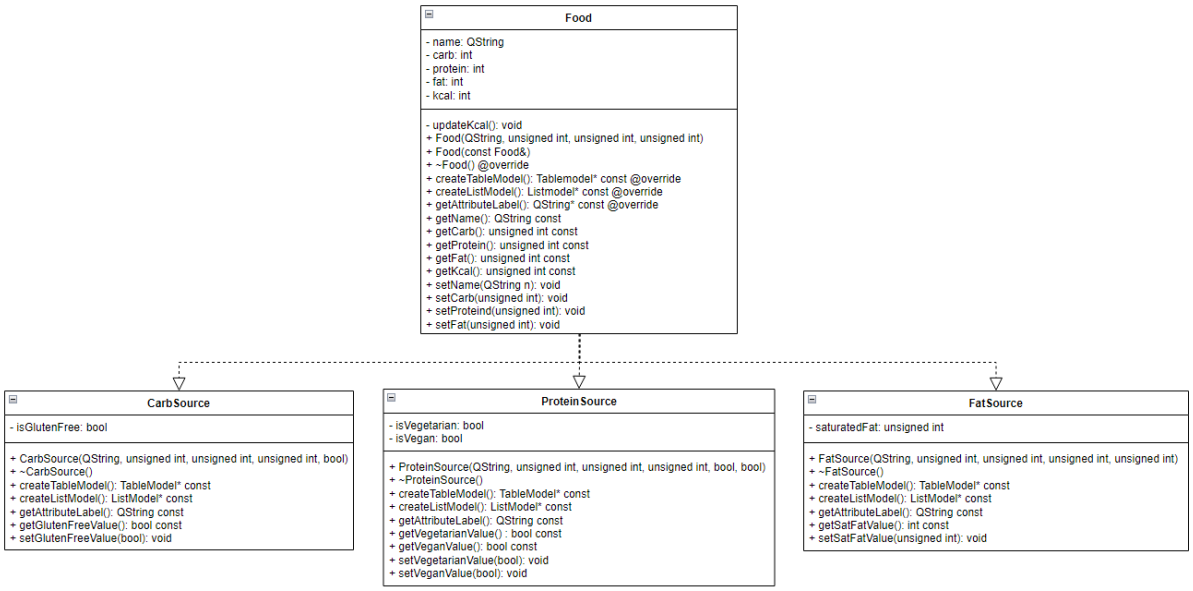
Oggetto	Relazione Progetto Programmazione ad Oggetti
Gruppo	Carraro Alessandro, mat. 2000548
Titolo	ProgettoPao

• Introduzione

ProgettoPao e' una semplice applicazione per la gestione dei pasti, che permette di creare combinazioni di alimenti, per cui è possibile aggiungere, togliere e modificare tali alimenti.
Gli alimenti sono naturalmente divisi secondo i macro nutrienti, che sono carboidrati, proteine e grassi, e ogni alimento e' caratterizzato dal nome, dal tipo (ovvero la macro categoria a cui e' associato per valori nutrizionali), e dai suoi principali valori nutrizionali.
L'applicazione fornisce, inoltre, una semplice tabella specifica di esempi e una semplice lista di informazioni per ogni categoria di alimento.
Infine è possibile visualizzare il menu' relativo ad un pasto, salvarlo oppure caricare menu' precedenti.
La scelta del tema nasce da una personale ricerca di un'applicazione con simili funzionalita', al fine di monitorare la mia alimentazione quotidiana, e non trovandola con facilita', e' stato naturale il desiderio di sviluppare questo tema come progetto accademico.

• Descrizione del modello



Il modello logico si basa sulla gerarchia con base astratta Food e sottoclassi derivate CarbSource, ProteinSource e FatSource.
Food rappresenta un alimento, ed essendo un concetto generico, viene considerato l'alimento secondo la sua principale fonte di nutrienti, per esempio carboidrati nella classe CarbSource, classe che intende rappresentare quegli alimenti come pasta, pane, riso, associati solitamente ai carboidrati.
Per le classi ProteinSource e FatSource il concetto e' analogo.
Inoltre è presente un contenitore in forma di linked list per contenere gli oggetti della gerarchia.
Per visualizzare un alimento è necessario inserirlo nel contenitore, che a sua volta e' visibile e modificabile tramite la classe FoodModification_box.

