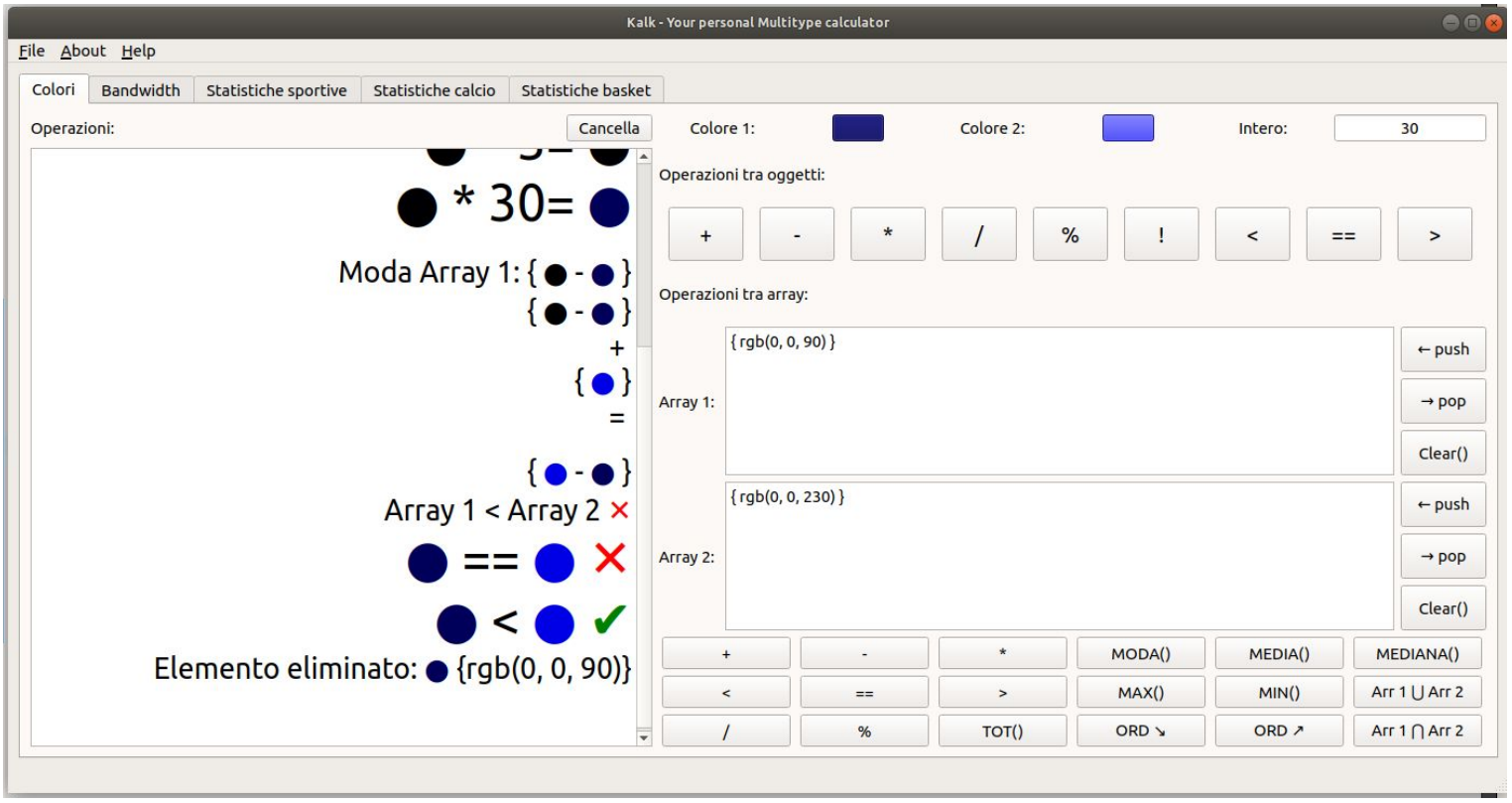
Ciprian Voinea
1143057

Progetto di Programmazione ad Oggetti a.a. *2017 - 2018*

'Multitype Kalk'



Sviluppato in collaborazione con:

