Contrôle terminal info1 / Semestre 2 (période 4)

R2.01 : Développement orientés objets

Nom du responsable :	**BORNE Isabelle**
Date du contrôle :	**13/06/22**
Durée du contrôle :	**1h30**
Nombre total de pages :	**3 pages**
Impression :	**Recto**
Documents autorisés :	**Cours, TD, TP**
Calculatrice autorisée :	**non**
Réponses :	**2 Copies**

Bien lire le sujet

La correction tiendra compte de la lisibilité du code, du respect des consignes de nommage en Java, du respect de l'énoncé et de la bonne indentation du code.

2 exercices à faire sur 2 copies séparées

Exercice 1 (30minutes) : 1 copie

Il s'agit de lire un fichier texte nommé **movies.txt** qui contient des titres de films classiques ainsi que leur date de sortie. Chaque film occupe 2 lignes dans le fichier : la date sur une ligne et le titre sur la suivante. Par exemple :

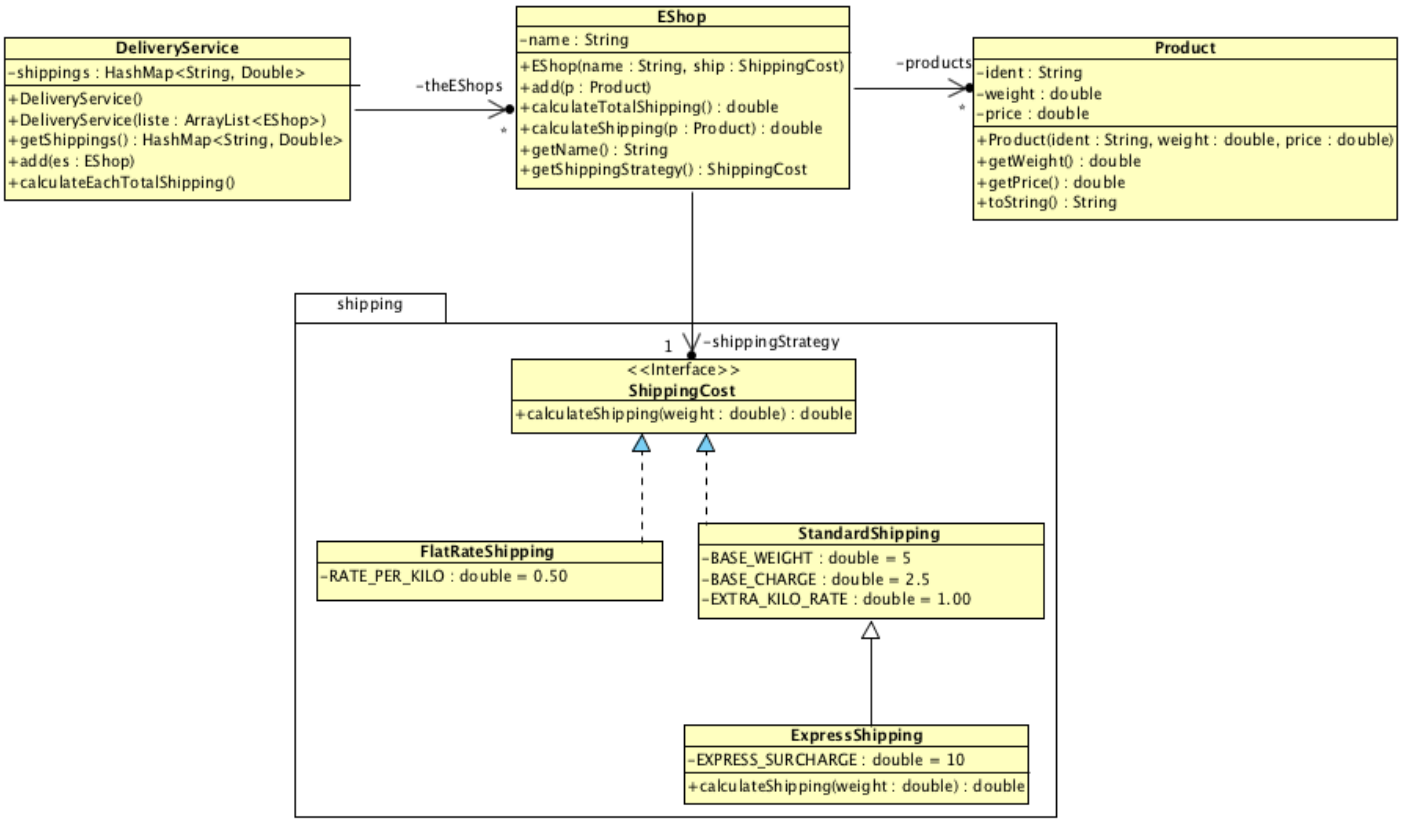
```
1942
Casablanca
1954
Fenêtre sur cour
```

Ecrire le code de la classe nommée **`ClassicMovies`** qui fait la lecture des titres de films et de leurs dates et les imprime sur la console.

