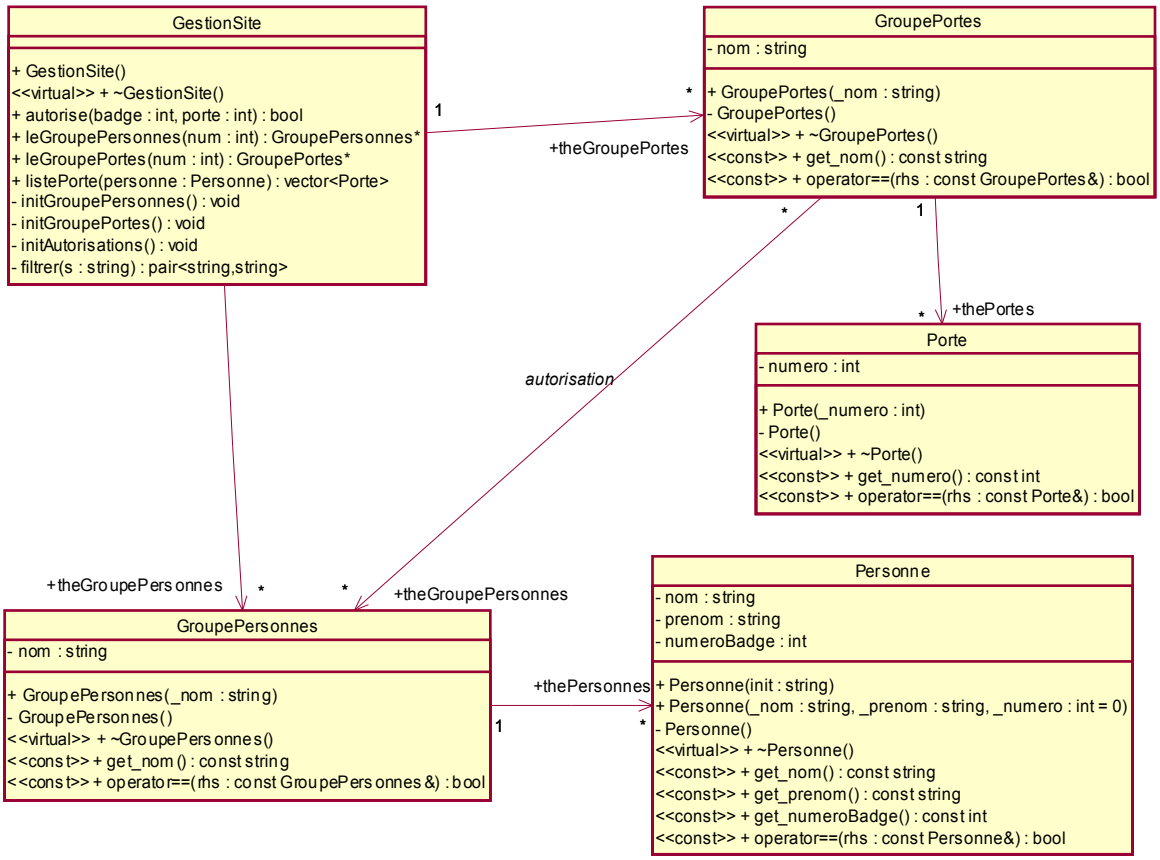
TP STL

Sujet : Contrôle d'accès à un bâtiment

Il a été décidé de restreindre l'accès à un site par l'installation de portes à fermeture automatiques. L'ouverture de chacune de ces portes est commandé par un lecteur de badge. Chaque zone disposant d'un ensemble de portes. Les personnes reçoivent chacun un badge et sont regroupés dans des groupes de personnes, de même les portes sont regroupées par groupe, afin de simplifier la gestion des autorisations d'accès aux zones du site.

Le site mis en œuvre ici est séparé en 4 zones : administration, rez de chaussée, 1er étage et 2ième étage. Les groupes de personnes étant des : administratifs, professeurs, étudiants, stagiaires.

La conception du modèle de base a été réalisée et correspond au diagramme de classes suivant



1) Donner pour chaque rôle la collection de la STL que l'on pourrait choisir et donner les critères de choix associés.

L'application est codée sous Visual C++ 6.0 en mode console, le squelette des classes ayant été obtenu par génération du code C++ à partir de l'AGL Rose 2002.
Ces classes sont persistantes et leur initialisation est réalisée à l'aide des fichiers textes suivants : gestionPortes.txt, administration.txt, rezChaussee.txt, 1erEtage.txt, 2iemeEtage.txt, groupePersonnes.txt, administratifs.txt, etudiants.txt, stagiaires.txt, professeurs.txt, autorisations.txt.
L'initialisation de l'application est déjà codée et fait fortement appel à la STL.