Lorenzo Busin
1143782

INDICE

1. Introduzione

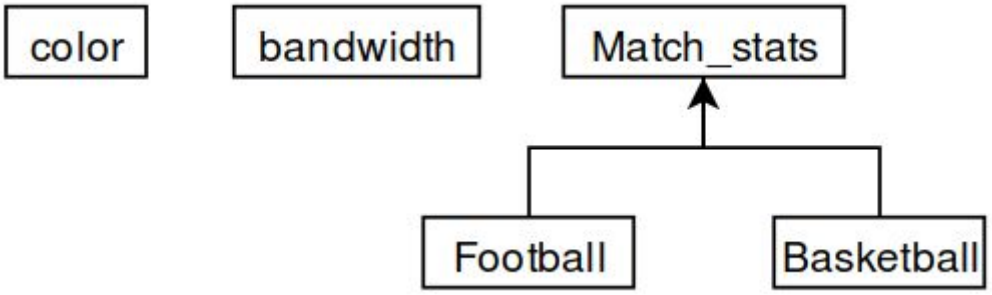
Lo scopo del progetto consiste nella realizzazione di una calcolatrice estendibile che possa permettere di effettuare operazioni su tipi di calcolo diversi e l'aggiunta futura in maniera modulare di nuovi tipi di calcolo definibili dal programmatore.
La *Multitype Kalk* sviluppata in questo progetto permette operazioni su cinque tipologie differenti di oggetti e la possibilità di creare ed operare su array di oggetti dello stesso tipo aggiungendo nuove funzionalità.

2. Suddivisione dei compiti

Per quanto riguarda la suddivisione dei compiti abbiamo lavorato in modo unito confrontandoci lungo tutto il periodo del progetto.
Io mi sono occupato di quel che riguarda l'oggetto color e interamente della GUI, mentre il mio compagno si è occupato maggiormente della parte di bandwidth, della gerarchia delle statistiche e della parte di JAVA.
L'utilizzo di una repository comune è stata fondamentale in quanto con questa abbiamo potuto controllare prontamente eventuali bug e possibili ottimizzazioni ed implementare il codice che ognuno ha scritto in modo coeso ed in linea con tutto il resto del progetto.

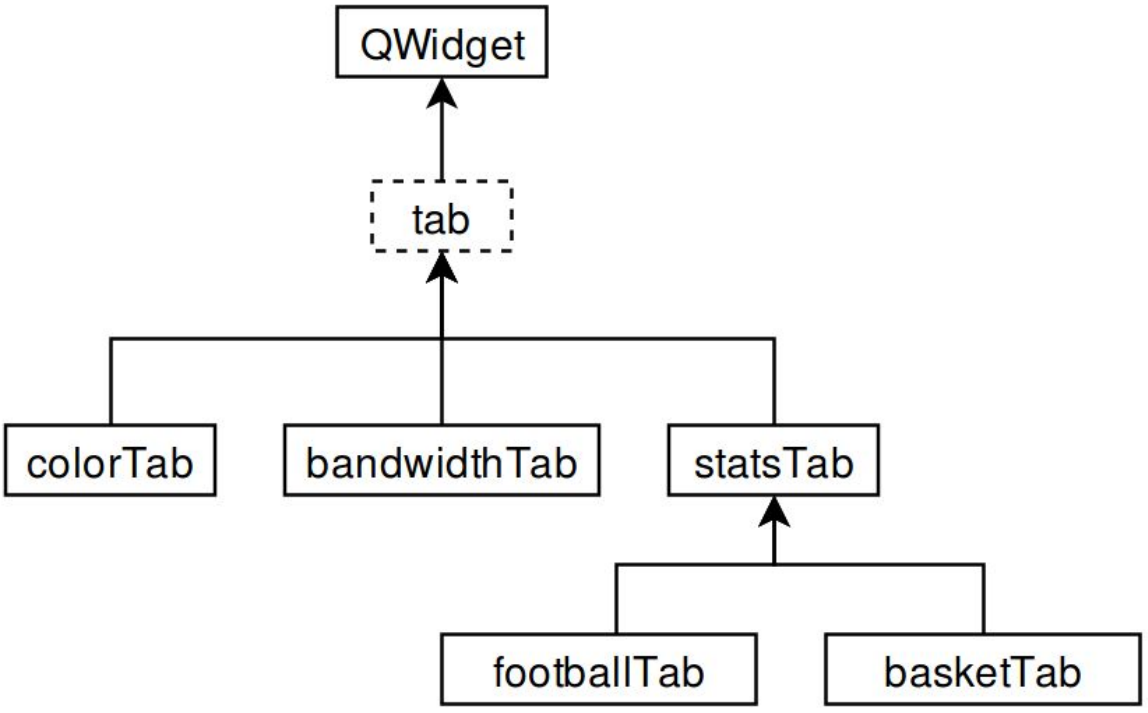
3. Descrizione delle classi e delle gerarchie

