Rapport

Projet d’C++

Système de gestion des joueurs



WANG Yifei

AMANGAR YASMINE

1. **Description d’application développé**

Selon le thème de la Coupe du monde, nous avons choisi de développer un système de gestion des membres de l’équipe, qui vise à ajouter, supprimer, interroger et autres travaux sur les entraîneurs, les joueurs et le personnel de soutien d’une équipe par le biais de notre programme. Le processus de demande comprend 8 classes, 3 niveaux d’héritage, le TSL utilisé comprend : <vector>, <move>, <set>, <list>, et les détails seront couverts dans la partie II.

1. **Mettre en valeur l’utilisation des contraintes**
2. **Class Object**

Cette classe est la classe parente de toutes les classes, où seules deux fonctions virtuelles pures sont utilisées pour renvoyer des noms de type et des informations d’objet, et une surcharge d’opérateur est utilisée pour générer des informations d’objet dans Object.

Deux fonctions virtuelles pures :

**virtual std::string getTypeName()const = 0;**

**virtual std::string toString()const = 0;**

1. **Class Person**

Cette classe hérite de la classe « Object », qui peut définir le nom et l’âge de la personne et peut déplacer l’objet via la fonction move dans STL.

Deux nouvelles propriétés, **m\_name** et **m\_age**

Nouvelles méthodes：

**void setName(const std::string&);**

**std::string getName()const;**

**void setAge(int);**

**int getAge()const;**

1. **Class Member**

Cette classe hérite de la classe « Person », dont nos prochaines classes sont dédiées à la gestion du personnel d’équipe.

Une nouvelle propriété, **m\_employer**

Nouvelles méthodes：

**std::string getEmployer()const;**

**void setEmployer(const std::string&);**

Opérateur surchargé **==** : Détermine si deux objets member sont identiques

**bool operator ==(const Member&);**

1. **Class Player**

Cette classe hérite de la classe « Member », qui permet de gérer les joueurs de l’équipe.

Propriétés **m\_number** et **m\_role** neuves

Nouvelles méthodes：

**void setRole(const std::string&);**

**std::string getRole()const;**

**bool setNumber(unsigned int);**

**unsigned int getNumber()const;**

Le conteneur **set std::set<int> m\_set** . Parce que le **m\_number** de chaque joueur est unique, **setNumber** peut Player de la valeur du nombre modifié, si la valeur n’existe pas dans le conteneur, elle peut être modifiée, sinon elle ne peut pas être modifiée.

1. **Class Coach**

Cette classe hérite de la classe « Member », qui sert à gérer les entraîneurs de l’équipe

Propriétés neuves **m\_years** et **m\_bChief**

Nouvelles méthodes：

**void setChiefCoach(bool = true);**

**bool IsChiefCoach()const;**

**void setYearsOfCoaching(int years);**

**int getYearsOfCoaching() const;**

Puisqu’il n’y a qu’un seul entraîneur-chef, le nouveau membre de la classe **Coach\* m\_pChiefCoach**;

Ajouter une fonction statique correspondante:

**static Coach\* getChiefCoach();**

1. Class Logistics

Cette classe hérite de la classe « Member », qui permet de gérer les logisticiens de l’équipe.

Utiliser le conteneur **vector<string>** pour enregistrer la tâche

surchargez l’**operator** **+=** pour ajouter la tâche

surcharger l’**operator** **-=** pour supprimer la tâche

1. Class IManage

Cette classe est utilisée pour gérer les membres, et les membres peuvent être récupérés, ajoutés et supprimés.

Quatre méthodes sont définies:

