ESAME DI FONDAMENTI DI INFORMATICA T-2 del 17/01/2018
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE e CARTELLA: CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE : CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)

Un'azienda di Quality Assurance deve, per lavoro, effettuare decine di misure di precisione, da cui estrarre valori statistici utili per le relazioni tecniche. A tal fine ha commissionato un'applicazione che elabori le misure (con i relativi dettagli) estraendone le statistiche richieste, personalizzabili con vari filtri, e li mostri graficamente.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

In generale, per verificare la qualità e la corrispondenza di un prodotto alle specifiche si effettuano una serie di misure, elaborandone poi le risposte in modo da estrarne uno o più parametri significativi (*indicatori*), che devono rientrare entro limiti prestabiliti (*tolleranze*). Esempi: percentuale di oggetti che rientrano entro l'1% nelle specifiche di peso o di lunghezza, percentuale di oggetti scartati perché troppo grandi, scoloriti, etc.

Nel caso in questione interessa soltanto il peso del prodotto, che deve rientrare entro i seguenti limiti:

- fino a 50g di peso: tolleranza del 9% del peso [quindi, fra 0 e 4.5g]
- da 50 a 100g: tolleranza fissa di 4.5g
- da 100 a 200g: tolleranza del 4.5% del peso [quindi, fra 4.5g e 9g]
- da 200 a 300g: tolleranza fissa di 9g
- da 300 a 500g: tolleranza del 3% del peso [quindi, fra 9g e 15g]
- da 500 a 1000g: tolleranza fissa di 15g
- oltre 1000g di peso: tolleranza del 1.5% del peso [quindi, da 15g in su, in proporzione al peso]

Si tratta chiaramente di una funzione continua a tratti, che consente (giustamente) tolleranze un po' maggiori sulle piccole confezioni, dove sarebbe difficile dosare il prodotto al grammo, e via via la riduce per le confezioni più grandi.

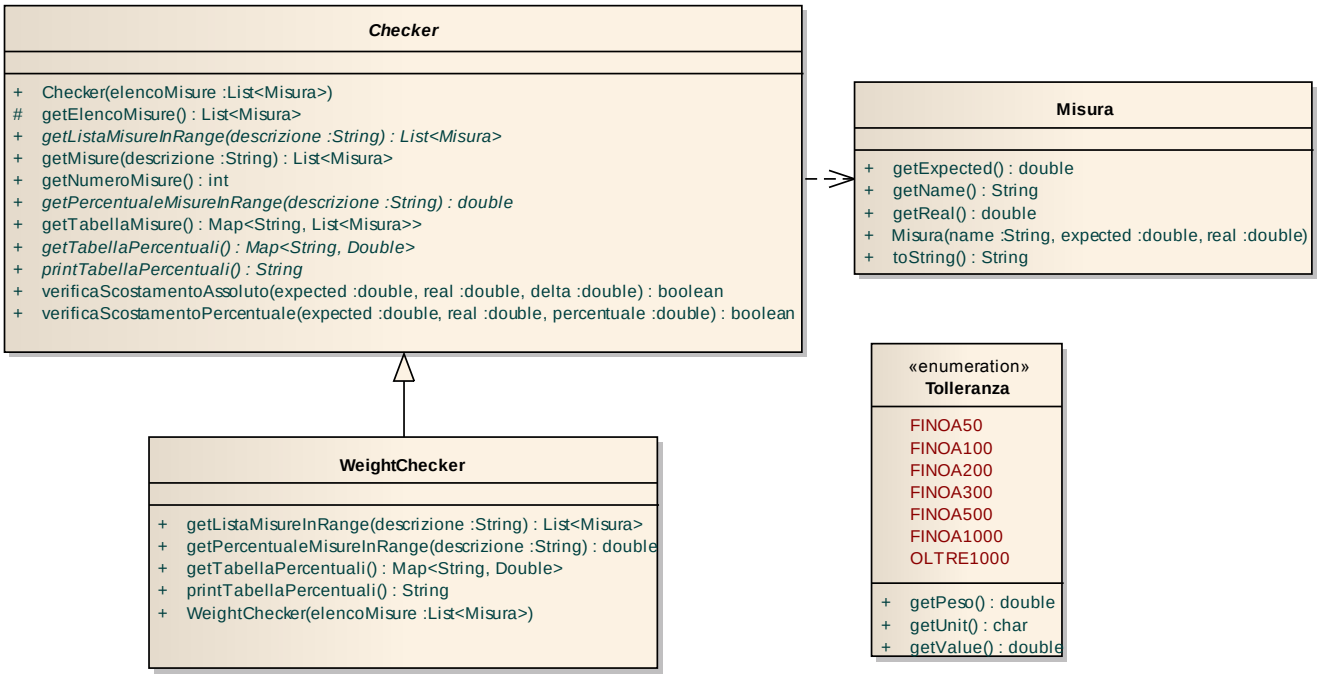
Ai fini della tutela del consumatore, però, interessano solo le confezioni **sotto-dosate**: se una confezione contiene più prodotto del previsto, infatti, l'azienda ci rimette ma il consumatore non ha di che lamentarsi.