Le classi più importanti del progetto sono quelle che definiscono gli operatori e quelle che definiscono la parte grafica.
Per la parte degli operatori le classi sono:

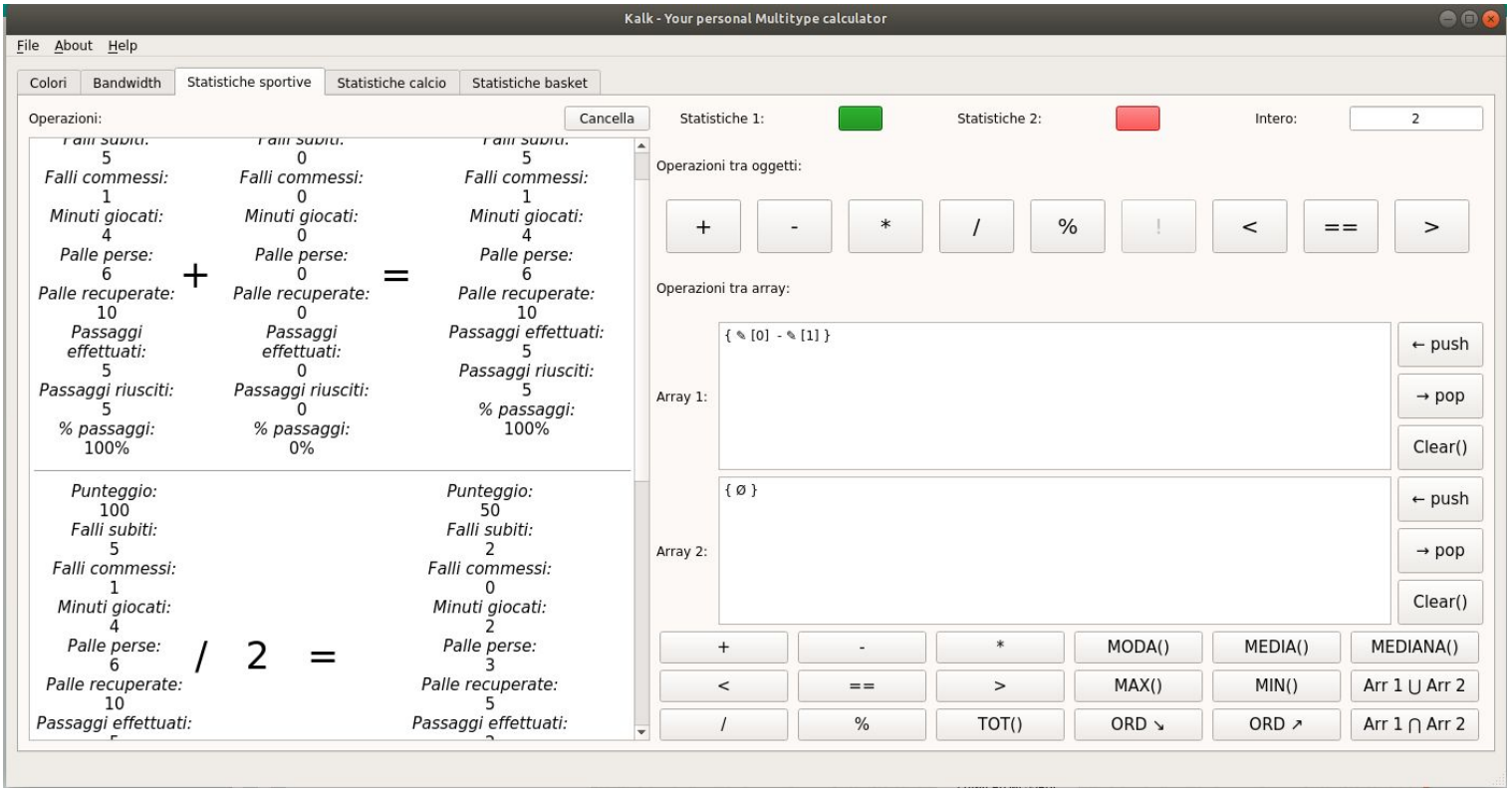


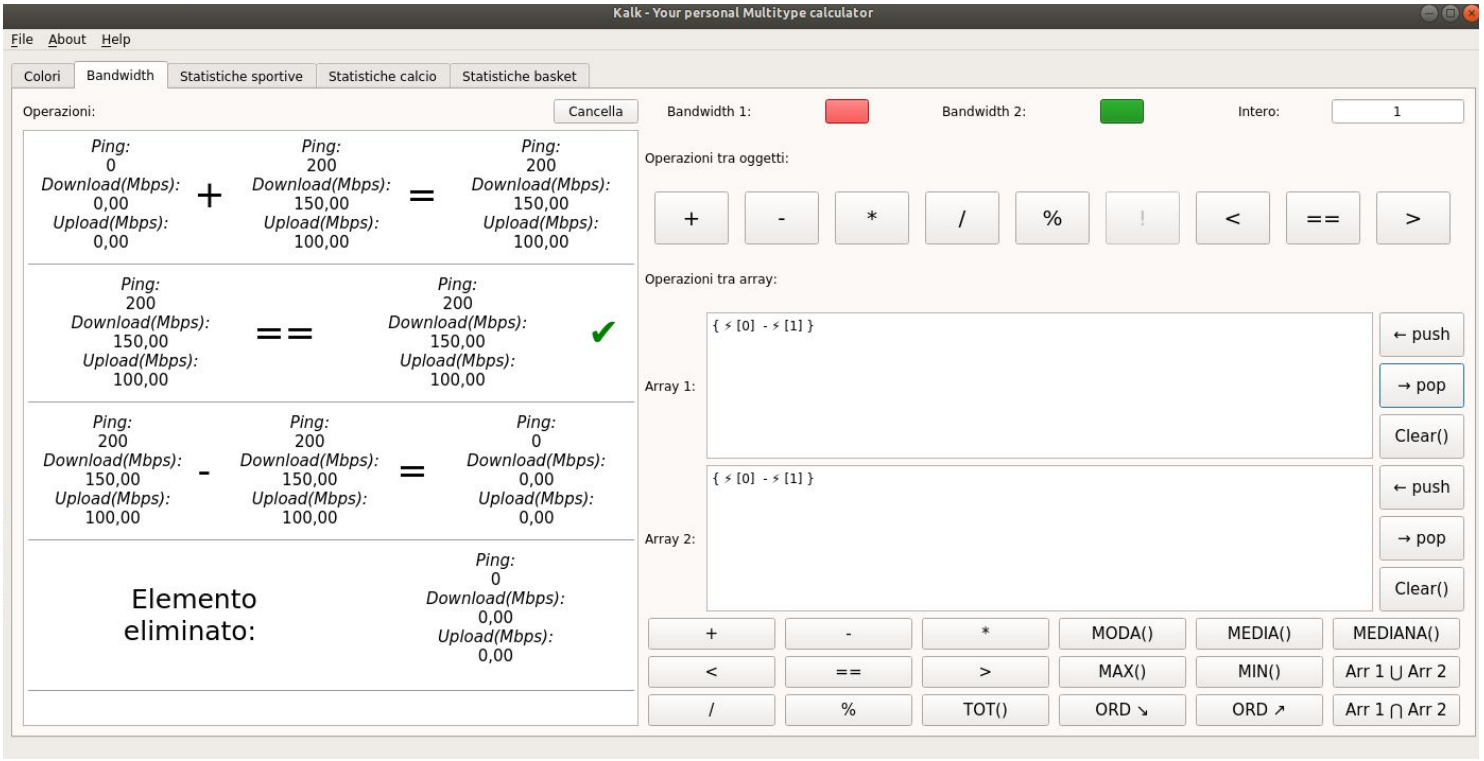
- color: oggetti che rappresentano colori in formato RGB
- bandwidth: oggetti che rappresentano valori di una connessione ad Internet
- Match_stats: oggetti che rappresentano statistiche in comune di molti sport
    - Football: oggetti che rappresentano statistiche specifiche del calcio
    - Basketball: oggetti che rappresentano statistiche specifiche del basket

