Programmation orientée objet

Série 6

Joel Cavat / 2020

Exercices

6.1 Exercice *(Account)*

Reprenez votre exercice sur les comptes et rendez-les complétement immutable. Réalisez l'implémentation minimal qui permet de compiler et d'exécuter le code ci-dessous:

```
Account stephanie = Account.withName("Stéphanie").withAmount(1000);
Account julia = Account.withName("Julia"); // initialisé à 0.-
Account stephanie2 = stephanie.deposite(1000);
```

Une opération d'un compte à un autre compte génère une transaction. Une transaction a pu se dérouler correctement ou non:

- si tout s'est bien passé, la transaction référence les nouvelles versions des deux comptes après la transaction
- s'il y a eu un problème, un message d'erreur peut être récupéré

Une telle utilisation doit se réaliser ainsi:

```
Transaction t = a2.transferTo(a1); // doesn't compile
a2.transferTo(a1); // do nothing interesting
Transaction t = a2.transferTo(a1).amount(350);
a1.transfer(a2).amount(350).ifSuccessOrElse(
  (originAccount, destinationAccount, amount) -> {
    System.out.println("All good");
    System.out.println("now a1 has " + originAccount.amount());
    System.out.println("and a2 has " + destinationAccount.amount());
  },
  (String errorMsg) -> System.out.println("operation dennied: " + errorMsg)
);
```

Pour cet exercice, employez des types imbriqués et des interfaces fonctionnelles.

6.2 Exercice *(Statut)*

Reprenez l'exercice de la série précédente et modifiez-le pour intégrer `On`, `Off` et `Err` en tant que classe interne de l'interface `Status`.

6.3 Exercice *(`ListInt` - `ArrayListInt`)*

Reprenez l'exercice de la série précédente sur les listes d'entiers et réalisez votre propre itérateur sur celle-ci.

```java
ListInt list = new ArrayListInt();
list.insert(0);
list.insert(3);
list.insertAll(2,1);

for (int i = 0; i < list.size(); i+=1) {
    int v = list.get(i);
    System.out.println("Value: " + v);
}

/* l'itérateur permet maintenant le code ci-dessous */
for (int v: list) {
    System.out.println("Value: " + v);
}

/* ou */
Iterator<Integer> it = list.iterator();
while (it.hasNext()) {
    System.out.println("Value: " + it.next());
}

/* et même */
list.forEach( v -> System.out.println("Value: " + v));
```

6.4 Exercice

Completez le code pour que les appels ci-dessous fonctionnent :

```java
interface Pushable {
  void push();
}
```

```java
public class Test {
  public static void push(Pushable p) {
    p.push();
    System.out.println("Button has been pushed");
  }

  public static void main(String[] args) {
    /* completez le code ci-dessous */
    push(




    ); /* Doit afficher:
      Push
      Button has been pushed
    */
  }
}
```