ESEMPIO
Si supponga di aver misurato il peso di decine di confezioni di pasta o biscotti, uscite da una catena di produzione: nominalmente dovrebbero essere da 100g, ma di fatto potrebbero contenere 99.80g, 98.85g, 101.16g, di prodotto. **Quelle di peso inferiore** non devono superare una certa percentuale: altrimenti, l'azienda non soddisfa i requisiti di qualità richiesti.

Il file di testo Misure.txt contiene i risultati delle misure (nel formato dettagliato più oltre), una per riga.

Parte 1 (punti: 18)

Dati (package qa.model) *(punti: 14)*



SEMANTICA:

a) L'enumerativo ***Tolleranza*** (fornito) definisce le sette costanti che catturano i casi della funzione continua a tratti: ogni valore dell'enumerativo incorpora il peso limite e la tolleranza corrispondente (in percentuale o in grammi, secondo i casi); sono forniti tre metodi accessor per recuperare peso, valore e unità di misura.

b) La classe ***Misura*** (fornita) rappresenta il risultato di una misura, caratterizzata dal nome del prodotto misurato e, rispettivamente, dal valore atteso e da quello effettivamente misurato;

c) la classe astratta ***Checker*** (fornita) incapsula i dati e l'algoritmo per verificarne la conformità. Il costruttore riceve un elenco di misure e lo memorizza internamente, popolando anche una mappa ausiliaria avente per chiave i nomi dei prodotti e per valori la lista di misure relative a tale prodotto. Sono forniti opportuni accessor per recuperare l'elenco di tutte le misure (*getElencoMisure*), il numero totale di misure presenti (*getNumeroMisure*), l'elenco delle sole misure relative a un certo prodotto (*getMisure*) e la mappa ausiliaria suddetta (*getTabellaMisure*).
Sono inoltre forniti i due metodi *verificaScostamentoPercentuale* e *verificaScostamentoAssoluto* che confrontano un valore atteso rispetto a quello effettivo e restituiscono ***true*** se il valore effettivo è entro il range previsto, applicando la semantica descritta nel dominio del problema (OVVERO: sono fuori range solo le confezioni **sotto-dosate** che eccedono la tolleranza prevista per il loro peso). I due metodi si differenziano solo per il criterio di confronto, che è percentuale nel primo caso e assoluto nel secondo.

> Ad esempio, se una confezione da 500g ne pesa in realtà 490, *verificaScostamentoPercentuale(500,490,3%)* restituisce ***true*** (3% di 500g = 15g), mentre *verificaScostamentoAssoluto (500,490,5)* restituisce ***false***, perché i 10g mancanti non rientrano nella fascia fissa di 5g specificata.
> Al contrario, se una confezione da 250 g ne pesa in realtà 252, entrambi i metodi restituiscono ***true*** perché il prodotto in eccesso non è mai un problema per il consumatore.

