Prog2 NHF

Dokumentáció

AUTHOR
Sánta Dániel

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

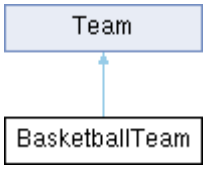
File List

Here is a list of all files with brief descriptions:

Class Documentation

BasketballTeam Class Reference

`#include <teams.hpp>`
Inheritance diagram for BasketballTeam:



Public Member Functions

- **BasketballTeam** (std::string name)
  *Construct a new Basketball **Team** object.*

- bool **addCheerleader** (const **Person** &cheerleader)
  Adds a Cheerleader to the team.

- bool **removeCheerleader** (const **Person** &cheerleader)
  Removes a Cheerleader from the team.

- **Person** & **getCheerleader** (size_t idx)
  Get the Cheerleader with the given index.

- size_t **numberOfCheerleaders** ()
  Get the Number of the Cheerleaders.

- std::ostream & **print** (std::ostream &os)
  Writes the Basketball team to the ostream.

- **~BasketballTeam** ()
  *Destroy the Basketball **Team** object.*

Public Member Functions inherited from Team

- **Team** (std::string name)
  *Construct a new **Team** object.*

- bool **addPlayer** (const **Person** &player)
  Adds a new player to the team.

- bool **removePlayer** (const **Person** &player)
  Removes a player from the team.

- bool **operator==** (const **Team** &team)
- **Person** & **getPlayer** (size_t idx)

Get the Player object with the given index.

- std::string **getName** () const
  Get the Name.

- size_t **size** () const
  Get the Size.

- virtual std::ostream & **print** (std::ostream &os)
  Writes the team to the ostream.

- virtual **~Team** ()
  *Destroy the **Team** object.*

Constructor & Destructor Documentation

BasketballTeam::BasketballTeam (std::string *name*) `[inline]`

Construct a new Basketball **Team** object.

Parameters

name	The team's name.

BasketballTeam::~BasketballTeam () `[inline]`

Destroy the Basketball **Team** object.

Member Function Documentation

bool BasketballTeam::addCheerleader (const Person & *cheerleader*)

Adds a Cheerleader to the team.

Parameters

cheerleader	

Returns

true If the Cheerleader is added.

false If the Cheerleader is already in the team.

Person & BasketballTeam::getCheerleader (size_t *idx*)

Get the Cheerleader with the given index.

Parameters

idx	the index of the needed cheerleader.

Returns

Person& A refenrece to the cheerleader with the given index.

size_t BasketballTeam::numberOfCheerleaders ()

Get the Number of the Cheerleaders.

Returns

The number.

std::ostream & BasketballTeam::print (std::ostream & *os*) `[virtual]`

Writes the Basketball team to the ostream.

Reimplemented from **Team** (*p.28*).

bool BasketballTeam::removeCheerleader (const Person & *cheerleader*)

Removes a Cheerleader from the team.

Parameters

cheerleader	

Returns

true If the Cheerleader is removed.

false If the Cheerleader is not in the team.

The documentation for this class was generated from the following files:

- headers/**teams.hpp**
- src/**teams.cpp**

Facility Class Reference

This is a heterogeneous collection.
`#include <facility.hpp>`

Public Member Functions

- bool **addTeam** (**Team** *team)
- bool **removeTeam** (**Team** team)
- bool **setCurrentTeam** (const size_t &n)
- size_t **getCurrentTeam** () const
- size_t **size** () const
- **Team** * **operator[]** (size_t idx)
- template<typename C > C * **getTeam** (size_t idx)
  Custom operator that returns the type of the team too.

- std::ostream & **listFootballTeams** (std::ostream &os)
  Lists the Football teams to the ostream.

- std::ostream & **listBasketballTeams** (std::ostream &os)
  Lists the Basketball teams to the ostream.

- std::ostream & **listHandballTeams** (std::ostream &os)
  Lists the Handball teams to the ostream.

- **~Facility** ()
  Destructor.

Detailed Description

This is a heterogeneous collection.

Constructor & Destructor Documentation

Facility::~Facility () `[inline]`

Destructor.

Member Function Documentation

bool Facility::addTeam (Team * *team*)

Parameters

The	team that the caller wants to add.

Returns

true if the team is added.

size_t Facility::getCurrentTeam () const

Returns

the index of the Current team.

template<typename C > C * Facility::getTeam (size_t *idx*) [inline]

Custom operator that returns the type of the team too.

Parameters

idx	the index of the team that the caller wants.

Returns

A team pointer with the current team type.

std::ostream & Facility::listBasketballTeams (std::ostream & *os*)

Lists the Basketball teams to the ostream.

std::ostream & Facility::listFootballTeams (std::ostream & *os*)

Lists the Football teams to the ostream.

std::ostream & Facility::listHandballTeams (std::ostream & *os*)

Lists the Handball teams to the ostream.

Team * Facility::operator[] (size_t *idx*)

Parameters

index.	

Returns

the the team with this index.

bool Facility::removeTeam (Team *team*)

Parameters

The	team that the caller wants to remove.

Returns

true if the team is removed.

bool Facility::setCurrentTeam (const size_t & *n*)

Parameters

The	index of the new current **Team**.

Returns

true if the change happened.

size_t Facility::size () const

Returns

the size of the **Facility**.