Questa classe gestite il contenitore, il bottone per la creazione di alimenti e il successivo inserimento nel contenitore stesso; inoltre gestisce le seguenti liste: la lista dei bottoni per l'eliminazione del cibo dal contenitore, la lista di oggetti della classe FoodSetWidget e la lista di oggetti della classe FoodExtraWidget, successivamente spiegate.
L'eliminazione avviene tramite un bottone che implementa la memorizzazione dell'indice del contenitore in cui e' aggiunto l'alimento; questo indice viene usato per determinare la posizione in cui eliminare gli oggetti all'interno delle liste e per effettuare la corretta chiamata polimorfa.
La classe FoodSetWidget e' estensione della classe widget e ha al suo interno tutti i widget necessari per la modifica dei cibi.
La classe FoodExtraWidget e' estensione della classe widget e ha al suo interno i bottoni per le chiamate polimorfe.
La classe Ribbon_Box funge da home per l'applicazione, contenendo i bottoni per le operazioni di input e output ed il bottone per la ricerca nel contenitore.
La classe ShowMenu_Box permette la visualizzazione degli alimenti, andando a mostrare i macro nutrienti e le calorie totali, così da permettere all'utente di vedere la combinazione di alimenti del pasto.
Infine la lettura da file andra' in automatico ad inserire gli alimenti nel contenitore, creando di conseguenza tutti i widget necessari per le modifiche, per le chiamate polimorfe e per la visualizzazione nella "Sezione Menu'".

● Polimorfismo

Nella classe Food sono presenti i metodi virtuali puri createTableModel e createListModel: il primo va a creare un puntatore alla classe TableModel ed il secondo alla classe ListModel, classi implementate da QAbstratModel.
In ogni classe derivata vengono ridefiniti i metodi andando a creare modelli congrui all'alimento di cui si vogliono le informazioni.
La classe FoodExtraWidget, estensione della classe QWidget si occupa della view della tabella e della lista; la classe riceve un Food* e successivamente andando ad analizzare il tipo dinamico crea i bottoni corretti per effettuare la giusta chiamata polimorfa.
La classe FoodSetWidget, estensione di QWidget, si occupa della modifica dei cibi; la classe riceve un Food* e analizzando il suo tipo dinamico aggiunge i widget necessari per la modifica di ogni specifico alimento. Ad esempio nel caso di un puntatore alla classe CarbSource aggiunge il label e la QCheckBox specifici per settare e visualizzare l'attributo "isGlutenFree".
Infine il metodo di scrittura su file va ad analizzare dinamicamente un Food* andando a scrivere gli attributi specifici della classe, mentre il metodo di lettura analizzando i dati su file va a creare il corretto oggetto della gerarchia.
I puntatori polimorfi sono tutti contenuti nel contenitore appositamente creato e vengono usati per la creazione di specifici widget all'interno della classe FoodSetWidget e per la corretta chiamata polimorfa all'interno della classe FoodExtraWidget. Infine vengono utilizzati per la corretta scrittura su file e per la lettura da file dei dati.

● Persistenza dei dati

Per la persistenza dei dati viene utilizzato il formato XML.
La scrittura e lettura di ogni cibo avviene tramite tag analoghi alle caratteristiche nutrizionali di un cibo: la tipologia, il nome, il valore di carboidrati, proteine e grassi presenti, più gli attributi specifici della classe derivata.
L'applicazione e' sviluppata per caricare anche piu' pasti, ad esempio per avere una visione dei pasti durante la giornata, andando ad inserire la colazione, il pranzo e la cena.

In alternativa c'e' la possibilita' di caricare un pasto per poi aggiungere, modificare o eliminare un cibo.
L'esempio pranzo.xml fornisce un semplice esempio di pasto per il pranzo, mentre gli esempi carb.xml, protein.xml e fat.xml forniscono dei dati per visualizzare le varie combinazioni di proprieta' per ogni tipologia di alimento.

● Funzionalita' implementate

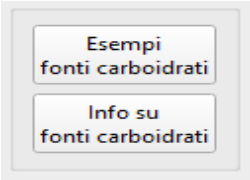
Funzionalita' oggetti della gerarchia:

- CREAZIONE: la creazione di oggetti della gerarchia avviene in due passaggi: il primo e' la pressione del bottone con l'icona "+" verde, che andra' a creare il puntatore Food*, e il secondo e' tramite un QComboBox che andra' dinamicamente a puntare ad un oggetto di una delle classi derivate.
  Il caricamento da file va in automatico a creare gli oggetti leggendo i dati nel file stesso.
- MODIFICA: la modifica degli oggetti avviene tramite widget grafici nella sezione della finestra denominata "Sezione Cibo"; le modifiche sono simultaneamente visibili nella sezione denominata "Sezione Menu'".
  La modifica avviene tramite oggetti della classe FoodSetWidget, per ogni alimento c'e' la possibilita' di modificare il nome, la quantita' di carboidrati, di proteine e di grassi; inoltre per ogni tipologia di fonte alimentare c'e' la possibilta' di modificare l'attributo specifico.
- ELIMINAZIONE: l'eliminazione degli oggetti avviene successivamente alla pressione del bottone con l'icona "X" rossa, andando a richiamare i costruttori specifici.