Rimangono astratti i quattro metodi *getTabellaPercentuali*, *printTabellaPercentuali*, *getListaMisureInRange* e *getPercentualeMisureInRange*.

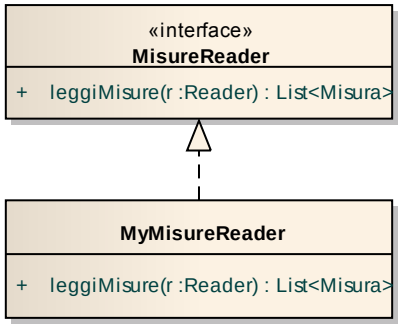
d) la classe ***WeightChecker*** (da realizzare) specializza ***Checker*** per le misure di peso secondo i criteri di tolleranza espressi dall'enumerativo ***Tolleranza.*** Il costruttore riceve lo stesso elenco misure della classe base, a cui delega il popolamento delle strutture dati. I quattro metodi da concretizzare devono:

- *getTabellaPercentuali*: restituire una mappa <String, Double> che, per ogni prodotto, riporta la percentuale di confezioni che soddisfano il criterio di qualità;
- *printTabellaPercentuali*: restituire una rappresentazione in forma di stringa di tale mappa, *in cui le percentuali sono correttamente formattate come tali, con due cifre decimali;*
- *getListaMisureInRange(String descrizione)*: restituire la lista delle misure in range del prodotto specificato. [Suggerimento: fattorizzare la logica di verifica in un metodo ausiliario privato]
- *getPercentualeMisureInRange(String descrizione)* : restituisce la percentuale di misure in range del prodotto specificato.

Persistenza (qa.persistence) *(punti 4)*

Il file di testo Misure.txt contiene una misura per riga: nell'ordine troviamo la descrizione del prodotto (chiave univoca), il peso nominale e quello reale, separati da virgole

```
Spicchi di luna, 250, 247
Spicchi di luna, 250, 245
Spicchi di luna, 250, 255
Rigatoncini, 500, 490
Rigatoncini, 500, 508
Rigatoncini, 500, 483
...
```



SEMANTICA:

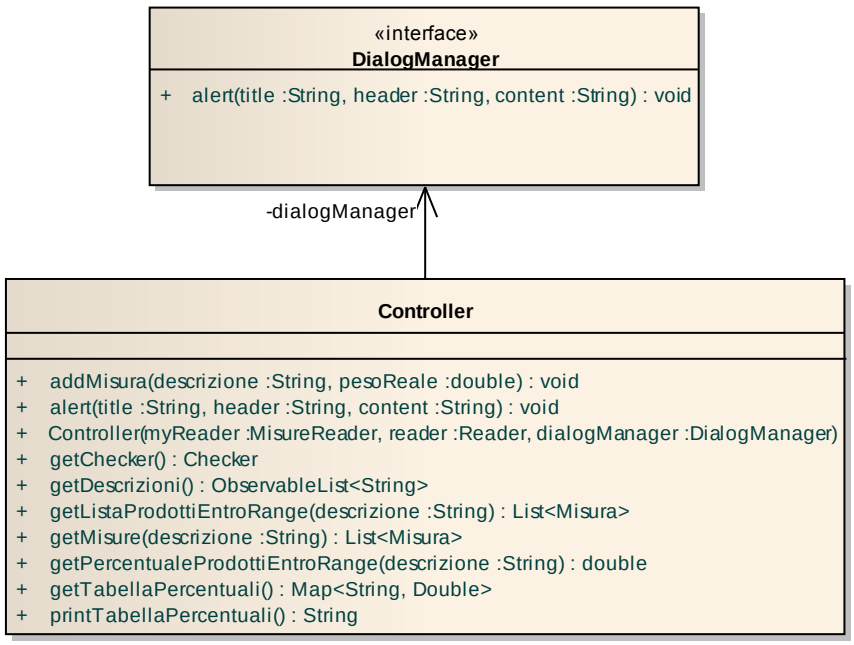
a) L'interfaccia ***MisureReader*** (fornita) dichiara il metodo *leggiMisure* che restituisce una lista di ***Misura*** lette dal *reader* ricevuto come argomento; in caso di problemi di I/O, il metodo propaga l'opportuna ***IOException***, mentre eventuali problemi nel formato del file sono incapsulati in ***BadFileFormatException***.