Per la parte grafica invece, la gerarchia di classi più importante che viene utilizzata è rappresentata dai tab:



Ad ogni classe tab corrisponde univocamente una classe operatore.
Ciascuna tab specifica deriva dalla classe base tab che inizializza gli oggetti comuni come layout e pulsanti, mentre lascia virtual le definizioni degli slot che andranno ad essere implementati in ciascuna sottoclasse.





4. Principali metodi utilizzati

Ciascuno dei cinque tipi di calcolo fornisce in l'overloading dei principali operatori.
Non sono permesse operazioni fra oggetti di tipo diverso.
I principali metodi dei tipi di calcolo da noi definiti che vengono utilizzati nel corso del programma sono:

- **print:** restituisce l'oggetto sotto forma di stringa.
- **operator+ :** esegue la somma campo a campo tra due oggetti.
- **operator- :** esegue la sottrazione campo a campo tra due oggetti.
- **operator* :** esegue il prodotto di ogni campo dati con un intero.
- **operator/ :** esegue la divisione di ogni campo dati con un intero.
- **operator% :** esegue il modulo di ogni campo dati con un intero.
- **operator==** : ritorna TRUE se i due oggetti sono uguali.
- **operator>** : ritorna TRUE se il primo oggetto è maggiore del secondo.
- **operator<** : ritorna TRUE se il primo oggetto è minore del secondo.

Per la classe color ha avuto senso implementare l'operatore ! (not) che restituisce il complementare dell'oggetto su cui viene applicato:

- **operator!** : ritorna il colore complementare

La classe templetizzata array fornisce inoltre i seguenti metodi:

- **operator+** : esegue la somma tra gli oggetti di due array che si trovano nella stessa i-esima posizione.
- **operator-** : esegue la sottrazione tra gli oggetti di due array che si trovano nella stessa i-esima posizione.
- **operator*** : esegue la moltiplicazione di ogni elemento dell'array con un intero.
- **operator/** : esegue la divisione di ogni elemento dell'array con un intero.
- **operator%** : esegue il modulo di ogni elemento dell'array con un intero.
- **operator==** : ritorna TRUE se tutti gli oggetti del primo array sono uguali a tutti gli oggetti del secondo array e si trovano nella stessa i-esima posizione.
- **operator>** : ritorna TRUE se il totale del primo array è maggiore di quello del secondo.
- **operator<** : ritorna TRUE se il totale del primo array è minore di quello del secondo.
- **tot** : restituisce un oggetto che è la somma di tutti gli oggetti dell'array.
- **media** : ritorna un oggetto che è la media di tutti gli oggetti dell'array.
- **mediana** : ritorna l'oggetto che si trova a metà dell'array.
- **moda** : ritorna l'oggetto o gli oggetti che si ripetono con più frequenza dentro l'array.
- **max** : ritorna il massimo dell'array.
- **min** : ritorna il minimo dell'array.
- **ordinamentoCrescente** : ordina gli elementi dell'array in modo crescente.
- **ordinamentoDecrescente** : ordina gli elementi dell'array in modo decrescente.
- **unione** : esegue l'unione tra due array.
- **intersezione** : ritorna gli oggetti in comune tra due array.
- **push_back** : inserisce un oggetto nella coda dell'array.
- **pop_back** : elimina e ritorna l'ultimo oggetto nella coda dell'array.

