TP 1

Enseignant: Abdoul Majid Thiombiano/@thiomajid

Description du projet

Notre objectif est de fournir une application de gestion de librairie. Au cours de ce TP, un premier prototype de notre application *Libzy*.

Etape 1: "Bienvenue chez Libzy"

A démarrage, l'application doit afficher le message "Bienvenue chez Lizbzy".
Créer un fichier nommé `Libzy.java` dans lequel vous allez écrire votre programme.

> **Info**
>
> Il est conseillé de sauvegarder vos programmes Java dans un répertoire. Vous pouvez par exemple le nommer `tp1`.

```java
public class Libzy {
    public static void main(String[] args){
        System.out.println("Bienvenue chez Libzy");
    }
}
```

Etape 2: Choix d'utilisateur

Notre application aura deux types d'utilisateur: un **gérant** et un **abonné**.
Après le message de bienvenue, afficher le menu suivant pour déterminer le type d'utilisateur:

- 0: Gérant
- 1: Abonné

Si l'utilisateur est un gérant, alors le menu suivant lui est affiché:

- 0: Ajouter un ouvrage
- 1: Mettre à jour un ouvrage
- 2: Lister vos ouvrages
- 3: Supprimer un ouvrage

Par contre, si l'utilisateur tape 1, afficher lui un message de salutation.
Sinon un message d'erreur sera affiché indiquant "Cette option n'existe pas".
Bonus: Comment faire pour que l'utilisateur puisse réessayer jusqu'à ce qu'il saisisse une valeur correcte ?

```
// Pour lire un entier à partir du clavier, on procède comme suit
import java.util.Scanner;

Scanner scanner = new Scanner(System.in);
int value = scanner.nextInt();

// NB: N'oublier pas de fermer l'objet Scanner pour libérer les ressources.
```

Etape 3: Notre offre

Libzy propose un riche catalogue d'ouvrage à vendre. Le tableau suivant présente certains de nos ouvrages, avec chaque ouvrage caractérisé par un ID, une référence, un nom, son auteur et un prix (TND).

ID	Référence	Nom	Auteur	Prix
0	Leirne0	Le Petit Prince	Antoine de Saint Exupéry	12
1	Mdlat1	Middle Earth	J. R. R. Tolkien	25
2	HryPtetlrsnirdAkbn2	Harry Potter et le prisonnier d'Azkaban	J. K. Rowling	15

Pour chaque colonne du tableau, créer et initialiser une liste de ces valeurs en utilisant la class `ArrayList`.

```
import java.util.ArrayList;

// pour créer une liste
ArrayList<type> myList = new ArrayList<type>();

// Les méthodes suivantes sont prédéfinies pour la class ArrayList
myList.add(item) // ajoute un élément à la liste
myList.remove(index) // supprime l'élément se trouvant à la position indiquée
myList.size() // retourne le nombre d'éléments contenu dans la liste
myList.get(index) // retourne l'élément à la position indiquée
myList.set(index, item) // remplace l'élément à la position indiquée avec une nou
```

Etape 4: Enregistrement d'un nouvel ouvrage

Nous avons reçu un nouvel ouvrage et voulons l'enregistrer dans notre registre, Il s'agit du livre "Harry Potter et la coupe de feu" de J. K. Rowling coûtant 20DT.

1. Dans un premier temps, on va écrire une fonction `createReference` qui à partir du nom et de l'identifiant d'un ouvrage retourner la référence de ce dernier. La référence d'un ouvrage est une chaîne de caractères composée des caractères de son nom d'indice pair, à laquelle on ajoute l'ID de l'ouvrage. Si le caractère est un *espace*, on l'ignore.
2. On écrira ensuite une fonction `registerItem` qui prend en entrée le *nom*, *l'auteur* et le *prix* d'un ouvrage pour l'ajouter à la liste de nos ouvrages.

NB: L'ID pourrait être une variable incrémenter à l'ajout d'un nouvel ouvrage.

```
// Pour déclarer une méthode, on suit la syntaxe suivante
type_de_retour nom_methode(type1 arg1, type2 arg2, ...){
    // instructions
}
```

Etape 5: Oh, la belle erreur !

Lors de l'enregistrement de notre nouvel ouvrage, on a fait une erreur au niveau du prix. Au lieu de 20DT, l'ouvrage devrait coûter 18DT.
Pour corriger notre problème, on écrira une méthode `updateItem` qui à partir de la *référence* d'un ouvrage et du *nouveau prix* saisi par l'utilisateur le met à jour.

Etape 6: Qu'avons-nous en stock ?

Maintenant que nous avons nos ouvrages, on aimerait bien les consulter. Pour faciliter la lecture, on voudrait les afficher de deux manières selon les prix: **croissant** et **décroissant**.

- Ecrire une fonction `displayByPriceAscending` qui affiche nos ouvrages suivant les valeurs croissantes du prix.
- Ecrire une fonction `displayByPriceDescending` qui affiche nos ouvrages suivant les valeurs décroissantes du prix.

> **Note**
>
> L'affichage doit se faire de la façon suivante:
> <ID> <Référence> <Nom> <Auteur> <Prix>

Etape 7: On a un intrus.

Après avoir consulté nos ouvrages, on se rend compte qu'il y a un ouvrage dans la liste que nous n'avons plus en stock. On aimerait bien le retirer de notre stock. Pour ça on écrira une fonction `deleteItem` qui à partir de la référence d'un ouvrage le supprimera de notre stock.

Etape 8: Sum up!

A partir du menu que nous avons établi à l'étape 2, exécuter le programme et tester les différents cas de figures évoqués dans les précédentes étapes.

“Simplicity is the highest sophistication.”, Leonardo Da Vinci.

e