b) La classe ***MyMisureReader*** (**da realizzare**) implementa ***MisureReader*** secondo tali specifiche.

Parte 2 (punti: 12)

Controller (qa.ui.controller)

Il Controller è fornito già pronto ed è organizzato secondo il diagramma UML in figura.



SEMANTICA:

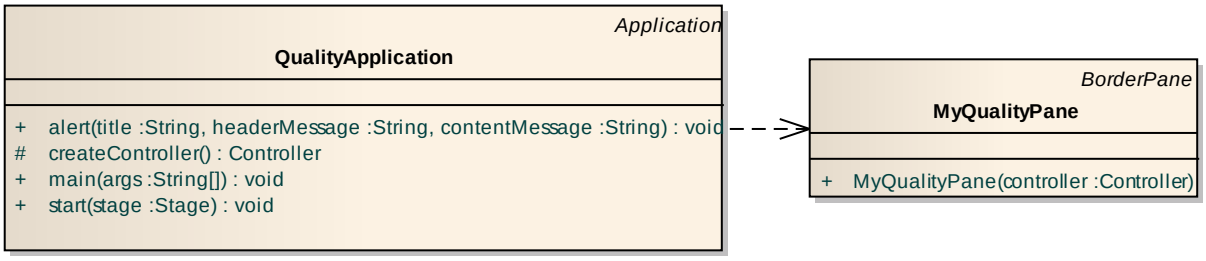
La classe ***Controller*** (fornita) fornisce una serie di metodi che richiamano gli analoghi metodi del checker; in più, il metodo ausiliario *alert* può mostrare avvisi all'utente.

Il metodo *addMisura* [utile solo per la versione Swing completa] consente di aggiungere una misura alla tabella interna del checker, specificandone semplicemente descrizione e peso attuale (il peso atteso, uguale per tutti i prodotti dello stesso tipo, viene recuperato automaticamente).

Interfaccia Utente JavaFX (qa.ui.javafx) **per studenti A.A. 2016/17** *(punti 12)*

