

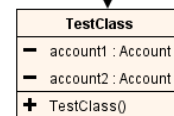
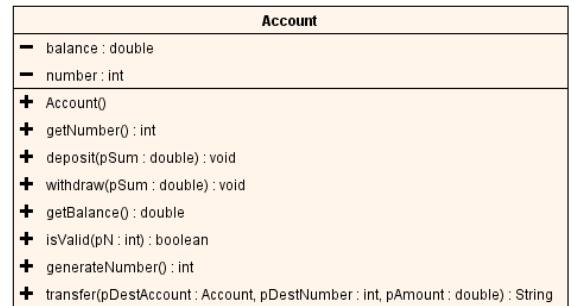
Exercice : bank accounts with several objects

La banque *RabloBank* a codé ses numéros de compte de façon à ce qu'il soit possible de contrôler si un numéro de compte a été entré correctement. Pour ce faire, les deux derniers chiffres du numéro correspondent à la somme des chiffres précédents (DE : *Quersumme*).

Les numéros de compte (chiffres de contrôle inclus) sont toujours des nombres à 9 chiffres.

Exemples :

```
123450015  est  donné  correctement,  car
1+2+3+4+5 = 15
110021106  est  donné  correctement,  car
1+1+2+1+1 = 06
643553947 n'est pas donné correctement, car
6+4+3+5+5+3+9 = 35 ≠ 47
```



1 Classe Account

Développez la classe **Account** (FR : compte; DE : Bankkonto) qui possède les propriétés et fonctionnalités suivantes :

1. Chaque compte possède un numéro de compte **number**, qui peut être consulté avec **getNumber()**. Le compte a un solde (nombre réel). On suppose que les valeurs entrées pour **pSum** sont positives. **balance** peut prendre des valeurs négatives.
2. Au début le compte est vide (le solde est zéro).
3. On peut ajouter un certain montant à un compte.
4. On peut retirer un certain montant d'un compte.
5. On peut demander le solde actuel (accesseur).
6. La méthode **isValid(long pN)** contrôle si un numéro de compte donné comme paramètre est valide (selon la description ci-dessus).
7. La méthode **generateNumber()** sert à générer un numéro de compte valide de façon aléatoire. (On néglige ici le fait qu'en réalité il est impossible d'employer une telle méthode parce qu'elle risque de produire deux fois le même numéro de compte.)
8. Lors de la création d'un compte, il faut vérifier que le numéro de compte donné comme paramètre est valide, sinon un numéro de compte valide doit être créé de façon aléatoire.
9. La méthode **transfer(...)** permet de transférer une somme d'argent du compte actuel vers un autre compte. Pour effectuer le transfert, il faut indiquer comme paramètre le compte (objet) destinataire, le numéro de compte destinataire (comme contrôle) et le montant à transférer. Avant d'effectuer le transfert, il faut vérifier,
 - a. * si le numéro de compte destinataire est valide,
 - b. * si le nom du compte destinataire correspond au numéro de compte donné,

c. * s'il reste assez d'argent pour effectuer le transfert.

Comme résultat, la méthode `transfer(...)` retourne un texte indiquant le succès de l'opération ou une description de l'erreur commise, p.ex. :

```
"ERROR: The remitte's account number is not valid!"
```

```
"SUCCESS: The transfer has been accomplished successfully!"
```

...

10. La méthode `toString()` retourne un texte contenant le numéro de compte et le solde actuel.

Classe TestClass

La classe `TestClass` permet de tester les différentes méthodes de la classe `Account`, particulièrement pour vérifier le bon fonctionnement du transfert d'argent entre deux comptes bancaires.

Etapes à réaliser :

1. Ajoutez les deux attributs comme montré sur le diagramme UML.
2. Le constructeur crée 2 instances de la classe `Account`. Les références sont stockées dans `account1` respectivement dans `account2`.
3. Dans le constructeur, implémentez les opérations suivantes :
 - a. Ajoutez 1.000.000.00 € sur le compte `account1`.
 - b. Transférez 1000.00€ d'`account1` vers `account2` en spécifiant 123456 comme compte destinataire. Affichez le résultat de cette opération à la console (une erreur s'est produite puisque le compte 123456 n'est pas un numéro de compte valable).
 - c. Lisez le numéro de compte bancaire `account2` et transférez à nouveau 1000.00€ d'`account1` vers `account2` en spécifiant le numéro de compte bancaire lu auparavant. Affichez le message de cette opération.
 - d. Affichez le solde des deux comptes de façon claire à la console.
 - e. Lisez le numéro de compte bancaire `account1` et transférez 49.60€ d'`account2` vers `account1` en spécifiant le numéro de compte bancaire lu auparavant. Affichez le résultat de cette opération à la console.
 - f. Affichez le solde des deux comptes de façon claire à la console.

Voici les résultats des différentes opérations affichés à la console :

```
ERROR: The remitte's account number is not valid!
```

```
SUCCESS: Amount correctly transferred
```

```
Solde compte 1: 999000.0
```

```
Solde compte 2: 1000.0
```

```
SUCCESS: Amount correctly transferred
```

```
Solde compte 1: 999049.6
```

```
Solde compte 2: 950.4
```