Exercice 2 (1 heure) : 1 autre copie

Il s'agit de modéliser une application qui calcule des frais d'expédition pour des boutiques en ligne. L'application proposée ne cherche pas à être proche de la réalité, c'est juste un cas d'école.

Voici son diagramme de classes :



Glossaire :

Eshop : boutique en ligne	FlateRateShipping : livraison à tarif fixe
Eshop : boutique en ligne	StandardShipping : livraison standard
Shipping : expédition ou livraison	ExpressShipping : livraison express
ShippingCost : frais de livraison	

1- Description sommaire des classes

Remarques :

- Seuls les éléments essentiels sont décrits, le reste est présent dans le diagramme de classes.
- Les constructeurs peuvent lancer une `IllegalArgumenException` si nécessaire.

Le package **`shipping`** contient des stratégies de calcul des prix de livraison :

- une interface **`ShippingCost`**
- **`FlateRateShipping`** possède une constante `RATE_PER_KILO` initialisée à 0.5 qui est le taux à multiplier par un poids pour obtenir les frais d'envoi. C'est le calcul fait par la méthode **`calculateShipping`** qui prend le poids d'un produit en paramètre.

- **`StandardShipping`** a trois attributs constants ; sa méthode **`calculateShipping`** calcule un prix de livraison fixe avec `BASE_CHARGE` pour les 5 premiers kilos (`BASE_WEIGHT`), plus 1 euro pour chaque kilo supplémentaire (`EXTRA_KILO_RATE`).
- **`ExpressShipping`** a un attribut constant dont la valeur est à ajouter au prix de livraison standard pour obtenir le prix de livraison.

Classe `Product`

Une simple structure de données qui modélise un produit avec un identifiant, un poids et un prix.

Classe `EShop`

Cette classe modélise la partie expédition d'une boutique en ligne.

Elle possède un `ArrayList` d'objets de type `Product` qui sont les produits à expédier.

Le constructeur prend en paramètre le nom de la boutique et un objet de type `ShippingCost` qui représente la stratégie de livraison de la boutique.

La méthode **`add`** ajoute un produi**t** dans la liste.

La méthode **`calculateShipping`** prend un produit en paramètre et retourne son prix de livraison. Le prix de livraison est calculé à l'aide de l'attribut `shippingStrategy`.

La méthode **`calculateTotalShipping`** calcule la somme des frais de livraisons de la liste des produits de la boutique.

Classe `DeliveryService`

La classe **`DeliveryService`** représente un service de livraison utilisée par les boutiques en ligne.

Cette classe possède un **`ArrayList`**`(theEShops)` des boutiques de type **`EShop`**.

Elle possède également un **`HashMap`** (`shippings`) dont les clés sont des noms de boutiques (supposés uniques) et les valeurs associées sont les frais de livraisons totale pour ces boutiques.

Le constructeur sans paramètre se contente de créer les attributs.

Le deuxième constructeur prend en paramètre une liste de boutiques, crée les attributs et appelle la méthode de calcul qui va ajouter des éléments à l'objet **`shippings.`**

La méthode **`add`** permet d'ajouter un objet de typeEShop à la liste.

La méthode **`calculateEachTotalShipping`** calcule l'ensemble des frais de livraison pour chaque boutique et ajoute le résultat dans **`shippings`** avec la clé qui est le nom de la boutique.

2- Questions

1. Ecrire le code complet du package **`shipping`**.
2. La classe Product est simple et sera utilisée sans fournir le code. Ecrire le code de la classe **`Eshop.`**
3. Ecrire le code de la classe **`DeliveryService.`**
4. Dans un scenario on suppose que l'on a créé deux `EShop` avec des produits. Ensuite on écrit le code suivant :

```
DeliveryService ds = new DeliveryService();
ds.add(eShop1);
ds.add(eShop2);
ds.calculateEachTotalShipping ();
```

Donner ensuite le code nécessaire pour imprimer à l'écran le contenu du `HashMap` **`shippings`** de l'objet `ds.`