LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2020

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI, CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :* 08/06/2020

Dans votre répertoire de travail (à définir par chaque lycée), vous trouverez un dossier nommé **EXAMEN_CI**. Renommez ce dossier en remplaçant le nom actuel par votre code de l'examen (exemple de notation : **LXY_CI2_05**). Tous vos fichiers devront être sauvegardés à l'intérieur de ce dossier, qui sera appelé **votre dossier** par la suite.

Business Audit

Partie modélisation (10 points)

- La partie modélisation doit être rendue avant que la partie implémentation puisse être débutée. Cette partie est à réaliser sur papier.
- La partie modélisation est à rendre après 35 minutes au plus tard, mais vous être libre de la rendre plus tôt.

Une entreprise de vente directe d'articles informatiques vous contacte pour développer une application permettant de réaliser un audit[1] de la performance de vente de ses vendeurs. A partir des fichiers journaux (en. : logfile) de l'entreprise, contenant des informations précises sur les ventes de chaque vendeur, l'application doit permettre une comparaison visuelle des différents employés à l'aide d'un diagramme à barres.

Vous trouverez une version exécutable du programme avec des fichiers journaux de l'entreprise dans votre dossier.

Principe de l'application

L'entreprise de vente dispose de deux types de vendeurs : les vendeurs « par téléphone » qui appellent des clients potentiels par ligne téléphonique et les vendeurs « de porte à porte » qui vont sonner à la porte. Le secrétariat de l'entreprise enregistre manuellement toutes les ventes dans des fichiers journaux qui sont organisés de la façon suivante :

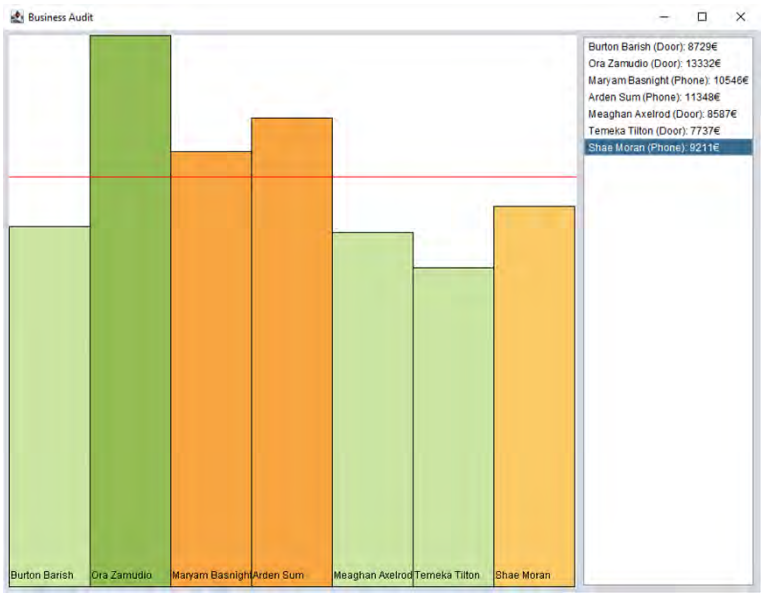
1. Au moment où un nouveau vendeur est employé par l'entreprise, le secrétariat crée manuellement un fichier « .txt » pour lui dans lequel seront placés ses ventes. Chaque vendeur a donc son propre fichier journal.

[1] Audit : activité de contrôle assurée par un auditeur qui délivre à une organisation des conseils pour améliorer ses opérations.

2. Le nom du fichier est composé de deux parties séparées par un point : 1) le type de vendeur qui est soit « phone » soit « door » et 2) le nom du vendeur. Le format du nom d'un tel fichier est donc : *« type ».« nom ».txt*. Exemples : « door.Meaghan Axelrod.txt », « phone.Shae Moran.txt », « door.Temeka Tilton.txt ».
3. Le fichier journal est placé dans le répertoire « employees » à l'intérieur d'une arborescence qui à première vue semble avoir une certaine logique mais qui n'est pas toujours respectée par le secrétariat. On ne peut donc pas être sûr où exactement se trouvent les fichiers journaux à part qu'ils se trouvent quelque part dans le répertoire « employees » ou des sous-répertoires. Il n'y a pas d'autres fichiers à l'intérieur du répertoire « employees » à part des fichiers journaux.
4. Le contenu d'un fichier journal dépend du type de vendeur :
    a. Si le vendeur est un vendeur par téléphone, alors une vente est représentée par une ligne de texte au format suivant : *« date de la vente »;« nom de l'article »;« prix de vente »;« numéro téléphone »*
    b. Si le vendeur est un vendeur porte à porte, alors une vente est représentée par une ligne de texte au format suivant : *« date de la vente »;« adresse »;« nom de l'article »;« prix de vente »*

