Cours de C++, 1ère année, HE-Arc

Serie 2: Premières classes

Exercice 1: CLASSE Compte bancaire

On veut développer un programme de gestion d'un compte bancaire. Pour cela, implémenter une classe `BankAccount`, avec laquelle on puisse :

- Initialiser le montant à zéro
- Déposer de l'argent (il faut vérifier que le montant déposé soit positif)
- Retirer de l'argent (il faut vérifier que le montant retiré soit positif et pas supérieur au montant disponible sur le compte)

La classe peut être définie dans le fichier main.c

Avec le programme `main` ci-dessous :

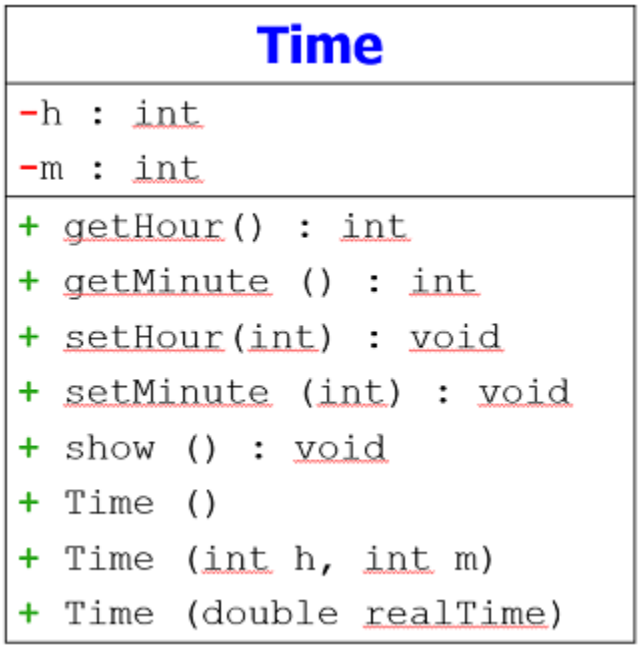
```
int main()
{
    BankAccount myBankAccount;
    myBankAccount.init();
    myBankAccount.show();
    myBankAccount.withdraw(100);    // impossible(opération annulée)
    myBankAccount.show();
    myBankAccount.deposit(100);
    myBankAccount.show();
    myBankAccount.withdraw(200);    // impossible(opération annulée)
    myBankAccount.show();
    myBankAccount.withdraw(20);
    myBankAccount.show();
}
```

le résultat sera le suivant:

```
The amount on your bank account is : 0.00
The amount on your bank account is : 0.00
The amount on your bank account is : 100.00
The amount on your bank account is : 100.00
The amount on your bank account is : 80.00

Process returned 0 (0x0)   execution time : 0.334 s
Press any key to continue.
```

Exercice 2: CLASSE Time

Créer une classe `Time` permettant de manipuler des mesures de temps (heure, minute) selon le diagramme UML suivant.



Elle disposera donc

• des attributs privés: `hour` , `minute`

• des constructeurs : par défaut(-->12H00), standard, et de conversion (5.75 --> 5H45')

• des accesseurs : `getHour()` , `getMinute()`

• des modificateurs : `setHour(h)` , `setHour(m)` qui valideront l'argument avant de modifier l'objet.

• de la méthode `show()` qui affichera heures et minutes avec 2 digits : ("18H45")

Utiliser une horloge comptant sur 24 heures.
Les valeurs excessives (>23, >59) seront remplacées par la valeur maximum possible
Les valeurs négatives seront considérées comme 0

La classe sera déclarée et définie dans des fichiers distincts: `Time.h`, `Time.cpp`

Tester cette classe dans un programme qui

- fasse appel aux différents constructeurs
- utilise les méthodes pour régler une heure à 16h33, ou à 16h87
- utilise accesseurs et modificateur pour augmenter un objet Time de 5 minutes
- affiche les objets `Time` avec la méthode `show`

```
Time t1;
Time t2(8,15), t3(11.25);

...
t1.setHour(36);
t1.show();
```

