

astroKalk

Relazione sul progetto di Programmazione ad oggetti

Federico Caldart

Matricola **1097005**

Università degli studi di Padova

Anno accademico 2017/2018

The screenshot shows the 'astroKalk' application window. It is divided into several functional areas:

- Creazione oggetti** (Object Creation): A section for creating celestial objects. It includes a dropdown menu for 'Asteroide' and various input fields for properties like 'Raggio' (Radius), 'Temperatura' (Temperature), 'Densità' (Density), 'Età' (Age), 'Vel. rotazione' (Rotation velocity), 'Semiassie' (Semi-major axis), 'Atmosfera' (Atmosphere), 'Velocità' (Velocity), 'Sole' (Sun), 'Pianeta' (Planet), 'M. assoluta' (Absolute magnitude), and 'M. apparente' (Apparent magnitude). A 'Crea' button is at the bottom.
- Calcolo** (Calculation): A section for performing calculations. It has a text input 'Su questo corpo celeste peseresti 18.9837 chili' and a dropdown for 'Pianeta'. Below are buttons for 'Operazioni binarie' (Binary operations) like 'Somma' (Sum) and 'Rapporto volume' (Volume ratio), and 'Operazioni unarie (primo operando)' (Unary operations (first operand)) like 'Volume', 'Superficie', 'Massa', 'Vel. di fuga', 'Peso Extraterrestre', 'Collisione con Terra', 'Distanza dalla Terra', 'Distanza dalla stella', 'Durata anno', 'Rotazione sincrona?', 'Abitabile?', 'Rivoluzioni in un anno', 'Vel. orbitale', and 'Durata giorno'.
- Disegno** (Drawing): A section for drawing celestial objects. It includes a 'Disegna in scala' (Draw in scale) input, a 'Disegna in ordine di età' (Draw by age) checkbox, and a 'Disegna in ordine di età' (Draw by age) checkbox. Below is a drawing area showing a sun and planets, and a legend for drawing in scale or by age.

Istruzioni di compilazione: per compilare, dare i comandi

```
qmake astrokalk.pro
```

```
make
```

Scopo del progetto

Il progetto si propone di realizzare una calcolatrice in grado di operare con gli oggetti celesti; tali oggetti possono essere di quattro tipi: *asteroide*, *stella*, *pianeta* e *satellite*.

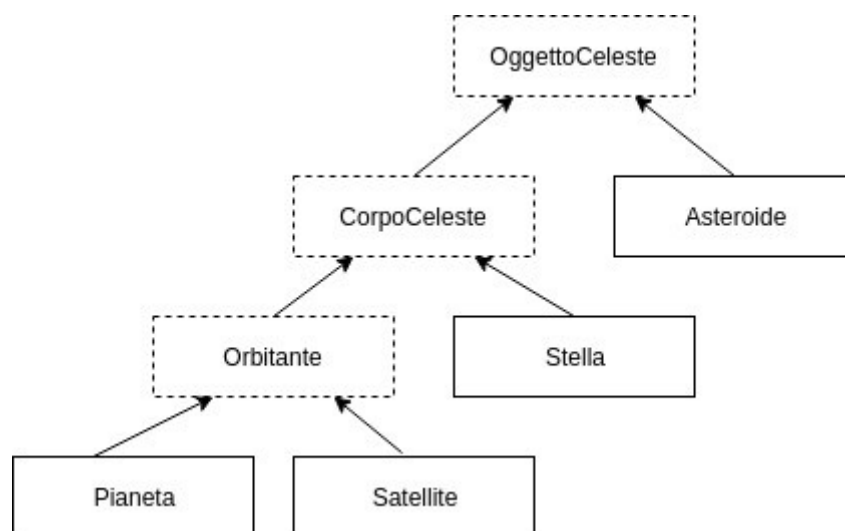
Principale scopo è offrire operazioni di natura numerica, ma sono disponibili anche modeste funzionalità di rappresentazione grafica.

I calcoli sono effettuati applicando agli astri formule di fisica o geometria, cercando di utilizzare unità di misura tali da facilitare la comprensione dei risultati all'utente e allo stesso tempo far fronte alla limitatezza della rappresentazione numerica dei calcolatori: riguardo quest'ultimo punto, volevo sottolineare che i risultati troppi grandi non sono trattati esplicitamente come errori; è pertanto possibile avere, in casi limite, risultati uguali a 0 o infinito, proprio come succederebbe in una calcolatrice tradizionale. Ovviamente, vista la dimensione dei dati e la complessità dell'argomento, non si assicura una precisione di calcolo del 100%, ma il quanto più realistica possibile.

