssueTask` object with the provided parameters.
- **Steps**:
  1. **Given**: Initialize parameters (title, description, dueDate).
  2. **When**: Create a new `SingleIssueTask` object using the initialized parameters.
  3. **Then**: Perform assertions to ensure that the created task has the expected values for its attributes (title, description, dueDate), and additional attributes like category (set to "Single Issue"), initial completion state, and non-null creation date.

`singleIssueTaskToString_ShouldReturnStringRepresentation()`
- **Objective**: This test ensures that the `toString()` method of the `SingleIssueTask` class generates the expected string representation.
- **Steps**:
  1. **Given**: Create a new `SingleIssueTask` object with specific attributes.
  2. **When**: Retrieve the string representation by calling the `toString()` method.
  3. **Then**: Validate that the generated string contains the expected substrings corresponding to different attributes of the single issue task (title, description, dueDate, category).

These tests play a crucial role in verifying that the `SingleIssueTask` class functions as expected, helping detect potential issues or regressions and ensuring the integrity of its implementation.

Timed Task related test case::

`timedTaskInitialization_ShouldCreateTimedTaskWithGivenParameters()`
- **Objective**: This test ensures the proper initialization of a `TimedTask` object with the provided parameters.
- **Steps**:
  1. **Given**: Initialize parameters (title, description, dueDate, durationMinutes).
  2. **When**: Create a new `TimedTask` object using the initialized parameters.
  3. **Then**: Perform assertions to ensure that the created task has the expected values for its attributes (title, description, dueDate, durationMinutes), as well as additional attributes like category (set to "Timed Task"), initial completion state, and non-null creation date.

`timedTaskDuration_ShouldSetAndGetDurationMinutes()`
- **Objective**: This test validates the functionality of setting and getting the duration in minutes for a `TimedTask`.
- **Steps**:
  1. **Given**: Create a new `TimedTask` object with a duration of 60 minutes.
  2. **When**: Check the initial duration value using `getDurationMinutes()`, and then set a new duration of 90 minutes using `setDurationMinutes()`.
  3. **Then**: Assert that after the change, the `getDurationMinutes()` method retrieves the updated duration value correctly.

The tests serve a critical role in validating the expected behavior and functionality of the `TimedTask` class, ensuring its reliability, and identifying any potential issues or regressions in the class implementation.

List Class related test case:

`listTaskInitialization_ShouldCreateListTaskWithGivenParameters()`
- **Objective**: This test aims to ensure the proper initialization of a `ListTask` object with the provided parameters.
- **Steps**:
  1. **Given**: Initialize parameters (title, description, dueDate, subTasks).
  2. **When**: Create a new `ListTask` object using the initialized parameters.
  3. **Then**: Perform assertions to ensure that the created task has the expected values for its attributes (title, description, dueDate), a category set to "List Task," initial completion state as false, and a non-null creation date.

This test contributes to ensuring the reliability and correctness of the `ListTask` class initialization, ensuring it functions as intended and helping identify potential issues or regressions related to its initialization process.

Launcher:

`mainWindowCreated()`
- **Objective**: This test checks whether a `MainWindow` object can be successfully created.
- **Steps**:
  1. **When**: Instantiates a new `MainWindow` object.
  2. **Then**: Asserts that the `MainWindow` object created is not null, confirming that the creation of the `MainWindow` instance was successful.

- The unit test focuses solely on ensuring the successful creation of a `MainWindow` object.
- It uses the `assertNotNull()` method from JUnit to verify that the instance of `MainWindow` created in the test is not null.