Remarque : contrairement à l'analyse de la 1er question, nous prendrons pour raisons de simplification la collection **vector** pour toutes les associations (1,*).

2) Mettre des commentaires pour chaque ligne du source de la méthode suivante :

```
void GestionSite::initGroupePortes()                    .....
{                                                       .....
    ifstream f1;                                        .....
    vector<string> gpNom;                               .....

    f1.open(fGroupePortes.c_str());                     .....
    copy(                                               .....
        istream_iterator<string>(f1),                   .....
        istream_iterator<string>(),                     .....
        back_inserter( gpNom )                          .....
    );                                                  .....
    f1.close();

    for (int i=0; i < gpNom.size(); i++) {              .....
        string s = path + gpNom[i] + ".txt";            .....
        GroupePortes gp ( gpNom[i] );                   .....
        f1.clear();                                     .....
        f1.open( s.c_str() );                           .....
        copy(                                           .....
            istream_iterator<int>(f1),                  .....
            istream_iterator<int>(),                    .....
            back_inserter( gp.thePortes )               .....
        );
        f1.close();                                     .....
        theGroupePortes.push_back (gp);                 .....
    }
}
```

3) Ecrire la fonction **listerPersonnesParGroupe(GestionSite& site)** qui permet d'afficher par groupe : le nom du groupe et les noms et prénoms de chaque personne de ce groupe :
   a. En utilisant l'opérateur [] des collections vector,
   b. En utilisant un itérateur const_iterator pour parcourir ces collections.

4) Ecrire la fonction **bool badgeAttribue(GestionSite& site, int num)** qui retourne le fait qu'un badge est ou non déjà attribué.

Autres question possibles :

5) Ecrire les fonctions qui permettent de savoir :
   - à quel groupe une porte appartient-elle ?
   - à quel groupe un badge appartient-il ?
   - si un badge donne l'accès à une porte !

Corrigé :

a)

```
void listerPersonnesParGroupe(GestionSite& site)
{
	for (int i=0; i < site.theGroupePersonnes.size(); i++) {
		cout << site.theGroupePersonnes[i].get_nom() << ":" << endl;
		for (int j=0; j < site.theGroupePersonnes[i].thePersonnes.size(); j++) {
			cout << site.theGroupePersonnes[i].thePersonnes[j].get_nom() << " "
				<< site.theGroupePersonnes[i].thePersonnes[j].get_prenom() << ", ";
		}
		cout << endl;
	}
}
```

b)

```
void listerPersonnesParGroupe(GestionSite& site)
{
	vector<GroupePersonnes>::const_iterator itgp;
	vector<Personne>::const_iterator itp;

	for (itgp = site.theGroupePersonnes.begin(); itgp != site.theGroupePersonnes.end(); itgp++) {
		cout << (*itgp).get_nom() << ":" << endl;
		for (itp = (*itgp).thePersonnes.begin(); itp != (*itgp).thePersonnes.end(); itp++) {
			cout << (*itp).get_nom() << " " << (*itp).get_prenom() << ", ";
		}
		cout << endl;
	}
}
```

Résultat :

administratifs:
Bourdelle Rene, Luckner Nicolas, Granet Frantois, Planchon Roger, Hora Josef, Horney Karen, Jacob Max,
etudiants:
Tonk Levis, Malos Hector, Simenon Georges, Grock Adrien, Cauchy Augustin, Grothendieck Alexandre, Grisi Carlotta, Malory Thomas, Beckmann Max, Ango Jean, Noyes Alfred,
professeurs:
Mique Richard, Desnos Robert, Solvay Ernest, Minne Georges, Tonnies Ferdinand,
stagiaires:
Peirce Charles, Milton John, Chardin Jean, Dumas Jean-pierre,
Press any key to continue

4)
cout << (badgeAttribuee(site, 14)? "trouve" : "pas trouve") << endl;

```
bool carteAttibuee(GestionSite& site, int num) {
	vector<GroupePersonnes>::const_iterator itgp;

	for (itgp = site.theGroupePersonnes.begin(); itgp != site.theGroupePersonnes.end(); itgp++) {
		if (find(
			(*itgp).thePersonnes.begin(),
			(*itgp).thePersonnes.end(),
			Personne("","",num)
		    ) != (*itgp).thePersonnes.end()
		)
		return true;
	}
/* ou
	for (int i=0; i< site.theGroupePersonnes.size(); i++) {
		if ( find(
			site.theGroupePersonnes[i].thePersonnes.begin(),
			site.theGroupePersonnes[i].thePersonnes.end(),
			Personne("","",num)
		     ) != site.theGroupePersonnes[i].thePersonnes.end())
		return true;
	}
*/
	return false;
}
```