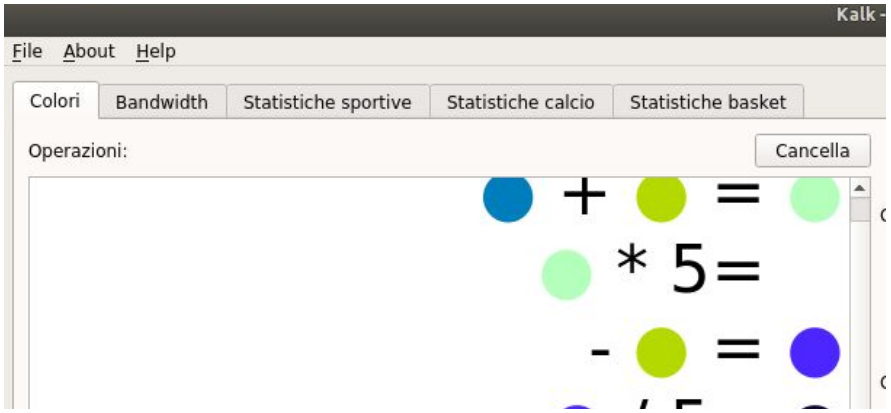
Oltre a queste operazioni in ogni classe sono definiti:
- costruttore, costruttore di copia e distruttore.
- ridefinizione dell'operatore di assegnazione.
- un metodo per settare tutti i campi dati.
- metodi di get per ogni campo dati

Alla maggior parte di questi metodi corrisponde un oggetto di tipo `QPushButton` che genera un segnale specifico per richiamare lo slot che la gestisce.

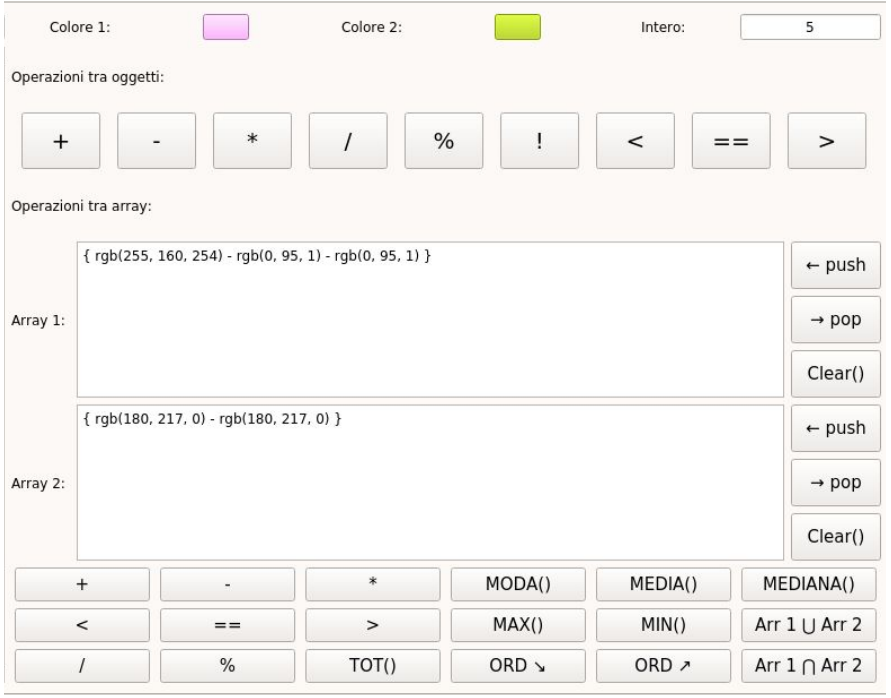
Per la parte della GUI, i metodi principali e più utilizzati sono dati dagli slot attivati che vengono collegati ai signal generati dai click dei `QPushButton`.
Sono quasi tutti dichiarati virtuali puri in quanto ogni implementazione di `tab` necessita di un utilizzo simile ma non uguale.

5. Esempio di utilizzo

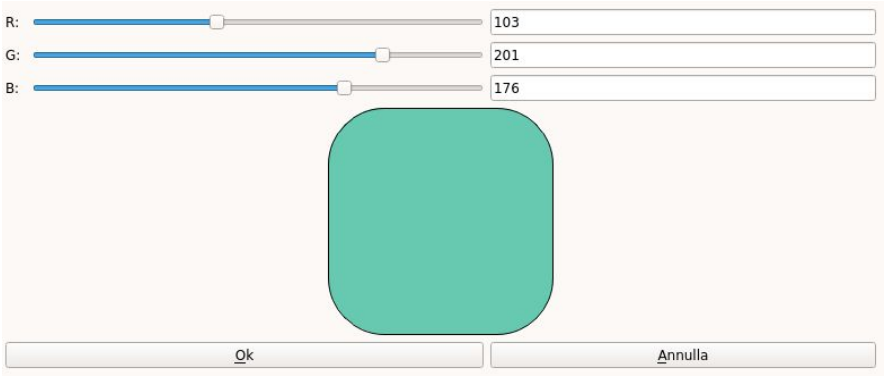
L'utilizzo è, dal mio punto di vista, semplice ed intuitivo. All'utente viene presentata una schermata suddivisa in vari tab ognuno con un tipo di calcolo specifico.
Un esempio con il tipo di calcolo colore:



