Master Informatique – 1ère année
Université Paris Descartes
F. CLOPPET/ D. GONZALEZ OBANDO

# POO Avancée - Langage C++ - 7 Novembre 2018

# Contrôle continu sur machine

## Consignes

1. Prenez un soin particulier pour écrire les programmes, toute portion de code illisible ou mal indentée ne sera pas corrigée.
2. N'oublier pas de lire les NB qui donnent les indications pour résoudre les exercices.
3. Respectez les noms donnés aux fonctions.
4. **Les documents, l'accès à internet, et la connexion de clés USB ne sont pas autorisés**
5. **Sauvegardez très régulièrement votre code …**

Lors de la mise à jour de l'infrastructure informatique de la Poste vous avez été embauché-e pour la conception du logiciel de gestion des commandes de timbres. Votre devoir pour ce contrôle sur machine sera de concevoir les structures de données nécessaires pour modéliser ce système de gestion de commandes. Les questions suivantes vous guideront dans la réalisation de ce programme. La plupart des questions sont indépendantes, si vous n'arrivez pas à répondre à l'une d'entre elles, vous pouvez quand même essayer les exercices suivants.

1/ On souhaite premièrement créer une classe Timbre pour représenter chaque timbre que les utilisateurs peuvent acheter. Cette classe possède les caractéristiques suivantes :

- Un identifiant du timbre (`int id`).
- Une valeur booléenne qui indique si le timbre est prioritaire ou standard (`bool prioritaire`).
- Les méthodes accesseurs pour les attributs de la classe Timbre.
- Une méthode `double calculerPrix()` qui calcule le prix du timbre. Le prix pour un timbre standard est 0€80 et 1€10 si le timbre est prioritaire.
- Une méthode `string toString()` qui renvoie une chaine avec les détails du timbre avec son id, les délai de livraison de la lettre (3 jours standard et 1 jour prioritaire) et son prix.

Seules les méthodes spécifiques sont indiquées ici. Il vous appartient de rajouter toutes méthodes utiles au bon fonctionnement de la classe.

2/ On souhaite créer aussi une classe `TimbrePersonnalise` pour représenter un nouveau type de timbre qui permettra aux acheteurs de personnaliser le timbre en rajoutant un message dans le timbre imprimé. Cette classe possède les caractéristiques suivantes :

- Un attribut `string message` qui contient le message à rajouter au timbre.
- Pour cette classe le prix du timbre, lors d'un appel à la méthode `calculerPrix()`, se calcule avec le prix d'un timbre simple plus une surcharge de 0€01 pour chaque caractère dans le message (espaces blancs inclus) si le timbre n'est pas prioritaire, et de 0€02 (par caractère) si le timbre est prioritaire.

- La méthode `toString()` qui renvoie une chaine avec les mêmes informations que celles de la méthode toString de la classe Timbre plus le message du timbre.
- 
- 

Seules les méthodes spécifiques sont indiquées ici. Il vous appartient de rajouter toutes méthodes utiles au bon fonctionnement de la classe.

**→Implémentez les clases Timbre et TimbrePersonnalise.**

3/ On souhaite ainsi créer la classe Carnet qui contiendra des timbres et comportera les informations indiquées ci-dessous :

- Un identifiant du carnet (`int id`).
- La quantité de timbres maximale dans le carnet (`int capacite`).
- Un tableau de timbres appartenant au carnet (`Timbre** timbres`).
- 
- Une méthode `int getNombreTimbresLibres()` qui permet de savoir la quantité de timbres qui reste à rajouter au carnet.
- Une méthode `int* getTimbresLibres()` qui renvoie les positions dans le tableau de timbres qui ne sont toujours pas associées à une instance de timbre.
- Une méthode `void rajouterTimbre(int pos, Timbre* t)` qui permet d'associer le timbre `t` au carnet dans la position `pos`.
- Une méthode `void supprimerTimbre(int idTimbre)` qui supprime le timbre avec identifiant `idTimbre`.
- Une methode `double getPrixCarnet()` qui calcule le prix total du carnet comme la somme des prix des timbres dans le carnet.
- La méthode `toString()` qui renvoie une chaine avec les informations du carnet : L'identifiant du carnet, la quantité de timbres dans le carnet et l'information de chaque timbre, ainsi que le prix du carnet.

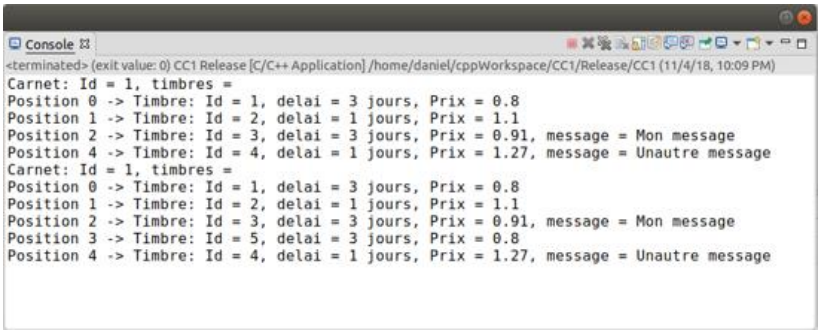
Seules les méthodes spécifiques sont indiquées ici. Il vous appartient de rajouter toutes méthodes utiles au bon fonctionnement de la classe.

**→Implémentez la classe Carnet. Prenez en compte que lors de la construction du carnet on indiquera seulement son *identifiant,* et la *capacité* du carnet.**

4/ On souhaite maintenant améliorer la classe Carnet afin d'enregistrer plus facilement des timbres en utilisant l'opérateur <, de tel sorte que c<t, rajoute le timbre t dans le carnet c à la première position libre dans le tableau de timbres.

**→Ajoutez l'opérateur < à la classe Carnet.**

Vous testerez votre programme avec le programme principal dans la page suivante et si tout a été bien programmé vous obtiendrez l'affichage suivant :




```cpp
#include <iostream>
using namespace std;

#include "Timbre.h"
#include "TimbrePersonnalise.h"
#include "Carnet.h"


int main() {
    Timbre* t1 = new Timbre(1, false);
    Timbre* t2 = new Timbre(2, true);
    Timbre* t3 = new TimbrePersonnalise(3, false, "Mon message");
    Timbre* t4 = new TimbrePersonnalise(4, true, "Unautre message");
    Timbre* t5 = new Timbre(5, false);

    Carnet* c = new Carnet(1, 5);
    c->rajouterTimbre(0, t1);
    c->rajouterTimbre(1, t2);
    c->rajouterTimbre(2, t3);
    c->rajouterTimbre(4, t4);

    cout << c->toString() << endl;

    (*c) < t5;

    cout << c->toString() << endl;

    delete c;
    delete t1;
    delete t2;
    delete t3;
    delete t4;
    delete t5;

    return 0;
}
```