Kata : Troupe théâtrale

Binôme :

Alycia Belkacem

Dahbia Moussi

Partie 1 :

L'utilisation de StringBuffer garantit que les opérations de modification de chaînes soient synchronisées.

Ajout d'un contrôle sur le type de pièce à la création de l'objet : créations des variables statiques pour les types de pièces de théâtre (`Tragedy` et `Comedy`) dans la classe Play, la méthode statique `createPlay()` est utilisée pour créer des instances de la classe Play en vérifiant d'abord si le type de pièce est valide pour lever une exception de type `Error` en cas de type de pièce inconnu. Cela permet de contrôler la création d'instances de Play tout en maintenant le constructeur privé, ce qui est une approche courante pour garantir l'intégrité des objets créés.

Gérer les sommes en flottant au lieu d'entiers /100 : Pour gérer les sommes en dollars avec des décimales, les principales modifications apportées à la classe `StatementPrinter` incluent l'utilisation de double pour les montants et le changement des valeurs des montants pour utiliser des valeurs en virgule flottante. Cela permet de gérer les montants de manière plus précise en évitant les erreurs d'arrondi associées à l'utilisation d'entiers pour les montants en centimes. Les valeurs sont mises à l'échelle pour correspondre au format de devise, mais elles sont stockées en tant que nombres à virgule flottante.

On a renommé la variable frmt en currencyFormatter pour plus de clarté.

Extraction des sous-méthodes de la méthode print() pour gérer des parties spécifiques du calcul `calculateVolumeCredits()`, `calculateAmount()` cela permettra de mieux organiser le code et de le rendre plus modulaire. Chaque sous-méthode a une responsabilité spécifique.

Partie 2 :

Ajout des méthodes `toHTML()` et `toText()` à la classe `Invoice`, et modifier la classe `StatementPrinter` pour supporter le rendu HTML.

Ces méthodes utiliseront la méthode `render` pour générer le rendu `HTML` et la méthode `print()` de la classe `StatementPrinter` pour générer le rendu texte.

Réduction de la duplication de code : Les méthodes `toHTMLFile` et `toTEXTFile` contiennent une duplication de code importante. On a extrait la logique commune dans une méthode privée, par `exportToFile`, pour éviter la duplication.

Partie 3 :

Pour anticiper l'ajout de types de représentation dans le futur, on a choisi d'utiliser des classes abstraites et des sous-classes. Afin de permettre une extensibilité plus facile de notre modèle de données.

Lorsqu'un nouvel objet Play est créé avec un type autre que ceux spécifiés dans les sous-classes (`Tragedy` et `Comedy`), une `Error` sera déclenchée au moment de la création de l'objet, car la vérification sera effectuée dans la classe abstraite. Cela permet de garantir que seuls les types de représentation connus sont autorisés (On a gardé le contrôle sur la création de l'objet de la partie 1).

Dans les sous-classes (`Tragedy` et `Comedy`), ajout d'un constructeur qui utilise `super` en spécifiant le type appropri.

Partie 4 :

Création d'une classe Customer avec les propriétés suivantes : nom du client, numéro de client et solde de points de fidélité.

Modification la classe `Invoice` pour utiliser l'objet `Customer` au lieu d'une chaîne de caractères pour représenter le client.

Pour éviter le chaînage direct des opérations dans les méthodes `print` et `render` sur `invoice.customer.loyaltyPoints`, on a stocké le client dans une variable avant de déduire les points de fidélité.

Ajout d'une méthode `deductLoyaltyPoints` dans la classe Customer, qui déduit un montant spécifié de points de fidélité du solde du client. Qui est ensuite utiliser les méthodes print et render pour déduire les points de fidélité du client lorsque la réduction de 15€ est appliquée.

Tests Unitaires

Utilisation des méthodes de configuration `@BeforeEach` pour réduire la duplication du code d'initialisation.

Correction de bug :

- Initialisation de la variable `statementPrinter` dans le constructeur.
- Simplification de la méthode `calculateVolumeCredits`.
- Les variables `name` et `type` de la classe `Play` sont déclarées comme protected, elles sont désormais accessibles par les sous-classes
- `toHTMLFile` et `toTEXTFile` ont été modifiées pour prendre uniquement deux arguments : String `fileName` et String `invoice`, au lieu des trois arguments précédents. L'appel de `exportToFile` a également été ajusté en conséquence.