CENTRE SCOLAIRE SAINTE-JULIENNE

TA 13 – Les piles et files

Exercices Java – Série 9 – Enoncés

I- Mise en situation

Tu es analyste-programmeur dans une société et tu dois passer un test en langage Java. A travers une série d'exercices, tu dois comprendre et maitriser le langage Java pour obtenir la prime salariale.

II- Objets d'apprentissage

Appliquer	Transférer
• Modéliser une logique de programmation orientée objet • Déclarer une classe • Instancier une classe (objet) • Utiliser les méthodes de l'objet instancié • Traduire un algorithme dans un langage de programmation • Commenter des lignes de codes. • Tester le programme conçu	• Développer une classe sur la base d'un cahier des charges en respectant le paradigme de la programmation orientée objet (POO) • Programmer en recourant aux classes nécessaires au développement d'une application orientée objet • Corriger un programme défaillant • Améliorer un programme pour répondre à un besoin défini

Connaître

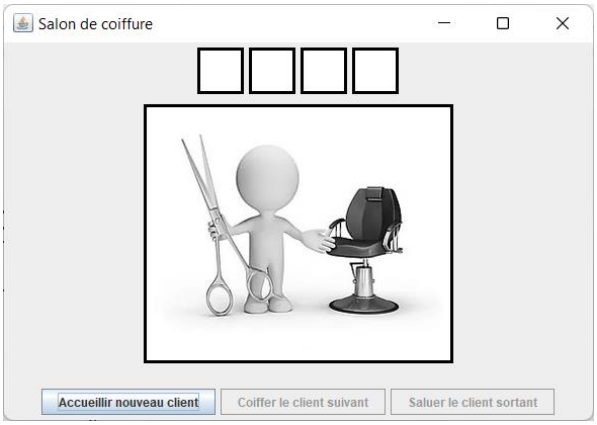
- Différencier la programmation impérative de la programmation orientée objet
- Caractériser une classe
- Décrire la création d'un objet (instanciation)
- Identifier l'instance d'une classe
- Caractériser les attributs dans une classe (encapsulation)
- Caractériser les méthodes dans une classe (encapsulation)
- Décrire la création d'un constructeur
- Différencier les types de visibilités

III- Travail à accomplir

1. Analyser l'énoncé du point IV correspondant au numéro de l'exercice demandé.
2. Modéliser en diagramme de classes l'exercice.
3. Réaliser l'exercice.
4. Commenter le travail.
5. Visualiser le travail.
6. Sauvegarder le document suivant les instructions données.
7. Imprimer le(s) document(s)

IV- Enoncés

1. Ex01 – Salon de coiffure
   Dessiner l'interface suivante:



Ecrire une classe **SalonDeCoifure** étendue JFrame qui contient:
- en 1ère ligne: un JPanel interne *panneauCanape* placé au ***Nord*** reprenant un tableau de JLabel en fonction du nombre de places sur le canapé défini lors du lancement de l'application.
- en 2ème ligne: un JPanel interne *panneauSiege* reprenant un JLabel placé au ***Centre***.
- en 3ème ligne: un bouton **Accueillir nouveau client**, un bouton **Coiffer le client suivant** et un bouton **Saluer le client sortant**. Ces boutons sont placés au ***Sud***.
    - **Accueillir nouveau client** appelle la méthode accueillirClient.
    - **Coiffer le client suivant** appelle la méthode coifferClient.
    - **Saluer le client sortant** appelle la méthode saluerClient.

Cette classe requiert les variables d'instance suivantes:
- *leCoiffeur* qui est une instance de la classe **Coiffeur**.
- *leCanape* qui est une instance de la classe **Canape**.
- *leSiege* qui est une instance de la classe **Siege**.

Et les méthodes suivantes:
- *accueillirClient()* ajoute un client sur le canapé (file) et appelle la méthode *majCanape.*
- *coifferClient()* transfère le premier client du canapé vers le siège s'il y a au moins un client sur le canapé et si le siège est libre. Si le siège n'est pas libre, il faut générer une SiegeOccupeException. De plus, cette méthode appelle également la méthode *majCanape*.
- *saluerClient()* libère le siège occupé et encaisse le client.
- *majCanape() est privée et* adapte le canapé graphique en fonction de la file.

Ecrire la classe **Coiffeur** qui contient les méthodes suivantes:
- *coiffer()* reçoit le client à coiffer (ClientSurSiege), appelle la méthode sePoser du client à coiffer et affiche un message dans la console: "Le coiffeur coiffe le client " + clientCoiffe.getNom().
- *encaisser*() reçoit le client à coiffer (ClientSurSiege), appelle les méthode seLever et payer du client à coiffer et affiche un message: "Le coiffeur encaisse le client " + clientCoiffe.getNom().

Ecrire la classe **Client** qui requiert une variable d'instance de type String *nom*, les méthodes acesseurs et la méthode toString.

Ecrire la classe **ClientSurCanape** qui hérite de la classe Client et contient les méthodes suivantes:
- *sePoser()* qui affiche: "Le client " + this.getNom() + " s'est assis sur le canapé."
- *seLever()* qui affiche: "Le client " + this.getNom() + " s'est levé du canapé."

Ecrire la classe **ClientSurSiege** qui hérite de la classe Client et contient les méthodes suivantes:
- *sePoser()* qui affiche: "Le client " + this.getNom() + " s'est assis sur le siège."
- *seLever()* qui affiche: "Le client " + this.getNom() + " s'est levé du siège."
- *payer()* qui affiche: "Le client " + this.getNom() + " paie le coiffeur."

Ecrire la classe **Canape** qui hérite de la classe LinkedList, requiert la variable d'instance nbPlaces de type byte et contient les méthodes traditionnelles et les méthodes suivantes:
- *ajouterClient()* qui reçoit un client à ajouter sur le canapé s'il y a encore de la place sinon génère une PlusDePlaceException;
- *retirerClient()* qui retire le premier client du canapé, appelle la méthode seLever de ce client s'il y a encore au moins un client sur le canapé sinon génère une CanapeVideException.

Ecrire la classe **Siege** qui requiert une variable d'instance *clientQuiEstCoiffe* de type ClientSurSiege et contient les méthodes habituelles.

Ecrire la classe **PlusDePlaceException** qui hérite de la classe Exception et contient la méthode *getMessage* où le message est: "Il n'y a plus de place sur le canapé!".

Ecrire la classe **CanapeVideException** qui hérite de la classe Exception et contient la méthode *getMessage* où le message est: "Il n'y a plus de client sur le canapé!".

Ecrire la classe **SiegeOccupeException** qui hérite de la classe Exception et contient la méthode *getMessage* où le message est: "Le siège n'est pas libre!".

Remarque:
Les images sont fournies et pour afficher une image dans un JLabel tu peux procéder comme suit:

```
JLabel jLabel = new JLabel();
ImageIcon imageIcon = new ImageIcon(getClass().getResource("coiffeur.jpg"));
jLabel.setIcon(imageIcon);
```

Changement d'image quand un client est placé sur le canapé: `placeVide.jpg` ➔ `client.jpg`.
Changement d'image quand un client est placé sur le siège: `coiffeur.jpg` ➔ `clientCoiffe.jpg`.
Sinon rétablir les images par défaut pour les deux cas.