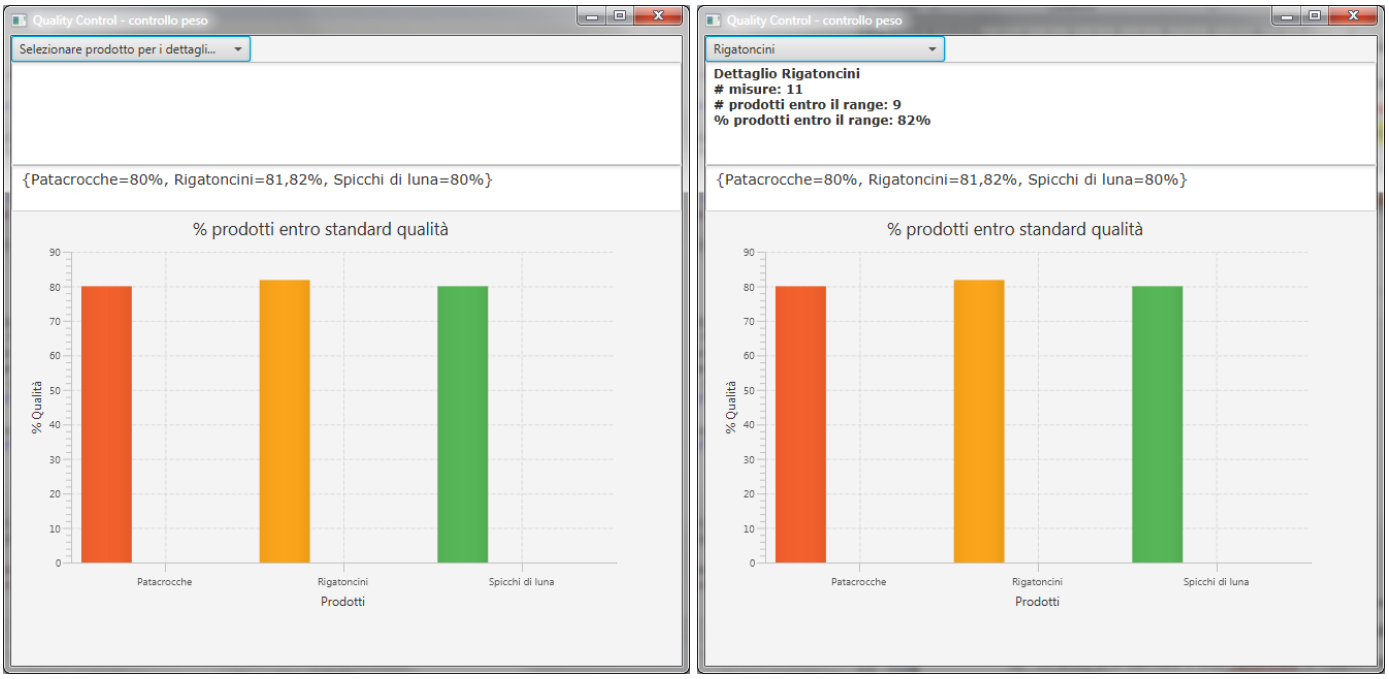
L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nelle figure seguenti.

L'architettura segue il modello sotto illustrato:



La classe ***QualityApplication*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire il file, il controller e incorporare l'apposita istanza di ***MyQualityPane*** (**da realizzare**). Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***QualityApplicationMock***.

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente.

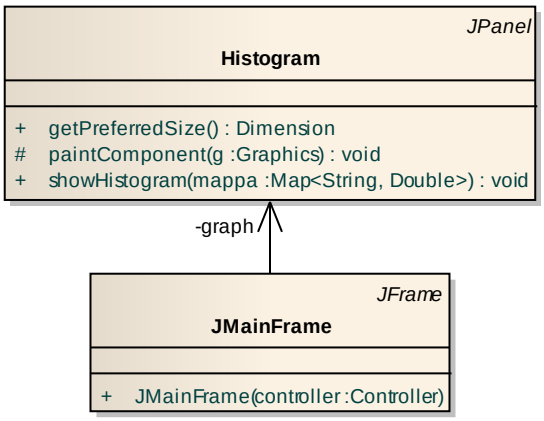


La classe ***MyQualityPane*** (**da realizzare**) deve estendere ***BorderPane***.

1) In alto, una ***Combo*** e due ***TextArea*** di opportuna dimensione consentono rispettivamente di scegliere quale prodotto approfondire, mostrarne il dettaglio, e visualizzare la tabella percentuali relativa al grafico mostrato sotto - *che non cambia nel tempo.*
2) Al centro un ***BarChart*** (senza legenda) mostra la distribuzione percentuale dei prodotti che superano il test di qualità, in forma grafica.
3) Quando l'utente seleziona un diverso elemento dalla combo, il dettaglio corrispondente nella prima textarea viene aggiornato.