L'entreprise souhaite une application avec les fonctionnalités suivantes :

1. Au lancement, l'application lit tous les fichiers journaux des vendeurs et affiche le total des ventes de chaque vendeur sous forme d'un diagramme à barres. Les vendeurs par téléphone sont représentés en orange et les vendeurs porte à porte en vert.



2. Le diagramme à barres doit toujours remplir l'intégralité de la largeur disponible du canevas de dessin (à quelques pixels près). La hauteur d'une barre représente le total des ventes d'un vendeur. Elle est proportionnelle au plus grand total obtenu par un vendeur. La plus haute barre occupe donc l'intégralité de la hauteur disponible du canevas de dessin et la hauteur de chaque autre barre est proportionnelle à la hauteur de cette barre.
3. Un trait rouge horizontal démarque la moyenne des totaux obtenus par tous les vendeurs.
4. Si le total d'un vendeur se trouve au-dessus de cette moyenne, la couleur de ce vendeur est plus foncée.
5. Une liste énumère tous les vendeurs de l'entreprise avec leurs totaux respectifs.

6. A l'aide de la souris, il est possible de cliquer dans une zone du canevas pour sélectionner le vendeur dans la liste auquel appartient cette barre. La zone d'un vendeur correspond à la barre de ce vendeur mais occupe toujours l'intégralité de la hauteur du canevas.

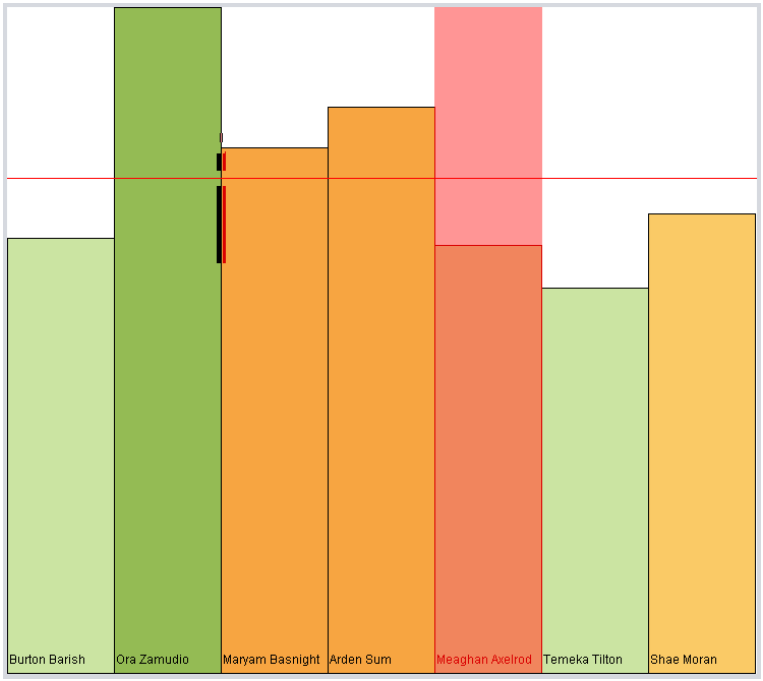


Figure 1: Mise en évidence de la zone de la vendeuse "Maeghan Axelrod"

Travail à réaliser

Réaliser un modèle de classe du **modèle** de l'application sur papier en respectant les conventions UML usuelles.

LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2020

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI, CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :* *08/06/2020*

Partie implémentation (50 points)

Créez dans votre dossier un nouveau projet **BusinessAudit**. Développez l'application en vous basant sur la version exécutable, le diagramme UML et les instructions données par la suite.

Classe « Employee » (6 points)

La classe abstraite **Employee** représente un employé/vendeur de l'entreprise.