Per quanto riguarda la classe **Use** di **Java**, si vuole fare notare che il main comprende delle stampe dei risultati dei calcoli, che sono state commentate e lasciate di proposito.

Gerarchie di tipi

Il progetto presenta due gerarchie di tipi. La più importante è quella riguardante gli oggetti operandi dei calcoli:



Essa non vuole essere una rappresentazione precisa della realtà, ma una semplificazione adattata agli scopi dell'applicazione; inizia con una classe base generica e astratta e scendendo nella gerarchia si specializza offrendo nuove funzionalità di calcolo e campi dati:

OggettoCeleste possiede i campi dato comuni raggio, temperatura media superficiale, densità media ed età, oltre ai primi metodi di calcolo unari (Volume, Massa, Superficie), binari (rapportaVolume, fusione) e statici, usati su contenitori di oggetti per le operazioni di disegno (ordinaPer, disegnoInScala, disegnoInScalaEta).

fusione non è un'operazione della calcolatrice, ma un metodo creato con lo scopo di modularizzare il codice, racchiudendo le istruzioni comuni per la definizione dell'operatore di somma delle classi concrete.

Avviene poi la prima ramificazione, con la concretizzazione della base in **Asteroidi**, che aggiunge il campo *velocità*, il quale indica la velocità con la quale l'asteroide viaggia nello spazio ed il metodo *collisione*, che calcola i possibili effetti della collisione dell'oggetto con la Terra.

Fratello di quest'ultimo è **CorpoCeleste**, classe che rimane astratta e rappresenta oggetti celesti tenuti insieme dalla gravità e aggiunge il campo dati *velocitaRotazione*, e alcuni metodi di calcolo: *CalcPeso* che, dato un peso sulla terra in kg, ne calcola approssimativamente la corrispondenza sul corpo celeste d'invocazione, *VelFuga* e *Giorno*, che calcola la durata di un giorno sul corpo (cioè quanto impiega a fare il giro su se stesso). Ha due derivazioni:

Stella, classe concreta che aggiunge i campi dati *magnitudineAss* e *magnitudineApp*, i quali memorizzano rispettivamente le magnitudini assoluta e apparente della stella. Aggiunge il metodo *distanzaTerra* che calcola la distanza approssimativa tra stella e Terra, basandosi sulle magnitudini.

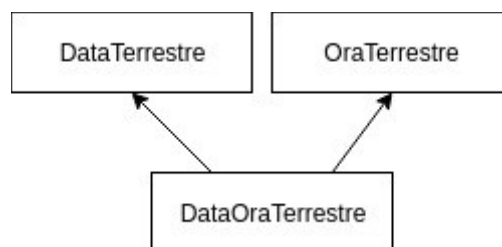
Orbitante è una classe che rimane astratta e definisce un *CorpoCeleste* che si muove seguendo un'orbita; aggiunge il campo dati *semiAsseOrbita*, che memorizza la lunghezza del semiasse maggiore dell'orbita dell'oggetto, e tre metodi virtuali puri: *velOrbitale*, *distanzaSole*, e *periodoOrbitale*.

Le ultime due classi sorelle sono:

Pianeta, classe concreta che aggiunge i campi *atmosfera* e *sole*, che indica la stella attorno a cui ruota il pianeta. Possiede tre nuovi metodi: *ESI*, che non è una vera e propria operazione della calcolatrice, ma un metodo a supporto di *Abitabile*, *etaExtraTerrestre*, che, data una data di nascita, calcola l'età che la persona nata quel giorno avrebbe sul pianeta d'invocazione e implementa tutti i metodi virtuali puri ereditati.

Satellite (naturale), classe concreta che aggiunge il campo dati *pianeta*, che indica il pianeta attorno a cui ruota il satellite. Aggiunge i metodi *rivoluzioniAnnue*, che calcola quante volte il satellite compie una rivoluzione intorno al pianeta nel tempo in cui il pianeta compie una sola rivoluzione intorno alla stella e *rotazioneSincrona*, che calcola se il satellite ha una rotazione sincrona al pianeta. Implementa inoltre tutti i metodi puri virtuali ereditati.

