SCOPO DEL PROGETTO:

Questo software è pensato per un'azienda di trasporto bagagli che ha bisogno di registrare tutti i bagagli che partono dalla propria città.
Un bagaglio può essere spedito in vari modi.
Un bagaglio è costituito dalle seguenti informazioni:

1. Tipo
2. Prezzo
3. Proprietario
4. Eventuale partenza del proprietario assieme al bagaglio
5. Modalità di viaggio
6. Città di destinazione
7. ID(univoco)
8. Data di partenza
9. Massa e dimensioni.

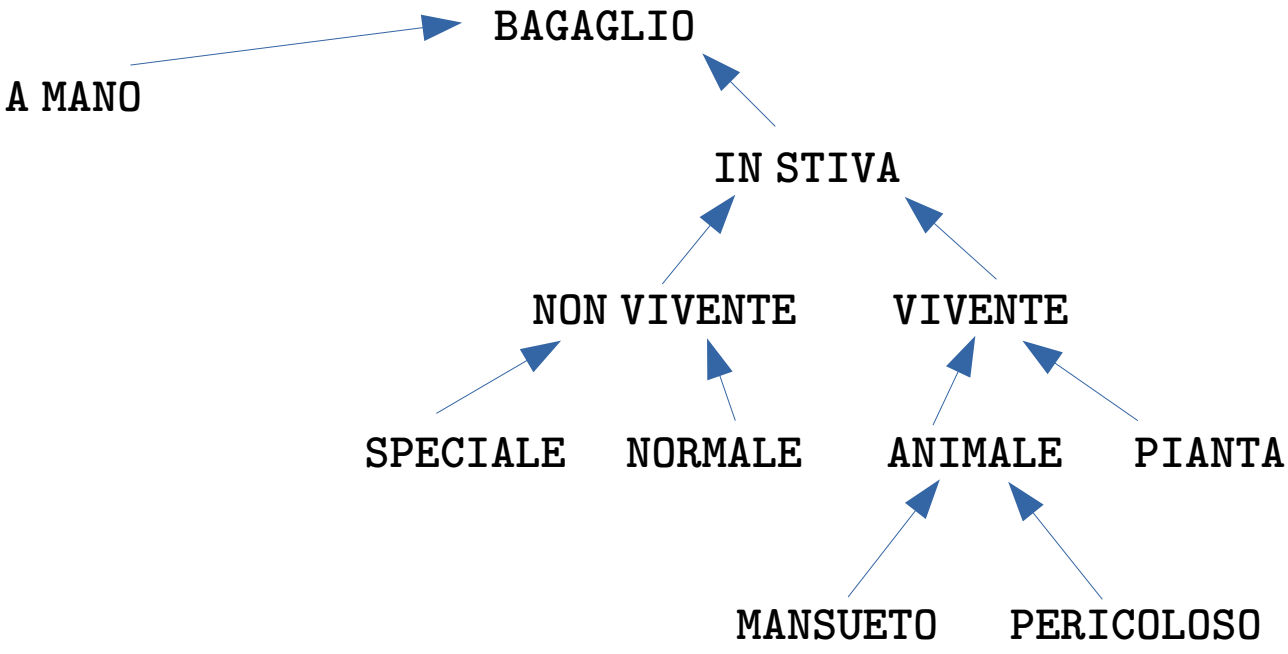
Il software permette le seguenti operazioni:

1. Aggiunta di un bagaglio
2. Eliminazione di un bagaglio(uno a scelta oppure l'ultimo inserito)
3. Eliminazione di tutti i bagagli.
4. Modifica di alcune informazioni(Massa e destinazione)
5. Salvataggio/Caricamento da file.xml (persistenza dei dati)
6. Ricerca di un bagaglio in un file in base a uno dei seguenti parametri: ID,destinazione, proprietario.
7. Visualizzazione dei bagagli presenti nel file xml sotto forma di lista grafica
8. Visualizzazione di tutte le informazioni di un bagaglio.

Per ogni input da parte dell'utente, esiste un controllo tramite espressioni regolari.

DESCRIZIONE DELLA GERARCHIA

Di seguito è illustrata la gerarchia dei tipi:



Utilità e ruolo dei tipi

Un bagaglio può essere di un certo tipo, e la gerarchia illustra i possibili tipi che un bagaglio può assumere.
Non sono stati pensati altri tipi di bagaglio, ma in caso vi sia la necessità, si può derivare un'altra classe nella gerarchia(nel rispetto dell'OWA).

Descrizione dell'uso del codice polimorfo

Oltre al distruttore, esistono i seguenti metodi virtuali.

```
virtual double Tassa() const;
```

Il prezzo di un bagaglio varia in base ad alcuni parametri, ed uno di questi è proprio il tipo (gli altri sono relativi a massa e dimensioni). Un bagaglio qualsiasi costa 0,50 euro al Kg e 35 euro al metro cubo. Esistono delle spese extra(diverse) se il bagaglio è speciale(esplosivo, tossico,fragile), se è vivente o se è un animale pericoloso.

```
virtual QString getTipo() const;
```

Nella lista grafica, esiste una colonna che descrive il tipo del bagaglio. Questo metodo, il quale a seconda del tipo a run-time potrebbe contenere chiamate allo stesso metodo della classe padre, ritorna la QString che identifica il tipo del bagaglio.

```
virtual QString getType() const;
```

È quasi come il metodo precedente, solo che la QString viene restituita in inglese e in modo più compatto, ovvero lo stesso modo con il quale la lista dei tipi disponibili è presentata all'utente(si veda il QcomboBox nell'interfaccia “add”).

MANUALE UTENTE DELLA GUI
In alto ci sono due menù a tendina dall'uso intuitivo. Lo stesso dicasi per i due bottoni immediatamente sotto.
Per facilitare l'utilizzo, quando si passa sopra un bottone compare un piccola etichetta.

GUI per aggiungere un bagaglio

- Il bottone con la “+” rossa serve per aggiungere il bagaglio.
- Il bottone affianco serve per resettare tutti i text-field.
- Ogni input è controllato secondo le regole di una precisa espressione regolare. Se l'input la viola, allora l'errore viene segnalato.

GUI per visualizzare i bagagli

- Il bottone in basso a sinistra serve per aggiornare le modifiche alla lista(attenzione, non è un salvataggio sul file)
- Il secondo serve per rimuovere l'ultimo bagaglio inserito
- Il terzo per i vari tipi di ricerca
- Il quarto per svuotare la lista.

SPECIFICHE TECNICHE
Sistema operativo di sviluppo: Linux Mint 17.1 Rebecca
Versione di Qt: 5.0.2 GCC 32 bit
Compilatore: GCC 4.8.4