Attributs, constructeur et accesseur : (1 point)

- **name** représente le nom du vendeur.
- **totalSales** représente le total des ventes obtenu par ce vendeur.
- **aboveAverage** définit si le total des ventes du vendeur est supérieure à la moyenne des totaux des ventes.
- Le **constructeur** initialise la valeur de l'attribut **name** et fait appel à la méthode **readSalesFromFile()** pour calculer le total des ventes de ce vendeur.
- **getTotalSales()** est un accesseur de l'attribut **totalSales.**

La méthode abstraite **readSalesFromFile()** prend en paramètre le nom de fichier du fichier journal du vendeur et calcule le total de ses ventes. La méthode doit lancer une exception du type *IOException* si un problème lors de la manipulation d'un fichier se produit. Comme le format des lignes des fichiers journaux varie selon la fonction du vendeur, cette méthode est implémentée dans les classes filles. (0.5 points)

La méthode **setAboveAverage()** prend en paramètre la moyenne des totaux des ventes des vendeurs et modifie la valeur de **aboveAverage**. (0.5 points)

La méthode **draw()** prend en paramètre la distance entre le côté gauche de la barre à dessiner et le bord gauche du canevas de dessin, la largeur de la barre, la hauteur du canevas de dessin et le plus grand total des ventes des vendeurs. Ensuite elle dessine la barre du vendeur avec un bord noir au bon endroit et avec les bonnes dimensions (cf. modélisation). Finalement elle dessine le nom du vendeur dans le coin inférieur gauche de la barre (vous n'avez pas besoin de traiter le cas où le nom est plus long que la largeur de la barre). (4 points)

Classe « PhoneSalesman » (5.5 points)

La classe **PhoneSalesman** est une classe fille de la classe **Employee** et représente un vendeur par téléphone.

Définissez l'héritage et le constructeur de la classe. (1 point)

Implémentez la méthode **readSalesFromFile()** définie dans la classe mère. Le format d'une vente dans le fichier journal d'un vendeur par téléphone est : *« date de la vente »;« nom de l'article »;« prix de vente »;« numéro téléphone »*. (2 points)

La méthode **draw()** redéfinit la méthode de la classe mère. Elle définit la couleur de dessin du vendeur par téléphone en fonction de la valeur de l'attribut **aboveAverage** et profite ensuite de l'héritage pour dessiner la barre du vendeur. La couleur foncée, à utiliser si son total est supérieur à la moyenne (cf. captures d'écran), est RGB(247, 165, 65) et la couleur non-foncée est RGB(250, 202, 102). (2 points)

La méthode **toString()** retourne la représentation textuelle d'un vendeur par téléphone. Déterminez le format du texte à partir de l'exécutable fourni. (0.5 points)

Classe « DoorSalesman » (3.5 points)

La classe **DoorSalesman** est une classe fille de la classe **Employee** et représente un vendeur porte à porte.

Définissez l'héritage et le constructeur de la classe. (1 point)

Implémentez la méthode **readSalesFromFile()** définie dans la classe mère. Le format d'une vente dans le fichier journal d'un vendeur porte à porte est : *« date de la vente »;« adresse »;« nom de l'article »;« prix de vente »*. (1 point)

La méthode **draw()** redéfinit la méthode de la classe mère. Elle définit la couleur de dessin du vendeur porte à porte en fonction de la valeur de l'attribut **aboveAverage** et profite ensuite de l'héritage pour dessiner la barre du vendeur. La couleur foncée, à utiliser si son total est supérieur à la moyenne (cf. captures d'écran), est RGB(148, 187, 84) et la couleur non-foncée est RGB(203, 228, 162). (1 point)

La méthode **toString()** retourne la représentation textuelle d'un vendeur porte à porte. Déterminez le format du texte à partir de l'exécutable fourni. (0.5 points)

Classe « Node » (3 points)

La classe **Node** représente un nœud d'une liste chaînée permettant de stocker un employé/vendeur. Implémentez la classe **Node** selon le schéma UML. La classe comprend un constructeur, des accesseurs et manipulateurs ainsi que la méthode **hasNext()** qui retourne **true** si le nœud possède un successeur et **false** s'il est le dernier nœud de la liste.

