Question 4 (30 points)

Dans cette exercice vous allez programmer une représentation d'un carnet d'adresses postales. Une adresse postale, peut avoir différents formats ; par exemple, pour un envoi international du Canada vers les USA, nous devons préciser le pays receveur de la lettre (ici USA) alors que pour un envoi à l'intérieur du Canada cela n'est pas nécessaire. Le format d'écriture de l'adresse est, quant à lui, dépendant du pays expéditeur. Une représentation simplifiée de ce format est comme suit:

à partir des USA	**à partir du Canada**
Nom de la personne Numéro et le nom de la rue Ville, État, et Code postal {Pays}	Nom de la personne Numéro et le nom de la rue Ville {Pays} Code postal

tableau -1-

Les { } indiquant un affichage optionnel: i.e. le pays ne sera pas nécessairement affiché pour un envoi interne.

Techniquement, ce carnet sera représenté par 4 classes, comme le montre le schéma d'héritage suivant:

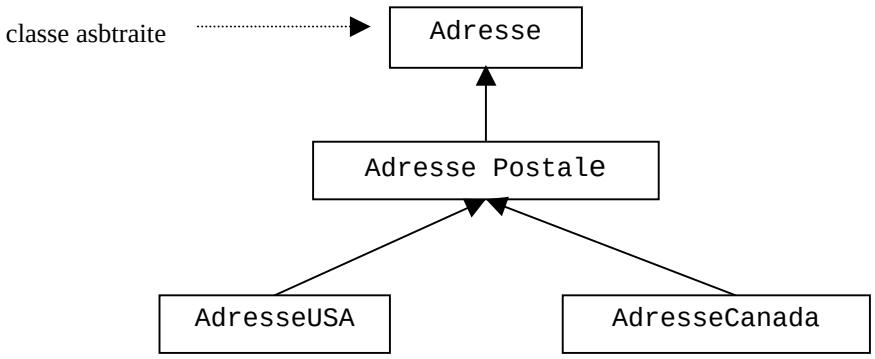

figure -1-

Q4.1 Définir la classe abstraite `Adresse` qui contient parmi ses membres les 3 fonctions virtuelles pures et publiques suivantes:

- `type`: sans argument, et qui retourne une chaîne de caractères spécifiant le type d'adresse auquel nous avons affaire (adresse postale, adresse courriel etc.).

- `affiche`: accepte un argument du type `int` et ne retourne rien. L'argument permet de spécifier si on a affaire à un envoi international, si c'est le cas, le pays destinataire sera affiché. Par défaut cet argument prendra la valeur 0, pour signifier que c'est un envoi intérieur. Cette fonction sert à afficher l'adresse postale comme indiqué dans le tableau -1-.

- `saisir`: sans argument et qui ne retourne aucune valeur. Cette fonction permet de saisir des données du clavier.

Q4.2 Définir la classe AdressePostale qui dérive publiquement de la classe Adresse, et qui doit respecter les contraintes suivantes:

- le principe d'encapsulation des données doit être respecté.
- le constructeur doit être accessible que par les classes dérivées.
- `AdressePostale` doit contenir les membres données suivants:

`char* nom;` // nom de la personne	`char* rue;` // nom de la rue
`char* numero;` // numéro de la maison	`char* ville;` // nom de la ville
`char* codepostale;` // le code postale	`char* pays;` // nom du pays

- pour chaque membre donnée, vous devez définir deux méthodes:

  `saisir`: accepte un argument du type chaîne de caractères et ne retourne rien. Cette fonction permet de passer l'information au membre donnée. À titre d'exemple pour le membre donnée `nom`, cette fonction portera le nom de: `saisirnom` (pour `numero`: `saisirnumero` etc.). Au total vous devez écrire 6 fonctions "saisir". Lors de la correction, il sera tenu compte de l'optimisation de votre code.

  `affiche`: sans argument ne retourne rien. Cette fonction permet d'afficher l'information sur le membre donnée concerné. À titre d'exemple pour le membre donnée `nom`, cette fonction portera le nom de: `affichenom` (pour `numero`: `affichenumero` etc.). Au total vous devez écrire 6 fonctions "affiche".

- la définition de la fonction `type`, décrite en Q4.1.

De la classe `AdressePostale` vont dériver deux classes: `AdresseCanada` et `AdresseUSA` (voir figure -1-, à la page 8).

Q4.3. Écrire la définition de la classe `AdresseUSA`. Parmi ses fonctions membres, les fonctions: `affiche` et `saisir` décrites en Q4.1 et en tableau -1- et que vous devez définir à ce niveau.

Q4.4. Écrire la définition de la classe `AdresseCanada`. Parmi ses fonctions membres, les fonctions: `affiche` et `saisir` décrites en Q4.1 et en tableau -1- et que vous devez définir à ce niveau.