TP 1 JUnit

Introduction

Les **tests unitaires** sont une pratique essentielle dans le développement logiciel visant à garantir la qualité, la fiabilité et la maintenabilité du code. L'objectif principal des tests unitaires est de vérifier si chaque unité individuelle d'un programme fonctionne correctement de manière isolée. Cela permet de s'assurer que chaque composant du logiciel fait ce qu'il est censé faire, indépendamment du reste du système.

JUnit est l'un des frameworks de test unitaire les plus populaires pour les applications Java. Il fournit un ensemble de annotations et de méthodes pour faciliter l'écriture et l'exécution de tests unitaires.

Exercice 1 :

Dans cet exercice, nous allons mettre en place un système de gestion de monnaie. Nous souhaitons permettre l'ajout d'une somme d'argent dans un portefeuille (wallet) avec une devise spécifique parmi trois options : EUR, USD et TND. Les étapes suivantes doivent être suivies :

Première partie :

1. **Création d'un projet Maven et intégration des bibliothèques :**
   a. Commencez par créer un nouveau projet Maven.
   b. Intégrez les bibliothèques nécessaires.

```
<dependencies>
    <!-- JUnit for testing -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.9.0</version>
        <scope>test</scope>
    </dependency>

    <!-- Apache PDFBox for PDF generation -->
    <dependency>
        <groupId>org.apache.pdfbox</groupId>
        <artifactId>pdfbox</artifactId>
        <version>2.0.27</version>
    </dependency>

    <!-- Apache POI for Excel -->
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi</artifactId>
        <version>5.2.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
        <version>5.2.0</version>
    </dependency>
</dependencies>
```

Lien pour consulter les bibliothèques et leurs versions : https://mvnrepository.com

2. **Création d'une énumération pour les devises (currency) :**
    a. Définissez un type énumération (enum) pour représenter les trois devises : EUR, USD, et TND.

```
    public enum Currency {
        USD("US Dollar"),
        EUR("Euro"),
        TND("Tunisian Dinar");

        private final String displayName;

        Currency(String displayName) {
            this.displayName = displayName;
        }

        public String getDisplayName() {
            return displayName;
        }
    }
```

3. **Création de la classe ExchangeRate :**
    a. Implémentez la classe ExchangeRate, qui aura pour responsabilité l'extraction du taux de change entre une devise de base et une devise cible.
    b. Assurez-vous que cette classe respecte la logique du taux de change.

```
public class ExchangeRate {
   private Map<String, Map<String, Double>> ratesMap;

    public void addExchangeRate(String baseCurrency, String
targetCurrency, double exchangeRate) {
       ratesMap
                       .computeIfAbsent(baseCurrency, k -> new
HashMap<>())
                .put(targetCurrency, exchangeRate);
   }
```

```
    public Double getExchangeRate(String baseCurrency, String
targetCurrency) {
        return ratesMap
                .getOrDefault(baseCurrency, new HashMap<>())
                .get(targetCurrency);
    }
}

public void initExchangeRates(){

    this.addExchangeRate("USD", "EUR", 0.93);
    this.addExchangeRate("USD", "TND", 3.13);

    this.addExchangeRate("EUR", "USD", 1.07);
    this.addExchangeRate("EUR", "TND", 3.63);

    this.addExchangeRate("TND", "USD", 0.32);
    this.addExchangeRate("TND", "EUR", 0.30);

    //printAllExchangeRates();
}
```

4. **Création de la classe Wallet :**
    a. Implémentez la classe Wallet, qui gèrera l'ajout d'une nouvelle somme d'argent au portefeuille avec une indication de la devise correspondante.
    b. Chaque portefeuille doit être associé à sa propre devise, définie lors de la création. La classe doit également permettre l'élimination d'un montant.

```
public class Wallet {
    private String walletRef;
    private double amount;
    private Currency currency;
```

```
public String getWalletRef() {
        return walletRef;
    }
     public void addAmount(double amount, Currency currency,
                    ExchangeRate exchangeRateMap) {
        //code
    }

         public double getAmount(Currency targetCurrency,
                    ExchangeRate exchangeRateMap) {
        //code
    }
}
```

Deuxième partie :

Dans la deuxième partie de l'exercice, nous aborderons la création des classes de test pour chaque classe (ExchangeRate et Wallet) afin de tester toutes les fonctionnalités. Chaque test devra lire les données à partir d'un fichier Excel. Une fois les tests terminés, un fichier PDF devra être généré, contenant le statut de chaque test (OK ou KO) ainsi que des indications sur la réussite ou l'échec de chaque test.

Troisième partie :

Dans cette partie, il faut envoyer des messages automatiques lorsque le test est terminé sur Slack

```
    <dependency>
       <groupId>com.github.seratch</groupId>
       <artifactId>jslack</artifactId>
       <version>3.4.2</version>
    </dependency>
```

```
public class SlackIntegration {
   private static String webHookUrl = "*********";
   private static String oAuthToken = "*********";
   private static String slackChannel = "*******";


public   static   void   sendMessage(String   message,   PDDocument
pdfDocument) throws Exception{
                                        Payload      payload      =
Payload.builder().channel(slackChannel).text(message).build();
                              WebhookResponse    webhookResponse    =
Slack.getInstance().send(webHookUrl, payload);
   }

}
```

En résumé, cet exercice consiste à mettre en place un système de gestion de monnaie avec des fonctionnalités spécifiques, suivies d'une phase de test approfondi avec des données lues depuis un fichier Excel, et enfin la génération d'un rapport au format PDF résumant les résultats des tests effectués.

Chargés TP : Mohamed Amine GUESMI
Chargé de cours et responsable pédagogique : Hatem BEN STA