Classe « Business » (26 points)

La classe **Business** représente l'entreprise de vente. Cette classe sauvegarde les employés de l'entreprise dans une liste chainée implémentée manuellement. L'attribut **root** représente le premier nœud (la tête) de cette liste, l'attribut **size** représente la taille de la liste et la méthode **getSize()** est un accesseur de l'attribut **size**. (1 point)

La méthode **addEmployee()** prend en paramètre un employé, l'ajoute à la fin de la liste chaînée et incrémente la taille de la liste. (3 points)

La méthode **getEmployee()** prend en paramètre l'index d'un nœud et retourne l'employé se trouvant à cette position dans la liste. Si l'index fourni n'existe pas dans la liste, la méthode lance une exception du type *IndexOutOfBoundsException*. (3 points)

La méthode **toArray()** retourne un tableau statique contenant les éléments de la liste chaînée. Cette méthode permet d'afficher les vendeurs dans une liste graphique. (3 points)

La méthode **readEmployeesFromFolder()** prend en paramètre le nom d'un répertoire. On suppose qu'à l'intérieur de ce répertoire se trouvent soit des sous-répertoires soit des fichiers journaux de vendeurs (cf. modélisation). Cette méthode remplit la liste chaînée avec tous les vendeurs contenus dans ce répertoire ou profite de la récursivité pour lire le contenu des sous-répertoires. Astuce : l'expression pour faire un split() d'un texte avec un point est "\\." (5 points)

Indication :
Utilisez les méthodes de la classe **java.io.File**, notamment: **isDirectory**, **isFile**, **listFiles**. Consultez *JavaDoc* à ce sujet.

La méthode **getMaximumSales()** détermine et retourne le maximum des totaux des ventes des vendeurs de la liste chaînée. La méthode ne doit pas considérer le cas où la liste serait vide. (2 points)

La méthode **getAverageSales()** détermine et retourne la moyenne des totaux des ventes des vendeurs de la liste chaînée. La méthode ne doit pas considérer le cas où la liste serait vide. (2 points)

La méthode **setAboveAverageEmployees()** utilise la méthode **setAboveAverage()** de la classe **Employee** pour définir les employés performant mieux que la moyenne. (2 points)

La méthode **draw()** prend en paramètre les dimensions du canevas de dessin. Après avoir calculé la largeur d'une barre, elle fait appel à la méthode **draw()** de la classe **Employee** pour dessiner les barres du diagramme. Finalement elle dessine le trait horizontal de la moyenne. (5 points)

Classe DrawPanel (1 point)

La classe **DrawPanel** (*JPanel*) est responsable de la visualisation du diagramme. Rajoutez l'attribut **business** et le manipulateur **setBusiness()**. Cette classe contient aussi la méthode **paintComponent()** qui affiche d'abord un fond blanc et ensuite le diagramme à barres.

Classe MainFrame (5 points)

Implémentez la classe **MainFrame** et son interface en vous basant sur le schéma UML et l'exécutable fourni. L'attribut **business** représente le modèle de l'application. (1 point)

Au lancement de l'application, celle-ci lit tous les vendeurs du répertoire « employees » et détermine les employés performant mieux que la moyenne. Si une erreur se produit lors de la lecture d'un fichier, le message de l'erreur doit être affiché à la console. (1.5 points)

La méthode **updateView()** actualise les valeurs affichées. (0.5 points)

En cliquant avec un bouton de la souris sur le canevas de dessin, le vendeur dans la zone duquel le joueur a cliqué, est sélectionné dans la liste graphique des vendeurs. (2 points)

Schéma UML

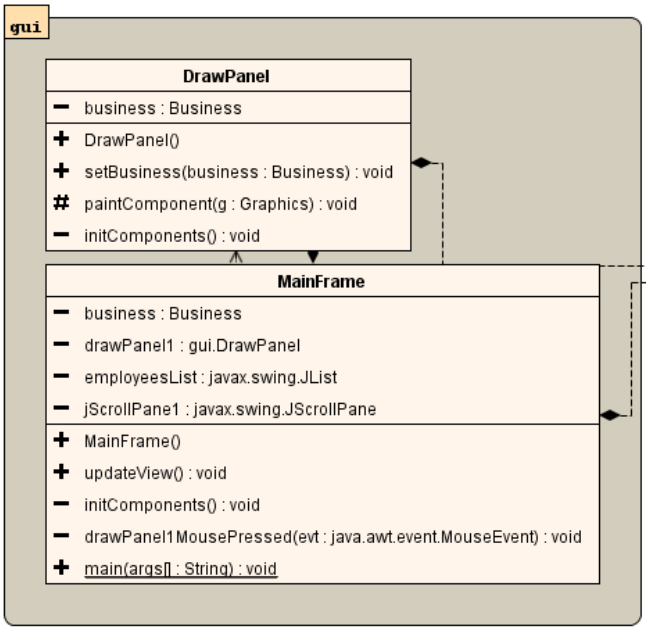


Figure 2: Schéma UML du paquet gui.

