Space Station

Eddy Pinarello mat. 2075535

Relazione progetto Programmazione a Oggetti

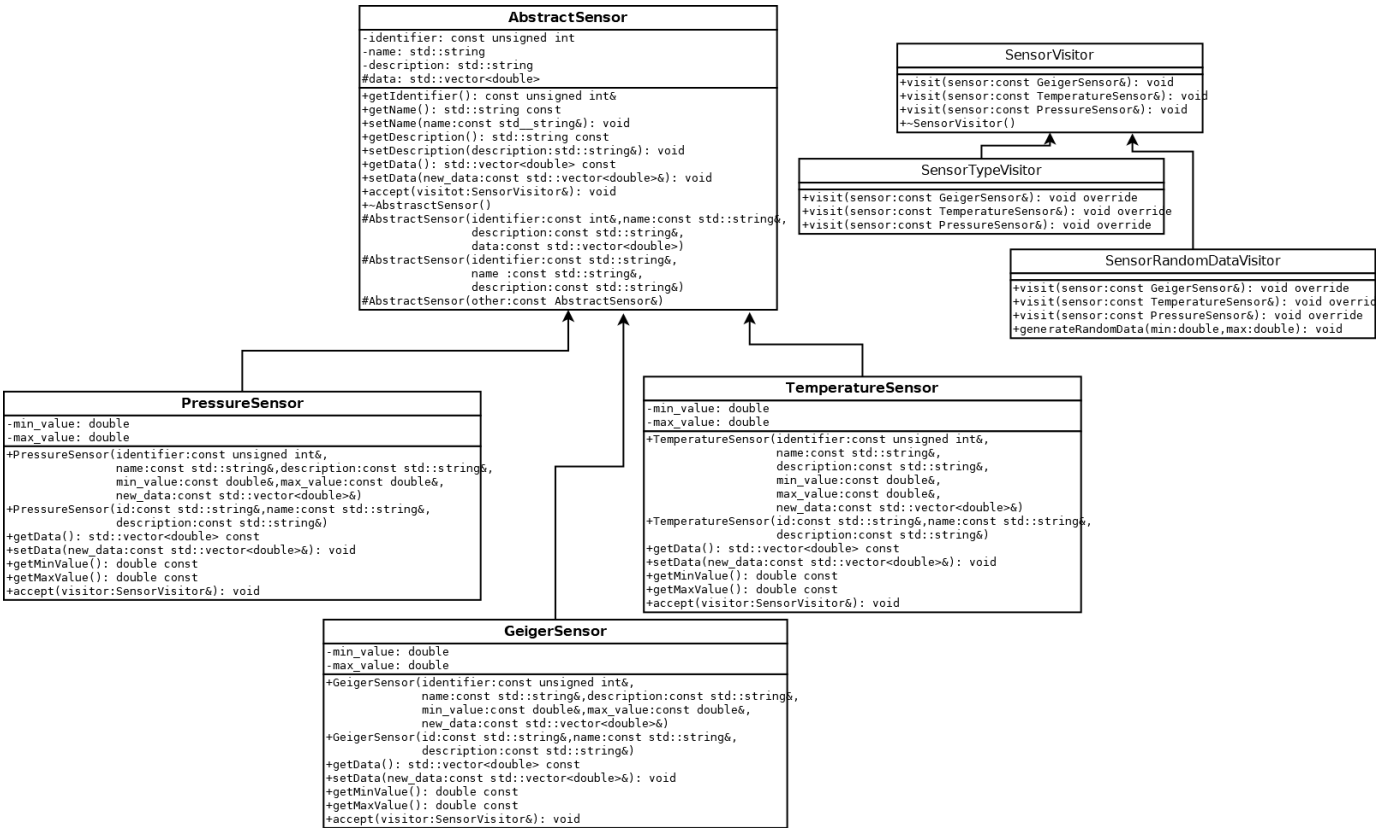
Introduzione

Il progetto prevede un'interfaccia per il test dei sensori di una stazione spaziale in orbita, simulando la raccolta dei dati. La stazione è dotata di un sensore di temperatura per monitorare la temperatura all'interno dell'abitacolo, di un sensore Geiger che misura le radiazioni ionizzanti per tenere sotto controllo la salute degli astronauti e di un sensore di pressione per studiare le pressioni a cui è sottoposta la cabina degli astronauti. Nell'interfaccia, è possibile aggiungere questi sensori tramite il pulsante "ADD SENSOR", dove per poter aggiungere correttamente il sensore bisogna inserire un ID numerico di almeno 5 caratteri, il nome del sensore che non deve essere nullo e la descrizione che può anche essere omessa. Dopo aver aggiunto un sensore all'interfaccia, è disponibile la barra di ricerca per nome tra i sensori, con la particolarità di cercare se contiene la parola inserita nel nome e non solo se comincia con essa. Sono inoltre disponibili i pulsanti MODIFY e DELETE che rispettivamente modificano nome e/o descrizione del sensore ed eliminano il sensore selezionato. È presente un'altra sezione per la simulazione dei sensori: dopo aver selezionato il sensore dalla lista e premendo su SIMULATE, si potrà generare un vettore casuale di rilevazioni e visualizzarlo come grafico in una porzione apposita dell'applicazione. Sono inoltre disponibili i pulsanti Open e Save che servono a salvare ed importare ID, Nome e Descrizione della nostra lista di sensori.

