Programmation orientée objet

Série 3

Joel Cavat / 2020

Exercices pratiques

3.1 Exercice *(Transvasement)*

Vous devez développer une fonctionnalité permettant de faire un transevasement d'un récipient d'origine à d'autres récipients de destination.

3.1.1 *(version "fonction")*

Cette version doit être réalisée à l'aide d'une méthode et un nombre arbitraire d'arguments

- le premier argument est le récipient d'origine
- le deuxième argument est le premier récipient de destination
- (ces deux premiers arguments sont obligatoires)
- les arguments suivants sont les récipients secondaires

Voici un exemple d'utilisation:

- `transfer(10, 3, 3, 3) -> {3,3,3}` (les trois récipients de destination sont remplis à ras bord)
- `transfer(10, 3, 3, 3, 3) -> {3,3,3,1}`
- `transfer(5, 3) -> {3}`
- `transfer(0, 3) -> {0}`
- `transfer(100, 10, 40, 30, 50) -> {10, 40, 30, 20}`

Utilisez un nombre arbitraire d'arguments tout en obligeant l'utilisateur à renseigner au moins deux arguments

3.1.2 *(version objet)*

Réalisez un système de classes permettant une utilisation suivante:

```
Container origin = Container.withCapacity(10); // récipient vide par défaut
origin.fillToTheMax(); // remplissage
Container destination1 = new Container(5); // idem que Container.withCapacity(5);
destination1.fillWith(2);
Container destination2 = Container.withCapacity(3);
Container destination3 = Container.withCapacity(10);
origin.fillTo(destination1, destination2, destination3);
```

```
assert destination1.isFull() ;
assert destination2.isFull() ;
assert destination2.remaining() == 0 ;
destination2.remove(2);
assert destination2.remaining() == 2 ;
assert !destination3.isFull() ;
assert destination3.quantity() == 4;
destination3.flush();
assert destination3.quantity() == 0;
```

Remarques:

- Utilisez un nombre minimal de champs
- Exposez que les fonctionnalités utiles à l'utilisateurs, les autres doivent rester privée
- La méthode `fillTo` doit accepter au minimum un récipient
- Gérez les états incohérents du mieux possible

3.2 Exercice *(Account - version 0.1)*

Réalisez des fonctionnalités sur des comptes bancaires. Nommez votre classe `Account`.

Liste des fonctionnalités:

- création d'un compte vide pour un propriétaire donné (à l'aide d'un constructeur);
- création d'un compte pour un propriétaire donné en spécifiant l'argent qu'il a initiallement. Un compte ne peut pas être créé avec un montant négatif (à l'aide d'un constructeur);
- retirer de l'argent d'un compte;
- déposer de l'argent sur un compte;
- virer de l'argent d'un compte à un autre;
- pour toutes les opérations, le solde d'un compte ne doit jamais être inférieur à 2000.-;
- récupérer le nom complet du propriétaire à l'aide d'une chaîne de caractères;
- comparer si deux comptes sont égaux. Ils le sont s'ils ont le même propriétaire et le même montant;