1 BankAccount

Dans cette exercice, on va travailler sur la classe `BankAccount` ci-dessous qui permet de gérer un compte en banque avec retrait et dépôt.

```
public class BankAccount {
  private double balance;

  public BankAccount(double initialBalance)
  {
    this.balance = initialBalance;
  }

  public void deposit(double depositAmount)
  {
    this.balance += depositAmount;
  }

  public void withdraw(double withdrawAmount)
  {
    this.balance -= withdrawAmount;
  }

  public double getBalance()
  {
    return this.balance;
  }

}
```

Question 1 : À quoi correspond l'attribut `balance` ? Pourquoi a-t-on utilisé le mot clé `private` afin de le rendre inaccessible à l'extérieur de la classe ?

Question 2 : Quel est le rôle du mot-clé `this`. Est-ce que son utilisation est indispensable ?

Question 3 : Quel est le rôle de chaque méthode ?

Un programmeur a écrit le code suivant qui utilise la classe `BankAccount` :

```java
public class MyBank{
  public static void main(String[] args){

    BankAccount account = new BankAccount(700);
    account.withdraw(-500);

    System.out.println("BankAccount has a balance of " + account.getBalance());
  }
}
```

Question 4 : Est-ce que cela vous paraît être une utilisation normale de la classe `BankAccount` ?

Question 5 : On souhaiterait rajouter une méthode `String toString()` à la classe afin de convertir le compte en une chaîne de caractère au format suivant : `"Compte crédit : 123$"` (exemple d'un compte ayant le numéro 12 et un crédit de 123). Que faut-t-il changer à la classe `BankAccount` ?

Question 6 : On souhaiterait empêcher les retraits (`withdraw`) entraînant un crédit négatif. Un retrait entraînant un solde négatif ne serait tout simplement pas exécuté. Comment peut-on changer la classe `BankAccount` pour réaliser cela ? Faut-il changer la signature de la méthode `withdraw` ?

Question 7 : On souhaiterait rajouter un attribut à la classe permettant de lier une personne à un compte en banque. Comment peut-on changer la classe `BankAccount` pour réaliser cela ?

2 Flacons (bonus)

Vous travaillez chez Bobard & Co., fabriquant exclusif du sirop *Mirifik*, un élixir issu de la recherche spatiale qui rend jeune, beau et intelligent et qui empêche la chute des cheveux et les tâches de transpiration. Dilué à différentes concentrations, le sirop Mirifik est commercialisé dans diverses sortes de flacons.

Nous nous intéressons à la classe `Bottle` dont les instances représentent les flacons de sirop Mirifik en stock. Cette classe comporte :

des attributs :

— une capacité en mL,
— un volume de liquide en mL,
— une concentration (le rapport volume de sirop / volume de sirop et d'eau),
— un étiquette (un texte quelconque),

des méthodes :

— `public void pourWater(double volume)` permettant d'ajouter de l'eau,
— `public void pourSyrup(double volume)` permettant d'ajouter du sirop,
— `public void transvase(Bottle pouringBottle, double volume)` permettant de verser du contenu provenant d'une autre bouteille.

— `public void drink(double volume)` permettant d'extraire une quantité de liquide égale à `volume` (toute la bouteille si `volume` est supérieure ou égale à la quantité de liquide présent dans la bouteille).
— `public String toString()` produisant une chaîne de caractères indiquant l'étiquette, le volume et la concentration de la bouteille.

Si une bouteille est pleine, tout ce qu'on tente d'y ajoute s'écoule à côté de la bouteille et est perdu.

Question 8 : Quels qualificateurs, `public`, `final` ou `private`, doit-on donner aux attributs ?

Question 9 : Est-il raisonnable d'ajouter des méthodes pour accéder aux attributs privées ?

Question 10 : Implémenter la classe `Bottle`.