Assigment 2

Descrizione del progetto

Un negozio di elettronica ha deciso di richiedere la realizzazione di un programma per il calcolo del prezzo totale delle ordinazioni on-line.

Il programma deve soddisfare i seguenti requisiti:

1. Dato un elenco di articoli (Processori, Schede Madri, Tastiere, Mouse) calcolare il totale;
2. Se vengono ordinati più di 5 Processori viene fatto uno sconto del 50% sul prezzo del Processori meno caro;
3. Se vengono ordinati più di 10 Mouse il meno caro viene regalato;
4. Se vengono ordinati lo stesso numero di Mouse e Tastiere viene regalato l'articolo meno caro.
5. Se l'importo totale degli articoli supera i 1000 euro viene fatto uno sconto del 10% sul totale;
6. Non è possibile avere un'ordinazione con più di 30 elementi (se accade prevedere un messaggio d'errore);
7. Se l'importo totale è inferiore a 10 € viene aggiunta una commissione di 2 €;
8. Prevedere la possibilità di regalare, in modo casuale, 10 ordini effettuati dalle 18:00 alle 19:00 da utenti minorenni differenti.

Interfaccia e oggetti

Sviluppare il progetto implementando la seguente interfaccia (molto simili a quelle viste nell'ultima parte del laboratorio 4)

```
package it.unipd.mtss.business;

import java.util.List;

import it.unipd.mtss.business.exception.BillException;
import it.unipd.mtss.model.MenuItem;

public interface Bill {

    double getOrderPrice(List<EItem> itemsOrdered, User user) throws
RestaurantBillException;

}
```

- La classe **EItem** (simile alla calsse Product dell'esempio visto nel laboratorio su JUnit) rappresenta un articolo dell'ordinazione e dovrà avere le seguenti caratteristiche:
  - **itemType**: enumerazione che rappresenta i tipi di elementi presenti nel menu (Processor,

    Motherboard, Mouse, Keyboard)
    - **name**: nome del articolo
    - **price**: prezzo unitario dell'articolo
- La classe **User** contiene l'identificativo e i dati anagrafici del utente reigistrato

Svolgimento

Sviluppare il progetto seguendo la pratica della Continuous Integration, pertanto si richiede di svolgere l'assignment nel seguente ordine:

- Predisposizione del repository github
- Definizione delle attività nel Issue Traking System
    - Suddividere le attività tra i due sviluppatori
- Creazione del progetto utilizzando l'archetipo maven quick start
- Configurazione del processo di build
- attivazione e configurazione dell'analisi statica
- Predisposizione della Continuous Integration con GitHub Actions
- Sviluppo del codice sorgente e dei test

Di seguito vengono specificate in dettaglio le configurazioni dei vari strumenti.

Issue Traking System

Gestire il progetto nel issue tracking system di github:

- registrare la nuova versione (vedi laboratorio ITS: Iterazioni o Milestone)
- creare la project board (vedi laboratorio 1: Bacheche (Project Board))
- creare le attività per la realizzazione del assigment 2
- suddividere le attività tra i due sviluppatori

Version Control System

Sviluppare il progetto utilizzando un repository git pubblico in Github.

Le attività possono essere gestite utilizzando il work flow *Feature Branch*.

Le attività devono essere sviluppate (compresi i test di unità) una per volta senza tenere in considerazione le attività successive.

Project Automation

- Creare il progetto con Maven utilizzando l'archetipo: `maven-archetype-quickstart`

- Utilizzare i seguenti parametri:
  - **groupId**: it.unipd.mtss
  - **artifactId**: e-shop-manager

Configurare il Build Lifecycle in modo da:

- eseguire la compilazione del progetto (fase compile)
- eseguire i test di unità (fase test)
- eseguire l'analisi statica del codice con checkstyle nella fase **package** (Vedi laboratorio Maven: Plugin - Universal reuse of business logic).

Test di Unità

- Sviluppare i test di unità per arrivare ad una copertura del code >= 85% dei sorgenti di produzione
- I test di unità devono essere sviluppati con il framework JUnit 4 o 5 e seguire le convenzioni Maven (vedi qui e qui)
- Le firme dei metodi di test devono essere parlanti. Vedi il formalismo riportato nel laboratorio JUnit. I test devono essere sviluppati seguendo il pattern Arrange/Act/Assert (AAA)
- Creare dei test di unità per soddisfare le caratteriste **A-TRIP** e **Right Bicep** viste a lezione. È richiesta la copertura del codice >= 85%.
- [Opzionale] sviluppare il progetto utilizzando l'approccio TDD

Analisi statica del codice

Configurare il plugin maven checkstyle in modo da effettuare le seguenti verifiche al codice di produzione (creare il file `checkstyle.xml` e configurare i seguenti moduli):

- BooleanExpressionComplexity
- CyclomaticComplexity
- FileLength
- LineLength
- MethodLength
- EmptyCatchBlock
- FileTabCharacter
- AvoidStarImport
- IllegalImport
- NeedBraces
- Header con il seguente valore (sostituire le variabili con i vostri dati)

```
//////////////////////////////////////////////////////////////////
// [NOME1] [COGNOME1] [MATRICOLA1]
// [NOME2] [COGNOME2] [MATRICOLA2]
//////////////////////////////////////////////////////////////////
```

Configurare il plugin in modo da far fallire la build se non vengono rispettati i controlli configurati.

Continuous Integration

- Attivare la continuous integration con GitHub Action.
- Pubblicare il risultato del processo di CI nella pagina README.md del progetto
- Attivare il caloclo del code coverage

> **TIP** vedi Jacoco e Coveralls

- [Opzionale] Pubblicare il risultato dell'analisi statica e del code coverage nella pagina README.md del progetto

Modalità di consegna

- L'assignment deve essere **completato e consegnato entro il 23/05/2022 alle 18:00**. Ogni consegna e modifica successiva non sarà valutata.
- Il codice prodotto deve essere rilasciato nel ramo **master** o **main**
- La consegna deve essere fatta tramite l'apposito form che verrà pubblicato nel sito del corso