Funzionalita' contenitore per oggetti della gerarchia:

- INSERIMENTO: l'inserimento di oggetti all'interno del contenitore avviene tramite la pressione del bottone con l'icona "+" verde, l'inserimento avviene sempre alla fine del contenitore.
  Inoltre il caricamente da file va automaticamente ad inserire i dati nel contenitore.
- LETTURA: la lettura del contenitore e' visibile nella sezione denominata "Sezione Menu", essa avviene successivamente al cambiamento di qualche parametro di un alimento nel contenitore.
- CANCELLAZIONE: la cancellazione dal contenitore avviene successivamente alla pressione del bottone con l'icona "X" rossa. L'eliminazione avviene secondo un indice, questo indice e' l'attributo specifico della classe MyButton, derivata da QPushButton, l'indice memorizza la posizione all'interno del contenitore e utilizza una specifica implementazione di signal e slots.
- RICERCA: la ricerca nel contenitore avviene successivamente alla pressione del bottone presente nella sezione denominata "Home", tramite un QComboBox si sceglie il valore di cui si vuole avere il massimo valore e viene restituito l'alimento con il massimo valore richiesto.

Sono state implementate le funzionalità per la lettura e la scrittura su file tramite pressione di QPushButton nella sezione denominata "Home".
Infine ci sono due funzionalita' legate al polimorfismo: la visualizzazione di una tabella con alcuni esempi di fonti alimentari e una lista con alcune informazioni sulle fonti alimentari, entrambe mostrano dati a seconda della tipologia di fonte inserita nel menu'.
Queste due funzioni sono presenti nella sezione denominata "Sezione Cibo" tramite specifici bottoni, nel caso di una fonte di carboidrati, i bottoni sono i seguenti:



● Rendicontazione ore

Attivita'	**Ore Previste**	**Ore Effettive**
Studio e progettazione	10	10
Sviluppo del codice del modello	10	15
Studio del framework Qt	10	5
Sviluppo del codice della GUI	10	15
Test e debug	5	5
Stesura relazione	5	5
Correzioni per seconda consegna	15	15
Totale	65	**70**

Il monte ore e' stato leggermente superato poiché lo sviluppo del codice e' stato complicato dalla scelta degli oggetti della gerarchia.
La creazione, la modifica e l'eliminazione di questi oggetti hanno richiesto lo sviluppo di classi specifiche: FoodSetWidget e MyButton.
Inoltre e' servito ulteriore tempo per lo sviluppo del codice della GUI poiche' il puntatore al contenitore e' condiviso nelle 3 sezioni del programma; questa condivisione ha richiesto particolari attenzioni ai metodi di update e delete.
Le ore richieste per lo studio del framework Qt sono limitate a 5 poiche' ho familiarita' con esso.
La correzione per la seconda consegna ha richiesto abbastanza ore poiché sono state riviste molte parti del vecchio codice, dalla gerarchia alla GUI.

● Differenze rispetto alla consegna precedente

Rispetto alla precedente consegna e' stato rivisto il polimorfismo.
Nella classe Food sono stati inseriti i metodi virtuali puri: createTableModel e createListModel, al posto del metodo virtuale puro Example, andando ad implementare un uso non banale del polimorfismo.
Nella classe derivata CarbSource e' stato inserito l'attributo isGlutenFree, di conseguenza aggiornati il costruttore e il distruttore ed implementati i metodi get e set per tale attributo.
Nella classe derivata ProteinSource sono stati inseriti gli attributi isVegetarian e isVegan, di conseguenza aggiornati il costruttore e il distruttore ed implementati i metodi get e set per tali attributi.
Nella classe derivata FatSource è stato inserito l'attributo saturedFat, di conseguenza aggiornati il costruttore e il distruttore ed implementati i metodi get e set per tale attributo.
Sono stati modificati i metodi di lettura e scrittura su file andando a considerare i nuovi attributi delle classi della gerarchia.
Sono stati rivisti i nomi, sia delle classi, sia degli attributi e sia dei metodi, cercando di dare nomi più parlanti.
E' stata modificata la funzionalita' per selezionare il file da aprire andando ad utilizzare una finestra con un file explorer, come da suggerimento.

La GUI è stata migliorata attraverso l’uso di icone ed elementi grafici per facilitare l’utilizzo e renderla piu’ intuitiva.
Il layout è stato modificato e reso piu’ parlante
Nella relazione e’ stata inserita la rappresentazione grafica della gerarchia.
Nella sezione Modello e’ stata descritta la gerarchia Food.
E’ stata corretta la falsita’ riguardo il metodo virtuale puro “info”.
E’ stato descritto l’uso dei puntatori polimorfi, invece di riportare semplicemente dove essi siano presenti nel codice.