Interfaccia Utente Swing (qa.ui.swing) **per studenti A.A. 2015/16 e precedenti** *(punti 12)*

In questo compito è offerta allo studente la possibilità di realizzare un'interfaccia grafica semplificata (7 punti) o completa (12 punti)



L'interfaccia utente <u>semplificata</u> deve essere simile all'esempio mostrato nella figura seguente:

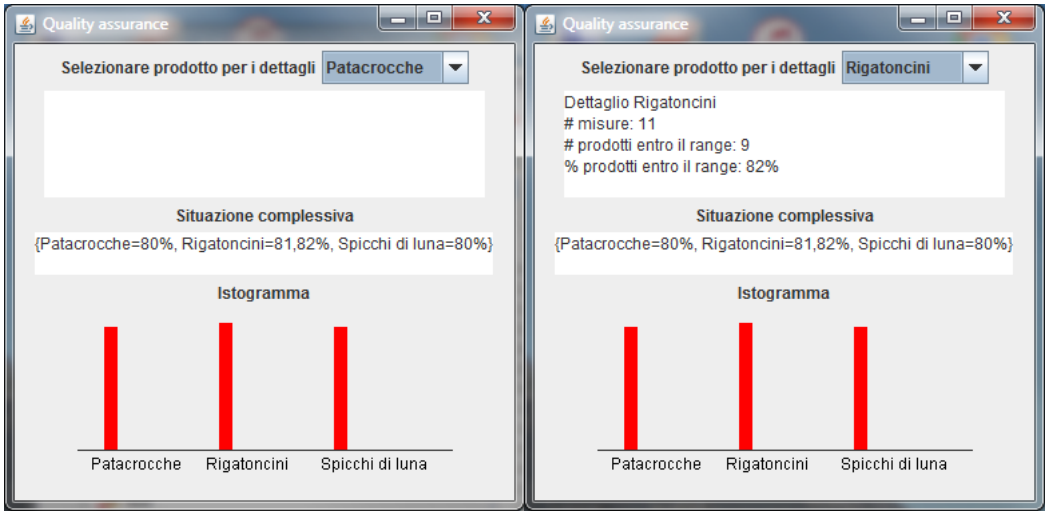


Fig. 1 Fig. 2

- La classe ***Program*** (fornita) contiene il *main* di partenza dell'intera applicazione, che si occupa di aprire il file e creare tutto il necessario. <u>Per consentire di collaudare la GUI anche in assenza della parte di persistenza</u>, è possibile avviare l'applicazione mediante la classe ***GUITest***.
- La classe ***Histogram*** (fornita) implementa un pannello-istogramma, in grado di mostrare un grafico a barre; prevede il solo metodo pubblico *showHistogram* che riceve i dati da mostrare sotto forma di mappa<String,Double>.

La classe ***JMainFrame*** (***da realizzare***) deve organizzare l'interfaccia come sopra illustrato (Fig. 1), ovvero:

1) in alto, una label seguita dalla combo con tutte le descrizioni dei prodotti;
2) subito sotto, un'area di testo (non editabile) per mostrare i dettagli del prodotto;
3) più sotto, un'altra label seguita da una seconda area di testo (anch'essa non editabile) che mostri la situazione complessiva;
4) Infine, in basso, una label e di seguito l'istogramma (fisso) con la situazione complessiva.

A ogni selezione della combo (Fig. 2), l'applicazione deve reagire mostrando i dettagli del prodotto: <u>in questa versione semplificata</u>, la situazione complessiva e il grafico rimangono *immutati*.

L'interfaccia utente completa (valore: 12 punti) dev'essere invece come nell'esempio della figura seguente.

Rispetto alla GUI semplice, questa versione (Fig. 3) ha in più una riga di controlli al centro, costituita da un bottone ("*Nuova misura*") e due campi di testo, che consentono di *aggiungere dinamicamente una nuova misura* non presente nel file, causando il corrispondente *l'aggiornamento dei dettagli, delle percentuali e del grafico.*

Inizialmente (Fig. 3), il bottone è abilitato mentre i due campi di testo sono disabilitati.

Premendo il pulsante (Fig. 4), il campo di testo più a destra si abilita, mentre nel primo viene ricopiata l'attuale selezione della combo (nell'esempio, le Patacrocche) e il pulsante viene disabilitato. Ciò consente all'utente di inserire il peso (attuale) della nuova misura nel textfield attivo (Fig. 5), confermandola poi con INVIO sulla tastiera.

