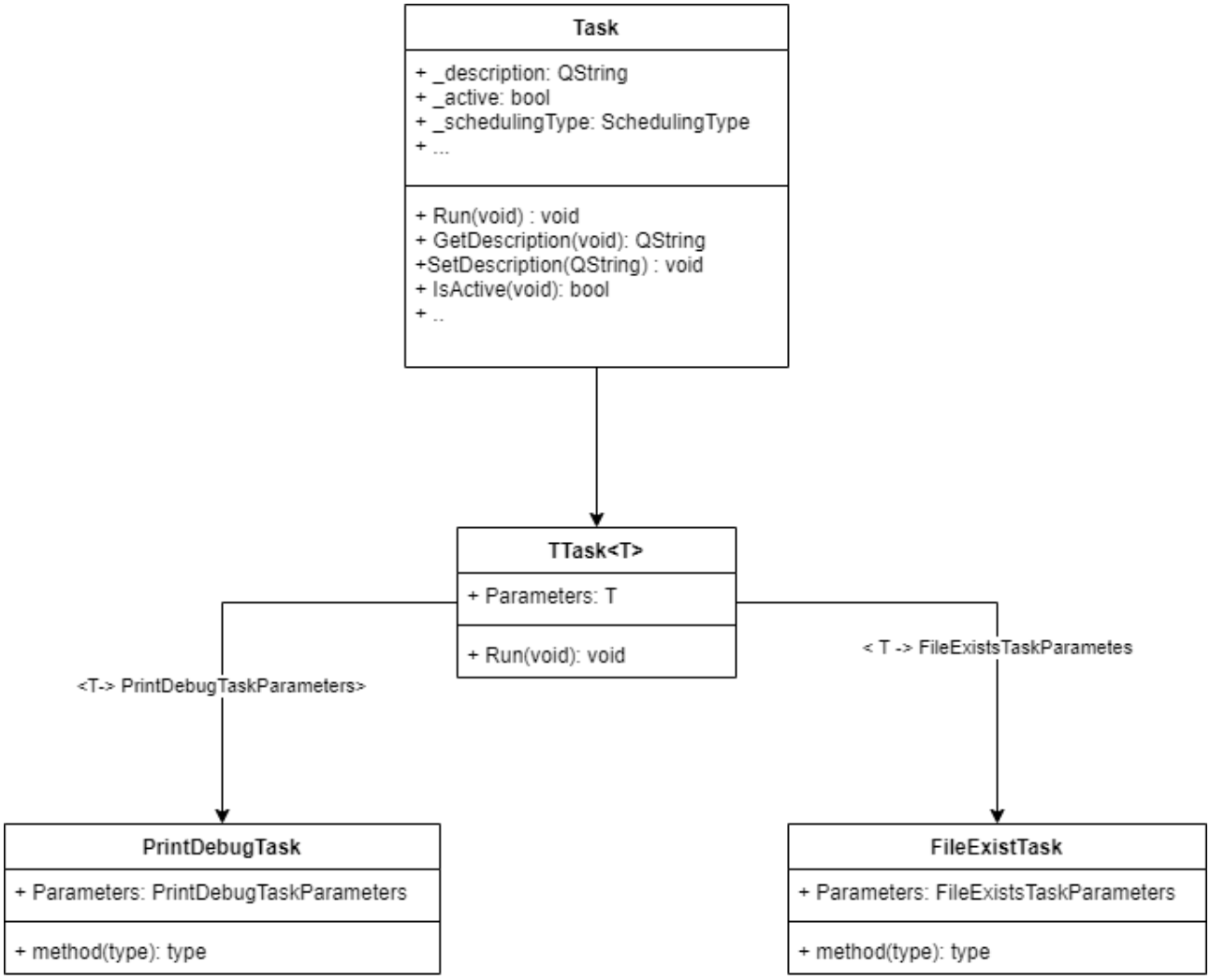
TI-Care Task Scheduling

Gestione dei task

La gestione dei task è basata sulla classe Task. La classe contiene tutto il codice necessario alla schedulazione del task e un metodo virtuale ***Run***() per l'esecuzione del codice specifico per ogni task. La classe Task non deve essere istanziata direttamente.

La schedulazione può essere programmata per tutti i giorni della settimana oppure specificando solo i giorni in cui il task deve essere eseguito. È inoltre possibile specificare se il task deve essere eseguito ad intervalli regolari oppure ad un preciso orario utilizzando l'enum SchedulingType (Periodioc/FixedTime). Nel primo caso è possibile specificare l'intervallo di esecuzione (in secondi) tramite il metodo **SetPeriod**(int period), nel secondo caso si può specificare l'orario di esecuzione tramite il metodo **SetClockTime**(QTime clockTime). L'orario è fisso per tutti i giorni della settimana. Per semplicità non è stata implementata una business logic per garantire la correttezza della configurazione. Se fosse necessario, è possibile disattivare un task temporaneamente utilizzando il metodo **SetActive**(bool active).

Per gestire i parametri necessari all'esecuzione del task, è stata creata un classe template TTask <T> derivata dalla classe Task. Per ogni task che sarà necessario usare nell'applicazione è possibile specializzare la classe TTask con una classe dedicata ai parametri da gestire. Per l'esempio richiesto sono state create le classi PrintDebugTask e FileExistsTask.

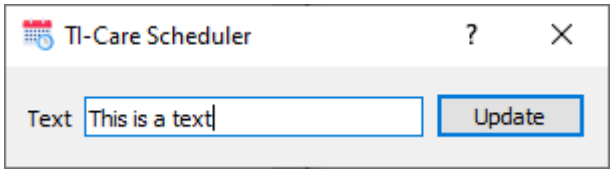


Le due classi implementate sono basate rispettivamente sulle classi parametri PrintDebugTaskParameters e FileExistsTaskParameters. Nell'architettura del software le classi dei parametri sono derivate da una classe base TaskParameters, nel caso serva avere del codice comune a tutte le classi dei parametri. Nonostante le due classi fossero molto simili (solo un attributo di testo) sono state differenziate perché nella realtà si presuppone che i parametri dei vari task siano diversi tra loro.

Interfaccia grafica

Come richiesto dalle specifiche l'interfaccia grafica prevede l'utilizzo della QSystemTrayIcon. La finestra principale è la classe TicScheduler (QDialog). Al caricamento utilizza la classe TaskList per caricare la lista dei task da un file di configurazione o da un database. Nell'esercizio il metodo TaskList::**Deserialize**() è stato implementato con la sola creazione dei due task necessari in modo hard-coded.

Per semplicità non è stato implementato un pattern Model/View o un altro pattern più complesso per la sincronizzazione dei dati tra modello e vista. In un progetto reale sarebbe possibile popolare la finestra principale con, per esempio, un tab per ogni task e declinare l'editor presente nel tab in base alla classe specifica del Task. Utilizzando l'ulteriore livello previsto dal Model sarebbe poi quindi possibile tenere aggiornati tutti i task presenti nella TaskList. L'esempio realizzato prevede solo la gestione del task PrintDebugTask e del suo parametro testuale. Il tasto "Update" permette di aggiornare il testo da stampare nel messaggio di debug richiamando il metodo TicScheduler::**updateTaskList**().



Per l'esecuzione dei task, si è utilizzato un QTimer che scandisce un tick di base per la verifica dei task da eseguire. Questa soluzione semplice sostituisce la creazione di un thread e permette, nonostante i suoi limiti, di non rallentare la risposta dell'interfaccia grafica. Per l'esempio dove la durata dei task non è significativa può essere una buona alternativa ad un thread separato con la gestione dei task in coda FIFO.