Exercice supplémentaire R3

Banking
- amount : double - interestRate : double
+ Banking(pAmount : double, pInterestRate : double) + getAmount() : double + setAmount(pAmount : double) : void + getInterestRate() : double + setInterestRate(pInterestRate : double) : void + computeYearlyInterest(pAmount : double, pInterestRate : double) : double + computeMonthlyInterest(pAmount : double, pInterestRate : double) : double + printCumulativeWins(years : int) : void + printLoanPayback(payback : double) : void + round(val : double, n : int) : double

Développez la classe Banking décrite par le diagramme UML suivant :
Cette classe permet de calculer et d'afficher des montants pour de l'épargne ou du crédit.

1. Programmez cette classe avec les attributs, accesseurs, modificateurs et constructeur indiqués.

   amount : le montant (*Geldsumme*) – en € – stocké sur le compte (*Konto*)

   interestRate : le taux d'intérêt **annuel** (par an) du crédit (*Kredit*) ou de l'épargne (*Ersparnis*), par exemple la valeur 2.5 signifie 2,5 %

2. Ajoutez computeYearlyInterest() dont le but est de calculer et de retourner la valeur des intérêts pour le montant et taux d'intérêts passés en paramètre pour **1 année**

   Exemples :
   ```
   computeYearlyInterest(1000, 2.5) → 25.0
   computeYearlyInterest(1025, 1.0) → 10.25
   ```

3. Ajoutez computeMonthlyInterest() dont le but est de calculer et de retourner la valeur des intérêts pour le montant et taux d'intérêts passés en paramètre pour **1 mois**

   Exemples :
   ```
   computeMonthlyInterest(1000, 1.0) → 0.83333333
   computeMonthlyInterest(2400, 2.5) → 5.0
   ```

4. Ajoutez `printCumulativeWins()` – simulation d'une épargne – dont le but est de calculer et d'afficher (imprimer dans la console) les gains **cumulés par an**, ainsi que le montant obtenu au fur-et-à-mesure pour une épargne. Le montant de départ et le taux d'intérêt sont les valeurs des attributs.

   Le principe est simple – pour chaque année (nombre total passé en paramètre) il faut :

   1. après une année, on calcule les intérêts sur le montant actuel
   2. ces intérêts sont ensuite ajoutés au montant actuel

   Note : les valeurs des attributs ne sont pas modifiées !

   Exemple :

```
Gains cummulés pour 1000.0€ à 2.5% (par an) pendant 5 années
année=1 --> 1000.0€ + 25.0€ = 1025.0€
année=2 --> 1025.0€ + 25.625€ = 1050.625€
année=3 --> 1050.625€ + 26.265625€ = 1076.890625€
année=4 --> 1076.890625€ + 26.922265625€ = 1103.812890625€
année=5 --> 1103.812890625€ + 27.595322265625€ = 1131.408212890625€
Gain = 131.408212890625€
```

5. Ajoutez `printLoanPayback()` – simulation d'un remboursement de crédit – dont le but est de calculer et d'afficher (imprimer dans la console) les intérêts **par mois**, ainsi que le montant obtenu au fur-et-à-mesure pour un crédit lorsqu'on rembourse (par mois) une certaine somme passée en paramètre. Le montant de départ et le taux d'intérêt sont les valeurs des attributs.

   Le principe est simple :

   1. après un mois, on calcule les intérêts sur le montant actuel
   2. après, ces intérêts sont ajoutés au montant actuel
   3. ensuite on soustrait (enlève) le somme remboursée (*payback*) du montant actuel
   4. les calculs et affichages sont faits à chaque mois, jusqu'à ce que le montant actuel a été totalement remboursé
   5. attention, à la fin il ne faut pas rembourser plus que nécessaire (cf exemple)

   Note : les valeurs des attributs ne sont pas modifiées !

   Exemple :

```
Crédit de 1000.0€ à 2.5% (par an) avec un remboursement de 100.0€ par mois
mois=1 --> 1000.0€ + 2.0833333333333335€ - 100.0€ = 902.0833333333334€
mois=2 --> 902.0833333333334€ + 1.8793402777777778€ - 100.0€ = 803.96267361111112€
mois=3 --> 803.96267361111112€ + 1.6749222366898149€ - 100.0€ = 705.6375958478016€
mois=4 --> 705.6375958478016€ + 1.4700783246829188€ - 100.0€ = 607.1076741724839€
mois=5 --> 607.1076741724839€ + 1.2648076545260083€ - 100.0€ = 508.37248182700996€
mois=6 --> 508.37248182700996€ + 1.0591093371396039€ - 100.0€ = 409.43159116414956€
mois=7 --> 409.43159116414956€ + 0.8529824815919781€ - 100.0€ = 310.2845736457415€
mois=8 --> 310.2845736457415€ + 0.6464261950952948€ - 100.0€ = 210.93099984083676€
mois=9 --> 210.93099984083676€ + 0.43943958300174324€ - 100.0€ = 111.37043942383852€
mois=10 --> 111.37043942383852€ + 0.23202174879966359€ - 100.0€ = 11.602461172638186€
mois=11 --> 11.602461172638186€ + 0.024171794109662886€ - 11.626632966747849€ = 0.0€
Intérêts payés en total = 11.626632966747799€
```

   Attention, dans cet exemple, le dernier remboursement n'est que de 11.63€ et pas de 100€ !

Pour les avancés…

Il faut ajouter une méthode `round(double val, int n)` qui permet d'arrondir des valeurs décimales à un nombre de places définies. Quelques exemples :

```
round(10.0449, 2)  →  10.04
round(10.045, 2)  →  10.05
round(10.046, 2)  →  10.05
round(10.049, 2)  →  10.05
round(10.050, 2)  →  10.05
round(10.051, 2)  →  10.05
round(10.049999999, 2)  →  10.05
round(10.049, 1)  →  10.0
round(10.05, 1)  →  10.1
round(10.4, 1)  →  10.4
round(10.49, 1)  →  10.5
round(10.50, 1)  →  10.5
round(10.99, 1)  →  11.0
```

Prenons deux exemples pour expliquer comment s'effectue cette opération (pour 2 places après la virgule) :

10.0449 * 100 = 1004.49	10.045 * 100 = 1004.5
1004.49 + 0.5 = 1004.99	1004.5 + 0.5 = 1005.0
partie entière : 1004.99 → 1004	partie entière : 1005.0 → 1005
1004 / 100 = 10.04	1005 / 100 = 10.05
résultat : 10.04	résultat : 10.05

Pour 1 place après la virgule il faut multiplier et puis diviser par 10 ; pour 3 places il faut utiliser la valeur 1000 ;pour 0 places il faut utiliser la valeur 1 ; pour 4 places il faut multiplier et diviser par 10000 ! La formule mathématique est assez simple… À noter que la valeur 0.5 reste toujours inchangée.