Descrizione modello

Il modello di interfaccia per i sensori della stazione spaziale inizia con una classe astratta denominata AbstractSensor, la quale racchiude le informazioni comuni a tutti i sensori. Queste informazioni includono un identificatore univoco, un nome e una descrizione. La classe AbstractSensor fornisce metodi getter e setter per l'accesso e la modifica di questi attributi, oltre a metodi per gestire i dati del sensore e per accettare visitatori definiti dalla classe SensorVisitor.
Le classi concrete PressureSensor, TemperatureSensor e GeigerSensor derivano dalla classe astratta AbstractSensor e rappresentano tipi

specifici di sensori. Oltre agli attributi ereditati, queste classi includono proprietà aggiuntive, come min_value e max_value, che indicano i valori minimo e massimo registrabili dai rispettivi sensori. Questi sensori implementano anche metodi specifici per ottenere e impostare i dati, nonché per accettare visitatori specializzati.
Il pattern visitor è implementato attraverso la classe SensorVisitor e le sue sottoclassi, come SensorTypeVisitor e SensorRandomDataVisitor, le quali forniscono metodi per visitare i vari tipi di sensori. Questi visitatori consentono di eseguire operazioni specifiche su ciascun tipo di sensore senza modificarne la struttura interna.



Polimorfismo

L'uso del polimorfismo non banale è stato implementato tramite il pattern visitor attraverso le classi SensorTypeVisitor e SensorRandomDataVisitor. Queste sono entrambe classi concrete derivate da SensorVisitor. La prima ha la funzione di riconoscere il tipo di sensore selezionato all'interno di un vettore di tipo AbstractSensor, rendendo possibile visualizzare a schermo il tipo del sensore. La seconda è una classe specifica per generare dati casuali da inserire nel vettore dei dati per la simulazione del sensore. Il problema principale è sempre riconoscere il tipo di sensore per poter determinare i suoi valori minimi e massimi. Dopo essere entrato nella funzione accept corretta, viene chiamata una funzione per la generazione di dati casuali all'interno di SensorRandomDataVisitor

Persistenza dei dati

Per la persistenza dei dati, senza il vincolo di gestire il formato JSON, ho implementato due pulsanti, "Open" e "Save", con le rispettive funzioni. Quando si preme "Save", si apre una finestra di dialogo per creare un file di testo con estensione .txt. In questo file salvo il vettore di sensori, dove ogni riga corrisponde a un sensore seguendo lo schema: ID, Name e Description. Quando si decide di aprire il file tramite il pulsante "Open", il contenuto viene letto e i sensori vengono caricati nel sistema.

Funzionalità implementate

L'applicazione è di tipo CRUD (create, read, update, delete), consentendo tramite funzioni nel main di creare, modificare ed eliminare un sensore. È inoltre disponibile un pulsante "Simula" che, ad ogni pressione, rigenera i dati casuali all'interno dei limiti definiti per ciascun tipo di sensore. Nella lista dei sensori è disponibile anche una funzionalità di ricerca per nome, che non si basa sull'inizio del nome ma verifica se la parola inserita è contenuta nel nome del sensore. Ad esempio, cercando "era", un sensore con il nome "Temperature" risulterà comunque un match.

Funzionalità:

1. Gestione di Sensori di Diversi Tipi:
    - Supporta tre tipologie di sensori: Geiger, Temperatura e Pressione.
    - Permette l'aggiunta, modifica e rimozione di sensori tramite interfaccia grafica.
2. Ricerca Sensori:
    - Ricerca dei sensori per nome con filtro in tempo reale nella lista dei sensori.
3. Simulazione Dati:
    - Generazione e visualizzazione di dati casuali per i sensori selezionati.
    - Visualizzazione dei dati simulati attraverso grafici interattivi.
4. Salvataggio e Caricamento Sensori:
    - Salvataggio della lista di sensori in un file di testo.
    - Caricamento della lista di sensori da un file di testo

Funzionalità estetiche:

1. Finestra di avviso se le constraints per ID o Name non sono rispettate
2. Finestre di avviso se i file sono stati caricati/salvati correttamente

Rendicontazione ore

Attività	Ore previste	Ore effettive
Studio e progettazione	10	10
Sviluppo del codice del modello	10	10
Studio del framework Qt	10	15
Sviluppo del codice della GUI	10	20
Test e debug	5	30
Stesura della relazione	5	5
Totale	50	**90**