La seconda gerarchia è stata creata per una facilitare la gestione delle funzioni che calcolano intervalli di tempo :



La libreria standard del c++ offre delle funzioni in grado di gestire e manipolare valori temporali, tuttavia essi sono rappresentati come tempo trascorso da un'epoca fissata (tipicamente 1900 o 1970) e identificano vere e proprie date. Lo scopo delle classi definite nel progetto è, invece, rappresentare intervalli di tempo, esterni al concetto di calendario.

DataTerrestre è rappresentata internamente con un numero di giorni, mentre **OraTerrestre** con il numero di secondi; da esse deriva **DataOraTerrestre** che non aggiunge campi dati né metodi, ma si limita a unire nello stesso tipo le funzionalità dei genitori. In questo modo, dato un oggetto di tipo *DataOraTerrestre*, è possibile

ricavarne ore, minuti, giorni, anni interi e “anni frazionari”, è cioè possibile utilizzarlo nella forme #anni, #giorni, #ore, #minuti oppure frazione di anno.

Utilizzo di codice polimorfo

All’ interno della gerarchia si fa uso di codice polimorfo, dato che la parte logica dell’applicazione sarà gestita da **model**, classe che memorizza gli oggetti operandi della calcolatrice in collezioni di puntatori polimorfi.


Innanzitutto il distruttore viene dichiarato virtuale nella classe base, per evitare memory leak, in modo che la distruzione su puntatore polimorfo a OggettoCeleste richiami il distruttore corretto.

In seguito, si incontrano i tre metodi puri di Orbitante, che hanno una definizione diversa nelle sue sottoclassi. Essi vengono usati in maniera polimorfa nel metodo *calcola* di model: essendo chiamati su puntatori polimorfi a Orbitante, verrà ogni volta selezionato a tempo d’esecuzione il giusto metodo da invocare (righe 157, 161 e 200 di model); inoltre, nell’overloading di *calcola* che inizia a riga 211, si fa uso del *dynamic_cast* sul primo operando per conoscere il tipo dei due operandi ed invocare il giusto operatore di somma.

In model viene poi utilizzato il *dynamic_cast* nei metodi *disegnoInScala* e *disegnoInScalaEta* per ricavare il tipo degli oggetti contenuti nella collezione passata, in modo da poter assegnare ad ogni oggetto nel vettore una stringa che ne identifichi il tipo, la quale sarà poi usata per il disegno, evitando di fare controlli RTTI nella parte grafica.

Manuale utente

La GUI è divisa in 4 sezioni distinte:

1. Area di **creazione**, nella quale è possibile creare gli oggetti che verranno poi sottoposti al calcolo. L’utente è invitato a scegliere il tipo di oggetto da creare, in modo che vengano impostati come editabili solo i campi specifici per quel tipo; ogni campo ha a fianco un’icona () che fa scattare il pop-up di un suggerimento qualora vi si passi sopra con il puntatore del mouse.
2. Area di **selezione**, nella quale è possibile visionare gli oggetti creati con i relativi dettagli: selezionando un oggetto nella lista verranno visualizzati tutte le sue specifiche.
3. Area di **calcolo**, come per l’area di creazione, l’utente è invitato a scegliere il tipo degli oggetti su cui vorrà fare i calcoli: i pulsanti nell’interfaccia cambieranno in base alla selezione. Le operazioni identificate come “unarie” operano sul primo oggetto (quello più a sinistra), mentre le “binarie” operano su entrambi gli oggetti, secondo l’ordine *oggetto_sinistra OP oggetto_destra*, dove OP identifica l’operazione:

