Cours MOOC EPFL d'introduction à la programmation orientée objet, illustré en C++

Premier devoir
Classes et objets

J.-C. Chappelier & J. Sam

1 Exercice 1 — IMC

Le but de cet exercice est de créer des « patients » qui ont un poids et une taille, et de calculer leur « Indice de Masse Corporelle » (IMC).

1.1 Description

Télécharger le programme `imc.cc` fourni et le compléter suivant les instructions données ci-dessous.

ATTENTION : vous ne devez en aucun cas modifier ni le début ni la fin du programme fourni, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc impératif de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `imc.cc` ou `imc.cpp` ;
2. écrire le code à fournir (voir ci-dessous) entre ces deux commentaires :

```
/*****************************************
 * Complétez le programme à partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier après cette ligne.
 *****************************************/
```

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs utilisées dans l'exemple de déroulement donné plus bas ;

4. soumettre le fichier modifié (toujours `imc.cc` ou `imc.cpp`) dans « My submission » puis « Create submission ».

Le code fourni

— crée un patient,
— lit son poids et sa taille depuis le clavier et les lui affecte ;
— affiche les données du patient ainsi que son IMC.

Les deux dernières étapes sont répétées tant que le poids et la taille ne sont pas nuls.

La définition de la classe `Patient` manque et c'est ce qu'il vous est demandé d'écrire.

Un patient est caractérisé par un poids et une taille. Vous nommerez les attributs correspondants respectivement `masse` et `hauteur` (veuillez respecter strictement ces consignes).

Par ailleurs, les méthodes spécifiques à un patient sont :

— une méthode `init` prenant en paramètre deux `double`, le premier pour initialiser le poids du patient et le second pour initialiser sa taille ;
ces données ne seront affectées aux attributs du patient que si elles sont toutes les deux positives ; dans le cas contraire, la taille et le poids du patient seront **tous deux initialisés à zéro** ; pour simplifier, il n'y a pas d'autre contrôle à faire sur ces données ;

— une méthode `afficher` permettant d'afficher sur le terminal les caractéristiques du patient en respectant ***strictement*** le format suivant :
`Patient : <poids> kg pour <taille> m`
où `<poids>` est à remplacer par le poids du patient et `<taille>` par sa taille ; **cet affichage sera terminé par un saut de ligne**.

— une méthode `poids` retournant le poids du patient ;

— une méthode `taille` retournant la taille du patient ;

— une méthode `imc` retournant l'IMC du patient : son poids divisé par le carré de sa taille ; **en cas de taille nulle, cette méthode retournera zéro**.

Ces méthodes feront partie de l'interface d'utilisation de la classe.

Un exemple de déroulement possible est fourni plus bas.

1.2 Exemple de déroulement

```
Entrez un poids (kg) et une taille (m) : 80.0 1.7
Patient : 80 kg pour 1.7 m
```

```
IMC : 27.6817
Entrez un poids (kg) et une taille (m) : 56.5 1.8
Patient : 56.5 kg pour 1.8 m
IMC : 17.4383
Entrez un poids (kg) et une taille (m) : 0 0
Patient : 0 kg pour 0 m
IMC : 0
```

2 Exercice 2 — Tirelire

Le but de cet exercice est de simuler une tirelire dans laquelle on stocke et retire de l'argent et que l'on souhaite utiliser pour payer un certain budget (de vacances, par exemple).

2.1 Description

Télécharger le programme `tirelire.cc` fourni et le compléter suivant les instructions données ci-dessous.

ATTENTION : vous ne devez en aucun cas modifier ni le début ni la fin du programme fourni, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc impératif de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `tirelire.cc` ou `tirelire.cpp` ;
2. écrire le code à fournir (voir ci-dessous) entre ces deux commentaires :

```
/*****************************************
 * Complétez le programme à partir d'ici.
 *****************************************/

/*****************************************
 * Ne rien modifier après cette ligne.
 *****************************************/
```

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs utilisées dans l'exemple de déroulement donné plus bas ;
4. soumettre le fichier modifié (toujours `tirelire.cc` ou `tirelire.cpp`) dans « My submission » puis « Create submission ».

