Programmation Objet Avancée - TD3

—o000o——o000o—

Héritage et interface

1 Téléphones et smartphones

Créez 2 classes `FeaturePhone` et `SmartPhone` ayant pour attributs communs :

- Une marque
- Un modèle
- Un numéro de téléphone
- Un opérateur téléphonique
- Un attribut spécifiant si l'appareil est bloqué chez cet opérateur

La classe `SmartPhone` possède en plus :

- Un attribut spécifiant le système d'exploitation (Android, iOS)
- Un attribut spécifiant si le smartphone possède un appareil photo
- Un attribut spécifiant si le smartphone possède un GPS

Par défaut, tous les téléphones sont bloqués chez un opérateur téléphonique.

1. Créez les classes nécessaires (attributs, constructeur, ...)
2. Créez également une méthode permettant d'afficher l'ensemble des valeurs d'un téléphone, une méthode permettant de débloquer un téléphone ainsi qu'une méthode permettant de changer le numéro de téléphone d'un téléphone
3. Créez une méthode `equals(Object o)` qui retourne `true` si et seulement si les 2 téléphones sont identiques (même modèle et de même marque)
4. Dans une classe de test, créez 2 `FeaturePhone` et 2 `SmartPhone`
5. Ajoutez ces 4 téléphones nouvellement créés dans la structure de données de votre choix
6. Appliquer la méthode de déblocage à l'ensemble des téléphones
7. Appliquer la méthode d'affichage à l'ensemble des téléphones
8. ⋆ ⋆ ⋆ **Pour aller plus loin**, utilisez la méthode `forEach` présent dans Java 8 pour répondre aux questions 6 et 7
9. Créez une interface `Call` contenant une unique méthode `call(Phone phone)` permettant d'afficher les 2 numéros en communication

2 Pages et enveloppe

Nous souhaitons modéliser un courrier. Pour ceci, nous allons définir les classes `Page` et `Enveloppe`.

Une page est composée :

- D'un recto (une chaîne de caractères)
- Et d'un verso (une autre chaîne de caractères)

Un seul côté est visible à la fois.

1. Écrire la classe `Page` contenant les méthodes suivantes :
   - Un constructeur prenant en paramètre 2 chaînes de caractères représentant le recto et le verso d'une feuille. Initialement, seul le recto de la feuille est visible
   - La redéfinition de la méthode `String toString()`. Pour rappel, un seul côté est visible
   - Une méthode `void flip()` permettant de retourner la page (rend visible le côté qui ne l'était pas)
2. Testez la classe `Page`
3. Écrire la classe `Envelope`. Elle aura pour propriétés :
   - Un tableau de `Page`
   - Le nombre maximal de pages qu'une enveloppe peut contenir
   - L'état de l'enveloppe (ouverte ou fermée)

   Et pour méthodes :
   - `void open()` et `void close()` permettant respectivement d'ouvrir ou de fermer l'enveloppe
   - `boolean addPage(Page page)` permettant d'ajouter une page à une enveloppe. N'oubliez pas de contrôler le nombre maximal de pages autorisées
   - ⋆ ⋆ ⋆ **Pour aller plus loin**, implémentez votre propre `ArrayList` et redéfinissez la méthode `boolean add(Page page)`
   - `void read()` permettant de lire l'ensemble des pages contenues dans l'enveloppe (recto et verso). La lecture n'est possible que si l'enveloppe est ouverte
   - `void size()` permettant de retourner le nombre de pages contenues dans une enveloppe
4. Testez la classe `Envelope`
5. ⋆⋆ Le papier étant consommateur d'arbres, nous essayons de limiter au maximum son utilisation. Modifiez la classe `Page` en y ajoutant, notamment, la méthode `void greenThreshold(int threshold)`, qui fixe le nombre maximum de pages au-delà duquel, le message "**Halte au gaspillage !**" sera affiché avant chaque création d'une nouvelle page.