Relazione Progetto Ingegneria del Software

Nome: Simone **Cognome**: Odierna **Matricola**: 1000015369

Introduzione breve

Si vuole realizzare un software che gestisca il funzionamento di una vending machine (distributore automatico). Sarà in grado di effettuare le operazioni di inserimento del credito, scelta del prodotto e ritorno del resto.

Progettazione

Per la costruzione del software sono stati utilizzati i seguenti design patterns:

- **State** per poter cambiare lo stato della vending machine in base all'operazione da svolgere;
- **Prototype** che consente di clonare dei prototipi, in questo caso servirà per i singoli prodotti.

Descrizione delle classi

Sono state create delle classi per poter definire le operazioni e i prodotti disponibili:

- **VendingMachine**
  Classe principale che contiene lo stato iniziale della vending machine e una lista di prodotti disponibili per l'acquisto. Contiene un metodo **getProducts()** che ritorna questa lista.

- **Prodotto**
  Classe che implementa **Cloneable**; Definisce l'oggetto prodotto da clonare con i suoi attributi. Sono presenti metodi di **get** e **set** per questi ultimi.

- **Prodotti**
  Si occupa della creazione della lista di prodotti da inserire nella vending machine; il metodo **addProduct(int id, String name, float price, int quantity)** consente di aggiungere e/o modificare un prodotto già esistente. **modQuantity(int id)** gestisce la quantità

del singolo prodotto ogni volta che questo viene acquistato. È anche presente un metodo per stampare a schermo la lista con tutti i prodotti attualmente disponibili, **printList()**.

- **Monete**
  Svolge l'operazione di inserimento delle monete; il metodo **insertMoney(float val)** controlla anche che la moneta che si vuole inserire sia tra quelle valide (inserite all'interno della lista **acceptedCoins**).

- **Selection**
  Utilizzata per l'operazione di selezione del prodotto; viene controllato che quest'ultimo sia disponibile e che il credito sia sufficiente tramite il metodo **selectProduct(int id, VendingMachine vm)**. Vengono gestite le eccezioni di **NullPointerException** e **NoSuchElementException** nel caso in cui il numero inserito non corrisponda a nessuno dei prodotti disponibili. È anche presente un metodo **abort()** utilizzato nel caso in cui l'utente abbia bisogno di annullare l'operazione.

- **Resto**
  La classe contiene il metodo **giveChange()** che gestisce il ritorno del resto all'utente.

Interfaccia utente

La classe **App** contiene il **main()**; viene gestita l'interfaccia grafica per far interagire l'utente con l'API della vending machine. Sono stati creati due metodi:

- **clearScreen()** pulisce lo schermo;
- **abort()** richiama la funzione di abort per annullare un'operazione.

L'interfaccia si presenta come un menù che richiede un input dall'utente per effettuare delle scelte tra quelle proposte; ad esempio, nella fase di selezione, si potrà scegliere se continuare con la scelta del prodotto da acquistare o annullare l'operazione.

Software testing

Il software è stato testato, con esito positivo, con delle apposite funzioni di testing che coprono le principali e più importanti operazioni:

- **@Test public void checkProductsLoad()**
  Controlla che i prodotti siano caricati all'interno della vending machine;

- **@Test public void checkGetList()**
  Si assicura che la lista caricata sia una **Hashtable**;

- **@Test public void checkGetAndAddProduct()**
  Verifica che i prodotti vengano aggiunti e trovati correttamente all'interno della lista;

- **@Test public void checkModQauntity()**
  Controlla che la quantità disponibile per un certo prodotto venga diminuita ad ogni acquisto;

- **@Test public void checkCoinInsert()**
  Si assicura che le monete vengano inserite nel modo corretto all'interno della vending machine;

- **@Test public void checkProductSelection()**
  Controlla che nella fase di selezione non venga selezionato un prodotto non esistente nella lista;

- **@Test public void checkProductSelectionAbort()**
  Verifica che, una volta chiamata la funzione di **abort()**, lo stato della macchina torni a quello iniziale; cioè l'inserimento delle monere;

- **@Test public void checkChange()**
  Si assicura che il resto venga calcolato nel modo corretto dopo un acquisto;

- **@Test public void checkStateChange()**
  Controlla che il passaggio degli stati avvenga correttamente.