	Asteroide	Stella	Pianeta	Satellite	Descrizione:
Volume	✓	✓	✓	✓	Calcola il volume dell'oggetto.
Superficie	✓	✓	✓	✓	Calcola la superficie dell'oggetto.
Massa	✓	✓	✓	✓	Calcola la massa dell'oggetto.
Vel. di fuga	×	✓	✓	✓	Calcola la velocità di fuga dall'oggetto.
Peso Extraterrestre	×	✓	✓	✓	Dato un peso in kg, calcola a quanto equivarrebbe su un altro oggetto celeste.
Collisione con Terra	✓	×	×	×	Calcola le ipotetiche conseguenze della collisione dell'asteroide con il pianeta Terra.
Distanza dalla Terra	×	✓	×	×	Calcola la distanza approssimata tra l'oggetto ed il pianeta Terra.
Durata anno	×	✓	✓	✓	Calcola la durata dell'anno sul corpo celeste, cioè il periodo che impiega il corpo a fare una rivoluzione completa intorno al proprio sole, in anni terrestri.
Abitabile?	×	×	✓	×	Indica se il pianeta è teoricamente abitabile, basandosi su comparazioni con il pianeta Terra.
Eta Extraterrestre	×	×	✓	×	Data una data di nascita (gg/mm/yyyy), calcola l'età che avrebbe una persona nata quel giorno sul pianeta.
Rotazione sincrona?	×	×	×	✓	Indica se il satellite ha una rotazione sincrona con il pianeta attorno a cui ruota, cioè se mostra sempre la stessa faccia o no.
Rivoluzioni in un anno	×	×	×	✓	Calcola quante rivoluzioni complete compie il satellite intorno al pianeta nel tempo in cui il pianeta compie una rivoluzione completa intorno al sole.
Vel. orbitale	×	×	✓	✓	Calcola la velocità alla quale viaggia l'orbitante sulla sua orbita.
Durata giorno	×	✓	✓	✓	Calcola la durata del giorno sul corpo celeste, cioè quanto tempo impiega a girare su se stesso, in tempo terrestre.

4. Area di **disegno**, all'interno della quale è possibile eseguire due diverse operazioni di rappresentazione su gruppi di oggetti: *Disegna in scala*, che dà una rappresentazione degli oggetti ordinati e scalati in base alla dimensione e *Disegna in ordine d'età*, che rappresenta gli oggetti ordinandoli per età.

In entrambi i casi, l'input richiede una sintassi particolare. Ogni oggetto che si voglia rappresentare deve essere identificato con:

1. Una lettera: A per asteroide, S per stella, P per pianeta e L per satellite (L = Luna);
2. Un trattino - ;
3. Un numero che lo identifica nella lista degli oggetti creati.

Ogni oggetto deve essere seguito da una virgola (,) o un punto (.) , dal primo all'ultimo. Quando l'input è completo, non va premuto *invio*, ma deve venir cliccato il pulsante.

Oltre al disegno, comparirà una riga di testo che servirà ad identificare gli oggetti rappresentati: i numeri, da sinistra a destra, si riferiranno ai disegnati, nello stesso ordine; per esempio, se il primo numero è 5 ed il primo oggetto disegnato è una stella, allora l'oggetto più piccolo tra i disegnati è la stella che ha indice 5 nella lista delle stelle create.

Precisazioni su alcune scelte implementative

L'intera applicazione è stata fornita di label provviste di tooltip utili a facilitare l'interazione dell'utente; inoltre, è stato deciso di inserire *astrokalk* in uno scrollable widget : la finestra creata dal programma ha un'ampiezza considerevole e se non fosse scorrevole causerebbe problemi di visualizzazione in monitor a bassa risoluzione.

OggettoCeleste e **CorpoCeleste** sono state create astratte, definendo virtuale puro il distruttore, scelta guidata dalla volontà di rendere la base un'interfaccia comune non istanziabile, pur non avendo veri e propri metodi puri: un **OggettoCeleste** è infatti la definizione di qualcosa che sta nello spazio senza un reale significato, ma si concretizza e assume nomi e forme diverse in base alla sua natura (stessa cosa vale per **CorpoCeleste**).

La classe **DataOraTerrestre** non è stata implementata in java perché il linguaggio fornisce la classe *Duration*, la quale offre esattamente le funzionalità richieste.

In **Model**, e più precisamente in *calcola*, si è deciso di usare `static_cast` perché i controlli effettuati preventivamente a livello di GUI sull'input dell'utente mi assicurano la sicurezza della conversione.

Ore utilizzate

analisi preliminare del problema: 7 ore.

progettazione modello: 8 ore.

progettazione grafica: 3 ore.

codifica modello: 12 ore.

codifica grafica: 20 ore.

apprendimento libreria qt: 6 ore.

test e debug: 2 ore.

totale: 58 ore.

Le ore aggiuntive sono state causate dal difficile reperimento di alcune delle formule utilizzate nei calcoli: la loro applicazione doveva essere precisa e doveva utilizzare le unità di misura giuste.