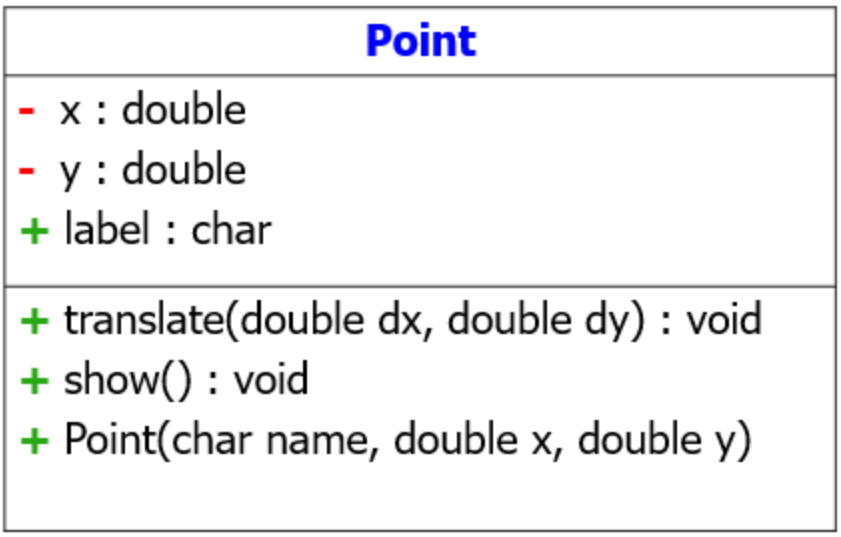
BONUS

Modifier la logique de contrôle de la validité des arguments aussi bien à la construction qu'à la modification d'un objet pour corriger une valeur excessive en reportant (minutes --> heures, bouclement des heures sur 24).

Exemple d'exécution (dernier affichage = bonus)

```
TEST DES CONSTRUCTEURS :

Time t1;   -> Appel du constructeur par defaut
12H00

Time t2(10,9);   -> Appel du constructeur standard
10H09

Time t3(17.75);   -> Appel du constructeur de conversion
17H45


TEST DES MODIFICATEURS :

t2.setHour(7); 07H09

t2.setMinute(-40); 07H00

t2.setMinute(86); 08H26
```

Exercice 3: CLASSE POINT



1) En s'inspirant des exemples du cours, concevoir puis implémenter une classe `Point` permettant de manipuler un point dans le plan.

Un point sera caractérisé par un label et ses coordonnées dans le plan.
La classe `Point` disposera des méthodes suivantes :

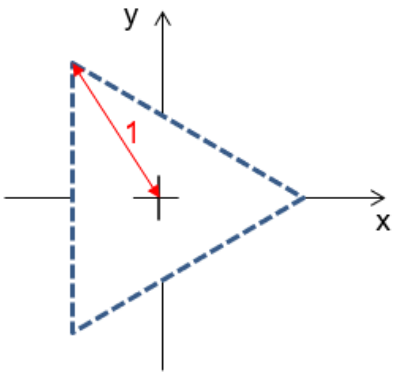
- Un constructeur qui permette de créer un point avec un label et de manière optionnelle spécifier ses coordonnées x,y (0 par défaut).
- Une méthode `show()` qui affiche le nom du point et ses coordonnées
- Une méthode `translate()` qui prend en arguments les composantes du vecteur de translation et modifie les attributs du point courant en conséquence.

La classe sera déclarée et définie dans des fichiers distincts: Point.h, Point.cpp

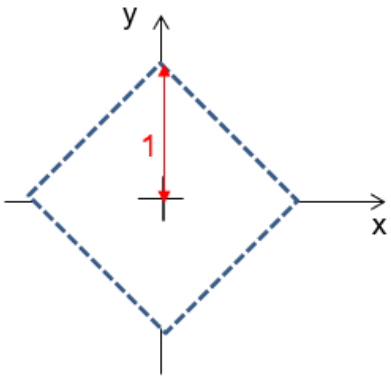
2) Ecrire une fonction `main()` permettant de tester la classe `Point` avec:

- instanciation de points avec chacun des constructeurs : Point p1, p2('A', 3, 4)
- affichage des points,
- translation de l'un des points et nouvel affichage.
- instanciation **dynamique** d'un objet point (syntaxe C++), et stockage de son adresse dans un pointeur
- affichage de ce point
- suppression de ce point (récupération de la mémoire)

*Ecrire la fonction `Point** generatePolygon(int n)` qui instancie dynamiquement autant de points que nécessaire à la réalisation d'un *polygone régulier* à n côtés, dont les points sont à une distance unitaire de l'origine. Par exemple : triangle(n=3), hexagone(n=6), ... La fonction renvoie un pointeur sur le tableau de pointeurs alloué dynamiquement qui contient les n pointeurs. Libérer la mémoire des points créés et du tableau de pointeurs avant la fin du programme. Note: le premier point sera toujours en (1;0), le second illustré en rouge (à un angle de 120°,90°,...,60° du premier, etc...)...

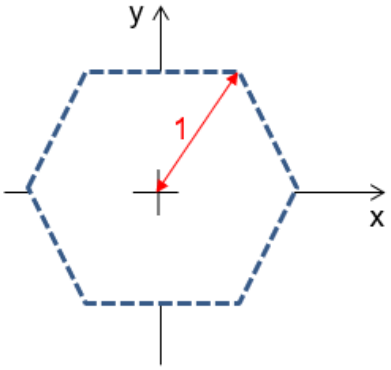


3



4

…



6