Tale conferma causa (Fig. 6) l'aggiunta immediata della nuova misura nel model (tramite il metodo *addMisura* del controller) e il conseguente ricalcolo dei dettagli, delle percentuali e del grafico. Contemporaneamente, il pulsante viene ri-abilitato, il campo di testo centrale svuotato e il campo di testo di destra ri-disabilitato, riportando così questa riga alla situazione iniziale.

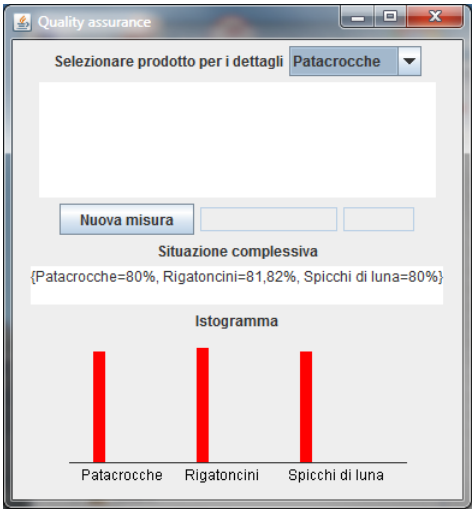


Fig. 3

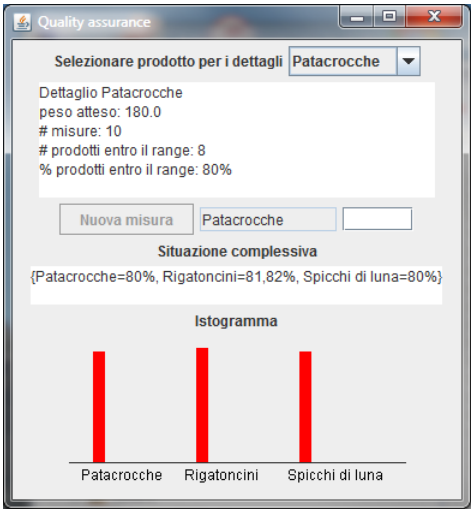


Fig. 4

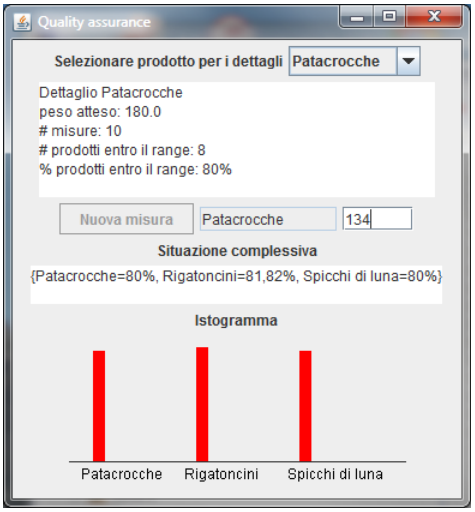


Fig. 5

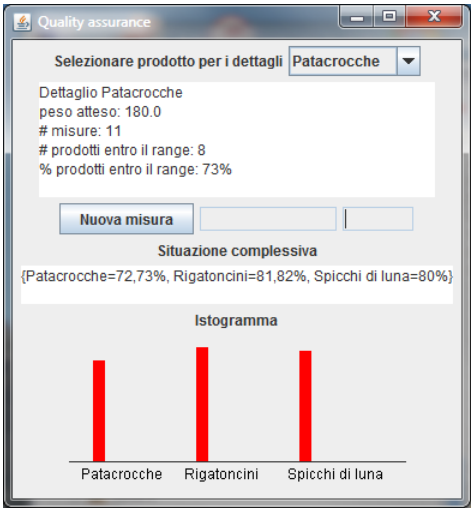


Fig. 6