Le code fourni crée une tirelire et lui fait subir divers manipulations (la vider, la secouer, en afficher le contenu etc.).

Ce programme demande aussi à l'utilisateur quel budget il aimerait consacrer à ses vacances.

Si la tirelire contient suffisamment d'argent (ce budget ou plus), il indique combien d'argent il resterait dans la tirelire après les vacances. Dans la cas contraire, il indique quel montant manque pour partir en vacances avec le budget souhaité.

La définition de la classe `Tirelire` manque et il vous est demandé de la fournir.

Une tirelire est caractérisée par le *montant* qu'elle contient. Vous nommerez cet attribut avec ce nom dans votre programme.

Les traitements qui lui sont spécifiques sont :

- une méthode `getMontant` retournant le montant de la tirelire ;
- une méthode `afficher` affichant les données de la tirelire sous le format suivant :
  - `Vous etes sans le sou.`
    si la tirelire ne contient pas d'argent
    (nous avons volontairement supprimé l'accent sur le 'e' de « êtes » ici ; veuillez respecter cette modification dans votre affichage) ;
  - `Vous avez : <montant> euros dans votre tirelire.`
    dans le cas contraire, où `<montant>` est le montant de la tirelire.
- une méthode `secouer` affichant sur le terminal le message « `Bing bing` », suivi d'un saut de ligne, dans le cas où la tirelire contient de l'argent, et qui n'affiche rien sinon ;
- la méthode `remplir` mettant un montant donné en paramètre (`double`) dans la tirelire. Seuls les montant positifs seront acceptés (dans le cas contraire on ne fait rien) ;
- une méthode `vider` (re)intialisant le montant de la tirelire à zéro ;
- une méthode `puiser` permettant de puiser dans la tirelire un montant donné en paramètre. Si le montant est négatif il sera ignoré. Si le montant en argument est plus grand que le montant disponible, la tirelire est alors vidée. La méthode `puiser` ne retourne rien.
- une méthode `montant_suffisant` à deux paramètres qui retourne `true` si la tirelire contient assez d'argent pour dépenser un budget donné en premier paramètre (type `double`) et `false` dans le cas contraire. S'il y a assez d'argent, le second paramètre `solde` (de type `double` et passé par référence) contiendra le montant qui resterait dans la tirelire si l'on y puisait le budget donné ; sinon il contiendra le montant (positif) manquant dans la tirelire pour arriver à ce budget.
  Si le budget est négatif (ou nul), la méthode `montant_suffisant` retourne `true` et le second paramètre `solde` est affecté au montant restant dans la tirelire : c'est comme si l'on demandait un budget nul.

— une méthode `calculerSolde` qui retourne la différence entre le montant de la tirelire et le budget que l'on souhaite dépenser (un `double`). Si le budget est négatif (ou nul), la méthode `calculerSolde` doit retourner le montant de la tirelire.

Ces méthodes feront partie de l'interface d'utilisation de la classe.

Deux exemples de déroulement possibles sont fournis plus bas.

2.2 Exemples de déroulement

```
Vous etes sans le sou.
Vous etes sans le sou.
Bing bing
Vous avez : 550 euros dans votre tirelire.
Vous avez : 535 euros dans votre tirelire.

Donnez le budget de vos vacances : 1000.0
Il vous manque 465 euros pour partir en vacances !
```

ou

```
Vous etes sans le sou.
Vous etes sans le sou.
Bing bing
Vous avez : 550 euros dans votre tirelire.
Vous avez : 535 euros dans votre tirelire.

Donnez le budget de vos vacances : 400.0
Vous êtes assez riche pour partir en vacances !
Il vous restera 135 euros à la rentrée.
```