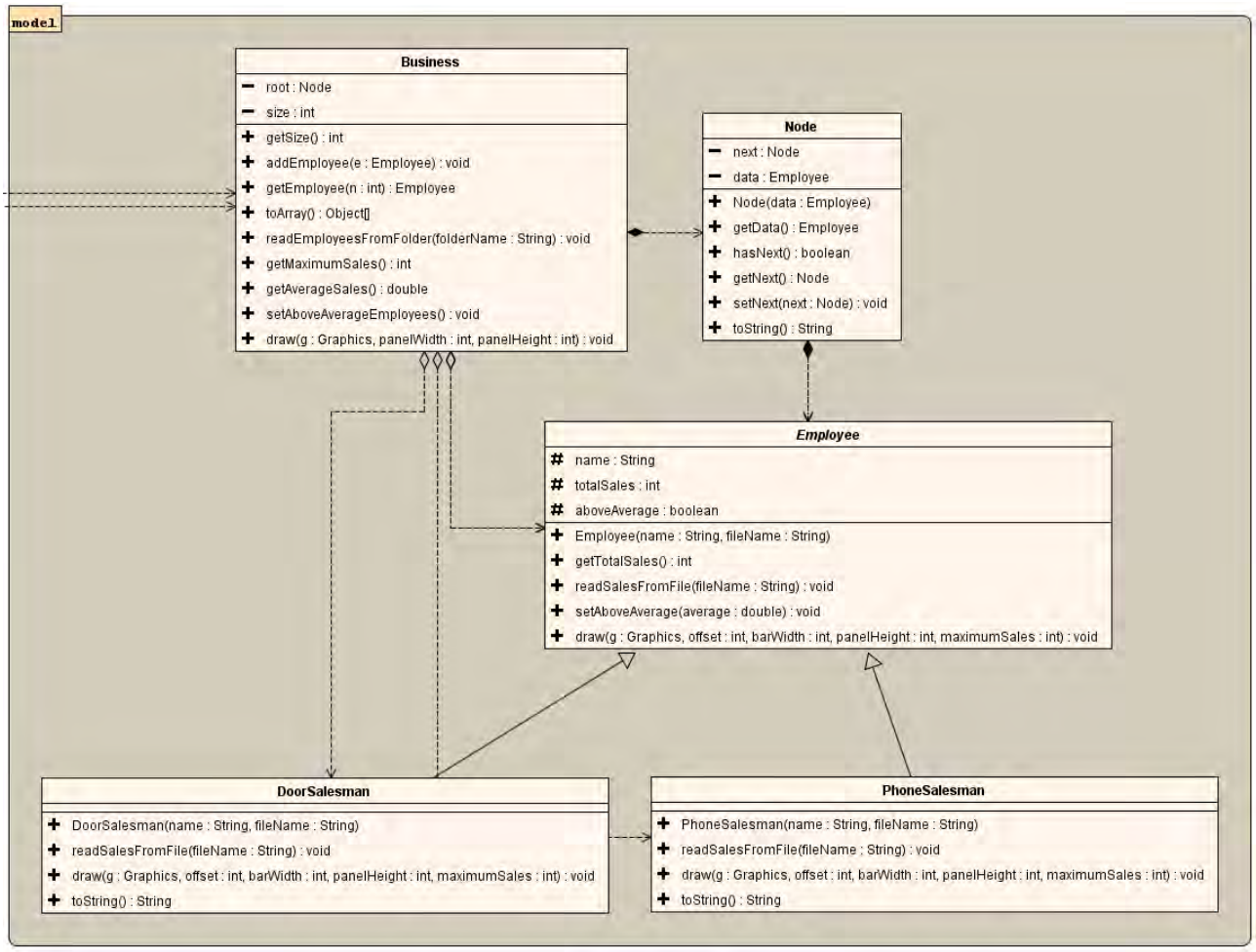


Figure 3: Schéma UML du paquet model.