sulla parte sinistra vengono mostrate le operazioni effettuate. Dal menu si può accedere ad una finestra di *About* e di *Help,* nelle quali verrà rispettivamente spiegato cos'è e come si utilizza questo programma.



Sulla parte destra si possono inserire gli oggetti, scegliere le operazioni da fare e gestire gli array di oggetti.
Quando viene cliccato su un oggetto viene cambiato layout in modo da mostrare un menu con i campi di inserimento specifici per ciascun tipo di calcolo.



Nell'inserimento, vengono visualizzati tre slider che permettono l'inserimento dei valori interi che rappresentano l'oggetto

la visualizzazione del mix di colori in tempo reale.

Il pulsante che rappresenta l'oggetto cambia colore in base ai valori inseriti, nel caso degli altri oggetti questo diventa rosso o verde a seconda se i campi contengono o meno valori non nulli (sport) oppure se sono tutti impostati e non nulli (bandwidth).
Nel caso di operazioni non disponibili tutti i messaggi di errore vengono visualizzati nella status bar, tranne nel caso della divisione per 0 che mostra il messaggio direttamente nel box `history`.

6. Uso del polimorfismo

Il polimorfismo è largamente utilizzato nella parte grafica specialmente per la parte dei tab. Ho deciso di rendere il programma il più modulare possibile creando una classe astratta tab che contiene tutti i metodi e gli oggetti comuni a tutte le classi derivate.
Alcuni metodi vengono implementati mentre altri vengono dichiarati virtuali puri e la loro implementazione viene lasciata alle singole sottoclassi.
I metodi `selector()` utilizzano uno `static_cast<QPushButton*>(sender())` per controllare quale dei due pulsanti che possono invocarla è stato premuto.

7. Estensibilità

Questa applicazione è facilmente estensibile in quanto per aggiungere un nuovo tipo di calcolo basta solamente creare la classe e definire un tab specifico derivandolo dalla classe base tab ed implementandone i metodi.
E' facilmente estendibile nella la parte del calcolo di statistiche sportive in quanto possibile derivare direttamente statsTab per poter implementare i calcoli di un nuovo sport.
Il fatto che la classe array sia stata implementata con un template rende immediato sfruttarla con nuovi tipi di calcolo.

8. Altre scelte progettuali

Una delle scelte progettuali degna di essere nominata nella relazione consiste nel non utilizzare try e catch in quanto tutti gli errori possibili (es. / e % per 0) vengono gestiti tramite if e controlli sui valori.
Un'altra scelta è stata quella di gestire la divisione per 0 in modo che nel caso di un valore nullo allora questo viene cambiato automaticamente ad 1 visualizzando questo cambiamento nel box `history`.

9. Ore richieste per la realizzazione

- Analisi preliminare del problema: 4 ore

- Progettazione modello: 5 ore
- Progettazione GUI: 5 ore
- Apprendimento libreria QT: 7 ore
- Codifica modello: 7 ore
- Codifica GUI: 20 ore
- Debugging: 4 ore
- Testing: 3 ore

- Totale: 55 ore

10. Ambiente di sviluppo

Sistema operativo: Ubuntu 17.10 x86_64
Compilatore: g++ (Ubuntu 7.2.0-8 ubuntu3) 7.2.0
QT Creator 5.9.1

11. Note sulla compilazione

In C++ il file *.pro da usare è quello presente nella cartella della consegna in quanto è stato adattato per la corretta compilazione del progetto aggiungendo opzioni come `CONFIG += c++11` e `QT += widgets` per via dell'utilizzo di oggetti come il menu (File / About / Help) ed i metodi `QAction()`:

➔ C++: `qmake ; make`
➔ Java: `javac *.java ; java Kalk`