**virtual bool findByName(const std::string& , std::vector<Member\*>& ) = 0;**

**virtual bool AddMembers(Member\*) = 0;**

**virtual bool RemoveMembers(Member\*) = 0;**

**virtual size\_t getCount()const = 0 ;**

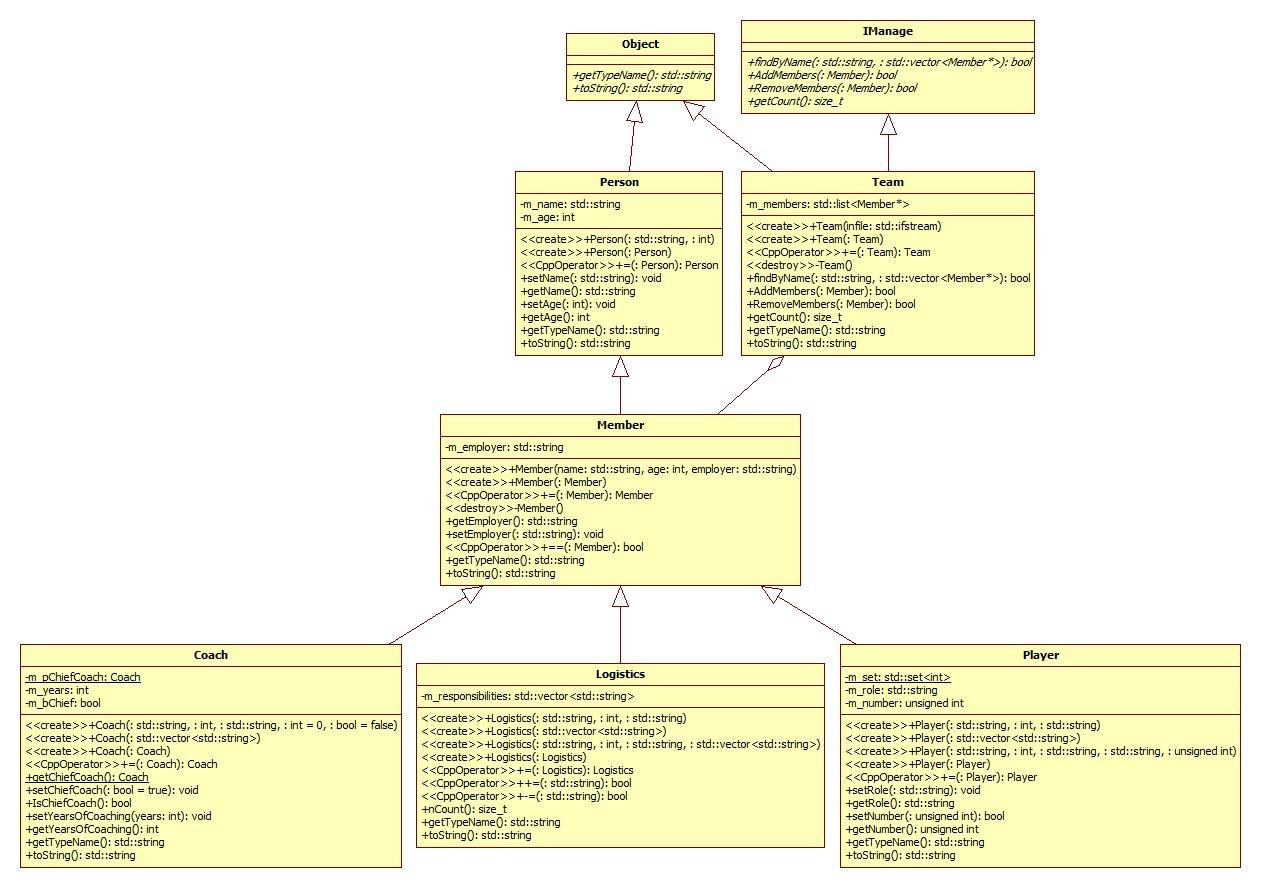
1. Class Team

Cette classe est une sous-classe de « Member » et « IManage », qui est utilisée pour la gestion d’équipe et implémente les quatre méthodes dans IManage.

Utilisez le conteneur **list<Member\*>** pour enregistrer les informations du Member de l’équipe.

Fournissez à un constructeur le paramètre **std::ifstream&** pour charger les informations de Member à partir du fichier.

1. Diagramme UML de l’application



1. Les bibliothèque et d’execution du code

IV.1 Les bibliothèque

#include <sstream>

#include <fstream>

#include <list>

#include <cctype>

#include <vector>

#include <string>

#include <cassert>

#include <iostream>

#include <set>

IV.2 D’execution du code

Nous testons notre programme avec <main.cc> fichiers, les fonctions de test incluent:

void TestPlayers()

void TestCoach()

void TestLogistics()

void TestTeam()

Voici comment l’exécuter :

1. Ouvrez le dossier contenant tout le code du système Linux
2. Ouvrez Terminal dans ce dossier
3. Entrez la commande « make » pour compiler tout le code
4. Entrez « ./demo », exécutez le code de test et le test est correct

