Secteur Tertiaire Informatique

Filière « Etude et développement »

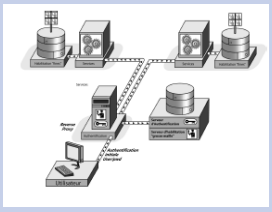
Entreprise EasyCar – seconde partie

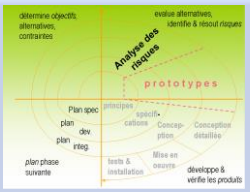
Programmation Orientée Objet

Apprentissage

Mise en situation

Evaluation





1. INTRODUCTION

Il vous est ici proposé de reprendre votre projet « EasyCar » afin de l'améliorer en implémentant, sur certaines classes, des interfaces communes du JDK.

Vous allez implémenter à votre logiciel les deux fonctionnalités suivantes :

- Le **tri d'une liste de clients par ordre croissant d'argent total dépensé**
- La sauvegarde binaire (aussi appelée « Sérialisation binaire ») d'information de client dans un fichier.

Pour vous permettre d'implémenter ces fonctionnalités, vous allez voir comment utiliser les interfaces suivantes :

- L'interface « Comparable » : utilisée pour la comparaison d'objets
- L'interface « Serializable » : permet d'implémenter la sérialisation

2. IMPLEMENTATION

2.1 TRI DES CLIENTS

Pour trier les clients d'une « ArrayList » dans l'ordre **croissant d'argent dépensé** vous pouvez utiliser la méthode statique « **sort** » de la classe « **Collections** » (disponible dans le package « java.util »).

Ci-dessous un extrait de la Javadoc définissant cette méthode (https://devdocs.io/openjdk~18/java.base/java/util/collections#sort(java.util.List)) :

sort

```
public static <T extends Comparable<? super T>> void sort(List<T> list)
```

Sorts the specified list into ascending order, according to the natural ordering of its elements. All elements in the list must implement the `Comparable` interface. Furthermore, all elements in the list must be *mutually comparable* (that is, `e1.compareTo(e2)` must not throw a `ClassCastException` for any elements `e1` and `e2` in the list).

This sort is guaranteed to be *stable*: equal elements will not be reordered as a result of the sort.

The specified list must be modifiable, but need not be resizable.

Vous pouvez utiliser le tutoriel suivant pour apprendre à utiliser cette méthode : https://codegym.cc/fr/groups/posts/fr.222.tri-des-collections-java-

A faire

Pour la classe « Customer », implémentez l'interface « Comparable » et comparez les objets en fonction de l'argent total dépensé.

Triez un objet de la classe « ArrayList » contenant plus de 3 clients avec la méthode « **sort** » de la classe « **Collection** ».

2.2 SERIALISATION BINAIRE

Vous allez sauvegarder dans un fichier attributs d'objets de la classe « Customer ».

Attributs à sauvegarder :

- Prénom
- Nom
- Adresse
- Ville
- Code postal

Attention

La liste des réservations d'un client n'est pas à prendre en compte, dans un premier temps.

Pour apprendre à mettre en place la sérialisation binaire, consultez l'article suivant : https://ydisanto.developpez.com/tutoriels/java/serialisation-binaire/

A faire

Implémentez l'interface « Serializable » pour la classe « Customer ».

Essayez tout d'abord de sérialiser des objets de la classe « Customer ».

Essayez ensuite de désérialiser les objets à partir d'un fichier (il vous faudra probablement mettre en place un système de saisie utilisateur afin de demander s'il souhaite sérialiser ou désérialiser les objets).

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 04/07/2024