**GUI:**

In der App **gui.py** wurde eine Klasse **App** definiert, die von der **Tk**-Klasse erbt. Diese Klasse enthält die Methode **\_\_init\_\_()**, die die GUI erstellt und die CAN-Schnittstelle initialisiert. In der Methode **\_\_init\_\_()** wird zunächst das Fenster auf Vollbildmodus gesetzt und zwei Label-Frames erstellt - eins für die Ausgabe von empfangenen Nachrichten und eins für das Senden von Nachrichten. Dann werden ein Button zum Beenden der Anwendung und ein Button zum Senden von Nachrichten erstellt. Darunter befindet sich ein Eingabefeld, in dem der Benutzer eine Nachricht eingeben kann. Das Feld wird validiert, um sicherzustellen, dass nur maximal 8 Zeichen eingegeben werden können. Dies ist in der Methode **validate\_input()** definiert. Diese Methode wird während der Erstellung des Eingabefelds über die **validatecommand**-Option aufgerufen. Wenn der Benutzer die Eingabetaste drückt, wird die Methode **send\_can\_message()** aufgerufen, um die eingegebene Nachricht zu senden.

**read\_messages()** liest empfangene CAN-Nachrichten von der CAN-Schnittstelle und gibt sie in der GUI aus. Die Methode wird in einem separaten Thread ausgeführt, um die GUI während des Empfangs nicht zu blockieren. Wenn eine Nachricht empfangen wird, wird sie zur Liste **can\_messages** hinzugefügt.

**handle\_messages()** verarbeitet die im Array **can\_messages** gespeicherten Nachrichte und gibt eine modifizierte Version der Nachricht in der GUI aus. Die Methode wird in einem separaten Thread ausgeführt, um **read\_messages()** nicht zu blockieren. Wenn eine Nachricht verarbeitet wurde, wird die modifizierte Nachricht in der **ScrolledText**-Box ausgegeben und aus der Liste **can\_messages** entfernt.

**send\_can\_message()** sendet eine Nachricht, die vom Benutzer im Eingabefeld eingegeben wurde, über die CAN-Schnittstelle. Wenn eine Nachricht gesendet wird, wird eine modifizierte Version der Nachricht in der **ScrolledText**-Box ausgegeben. Den Aufruf habe ich jedoch deaktiviert, weil das nicht gefordert war.

**modify\_message()** formatiert eine Nachricht, um sie in der **ScrolledText**-Box auszugeben. Die Methode fügt der Nachricht einen Zeitstempel hinzu und gibt an, ob die Nachricht gesendet oder empfangen wurde.

**quit()** wird aufgerufen, wenn der Beenden-Button oder die ESC-Taste gedrückt wird. Diese Methode setzt das **running**-Flag auf **False**, um den Thread zu signalisieren, dass er beendet werden soll, und wartet auf das Ende des Threads. Dann wird die CAN-Schnittstelle geschlossen und die Anwendung wird beendet.

**CanInterface:**  
Die Klasse **CanInterface** verwendet die Python-Bibliothek **python-can**, um mit dem CAN-Bus zu kommunizieren. Die Schnittstelle wird durch die Methode **can.interface.Bus()** aus der **python-can**-Bibliothek erstellt.

In der Klasse wird ein **CanInterface**-Objekt mit dem Namen des Bus-Interfaces erstellt und initialisiert. Die Standard-Bitrate ist auf 500.000 Bit/s eingestellt.

Falls beim Erstellen der Busschnittstelle ein Fehler auftritt, wird eine **CanInterfaceError**-Ausnahme mit einer Fehlermeldung ausgelöst. Diese Ausnahme wird auch ausgelöst, wenn beim Senden oder Empfangen von CAN-Nachrichten ein Fehler auftritt.

**send\_message()** es wird eine CAN-Nachricht mit der übergebenen **message** und der **ID** erstellt und über die CAN-Schnittstelle gesendet.

**read\_message()** es wird eine CAN-Nachricht empfangen und zurückgegeben.

**close()** wird verwendet, um die CAN-Schnittstelle sauber zu schließen und alle damit verbundenen Ressourcen freizugeben.