The documentation for this class was generated from the following files:

- headers/**facility.hpp**
- src/**facility.cpp**

FileHandler Class Reference

`#include <file.hpp>`

Public Member Functions

- **FileHandler** (std::string fileF, std::string fileB, std::string fileH)
  Construct a new File Handler object.

- std::vector< **FootballTeam** * > **readFootballTeams** ()
  Reads all the Football teams from fileF.

- std::vector< **BasketballTeam** * > **readBasketballTeams** ()
  Reads all the Basketball teams from fileB.

- std::vector< **HandballTeam** * > **readHandballTeams** ()
  Reads all the Handball teams from fileH.

- void **writeFootballTeam** (std::vector< **FootballTeam** > ft)
  At the end of the program writes the changed Football teams back to FileF.

- void **writeBasketballTeam** (std::vector< **BasketballTeam** > ft)
  At the end of the program writes the changed Basketball teams back to FileB.

- void **writeHandballTeam** (std::vector< **HandballTeam** > ft)
  At the end of the program writes the changed Handball teams back to FileH.

- **Facility readFacility** ()
  Reads all the teams into the facility.

- void **writeFacility** (**Facility** changedFacility)
  Separates all the teams and writes them back to the files one by one.

Constructor & Destructor Documentation

FileHandler::FileHandler (std::string *fileF*, std::string *fileB*, std::string *fileH*) `[inline]`

Construct a new File Handler object.

Parameters

fileF	The name and location of the file that contains the Football teams.
fileB	The name and location of the file that contains the Basketball teams.
fileH	The name and location of the file that contains the Handball teams.

Member Function Documentation

std::vector< BasketballTeam * > FileHandler::readBasketballTeams ()

Reads all the Basketball teams from fileB.

Returns

A vector that contains all the Basketball teams.

Facility FileHandler::readFacility ()

Reads all the teams into the facility.

Returns

The facility that was created from all the different files.

std::vector< FootballTeam * > FileHandler::readFootballTeams ()

Reads all the Football teams from fileF.

Returns

A vector that contains all the Football teams.

std::vector< HandballTeam * > FileHandler::readHandballTeams ()

Reads all the Handball teams from fileH.

Returns

A vector that contains all the Handball teams.

void FileHandler::writeBasketballTeam (std::vector< BasketballTeam > *ft*)

At the end of the program writes the changed Basketball teams back to FileB.

Parameters

ft	the separated Basketball teams.

void FileHandler::writeFacility (Facility *changedFacility*)

Separates all the teams and writes them back to the files one by one.

Parameters

changedFacility	the changed facility at the and of the program.

void FileHandler::writeFootballTeam (std::vector< FootballTeam > *ft*)

At the end of the program writes the changed Football teams back to FileF.

Parameters

ft	the separated Football teams.

void FileHandler::writeHandballTeam (std::vector< HandballTeam > *ft*)

At the end of the program writes the changed Handball teams back to FileH.

Parameters

ft	the separated Handball teams.

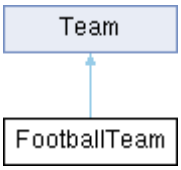
The documentation for this class was generated from the following files:

- headers/**file.hpp**
- src/**file.cpp**

FootballTeam Class Reference

`#include <teams.hpp>`

Inheritance diagram for FootballTeam:



Public Member Functions

- **FootballTeam** (std::string name)
  *Construct a new Football **Team** object.*

- bool **addCoach** (const **Person** &coach)
  Adds a Coach to the team.

- bool **removeCoach** (const **Person** &coach)
  Removes a Coach from the team.

- **Person** & **getCoach** (size_t idx)
  Get the Coach with the given index.

- size_t **numberOfCoaches** () const
  Get the Number of the coaches. (0, 1 or 2)

- std::ostream & **print** (std::ostream &os)
  Writes the Football team to the ostream.

- **~FootballTeam** ()
  *Destroy the Football **Team** object.*

Public Member Functions inherited from Team

- **Team** (std::string name)
  *Construct a new **Team** object.*

- bool **addPlayer** (const **Person** &player)
  Adds a new player to the team.

- bool **removePlayer** (const **Person** &player)
  Removes a player from the team.

- bool **operator==** (const **Team** &team)
- **Person** & **getPlayer** (size_t idx)
  Get the Player object with the given index.

- std::string **getName** () const
  Get the Name.

- size_t **size** () const
  Get the Size.

- virtual std::ostream & **print** (std::ostream &os)
  Writes the team to the ostream.

- virtual **~Team** ()
  *Destroy the **Team** object.*

Constructor & Destructor Documentation

FootballTeam::FootballTeam (std::string *name*) `[inline]`

Construct a new Football **Team** object.

Parameters

name	The team's name.

FootballTeam::~FootballTeam () `[inline]`

Destroy the Football **Team** object.

Member Function Documentation

bool FootballTeam::addCoach (const Person & *coach*)

Adds a Coach to the team.

Parameters

coach	

Returns

true If the Coach is added.

false If the Coach is already in the team.

Person & FootballTeam::getCoach (size_t *idx*)

Get the Coach with the given index.

Parameters

idx	the index of the needed coach.

Returns

Person& A refenrece to the coach with the given index.

size_t FootballTeam::numberOfCoaches () const

Get the Number of the coaches. (0, 1 or 2)

Returns

The number.

std::ostream & FootballTeam::print (std::ostream & *os*) `[virtual]`

Writes the Football team to the ostream.

Reimplemented from **Team** (*p.28*).

bool FootballTeam::removeCoach (const Person & *coach*)

Removes a Coach from the team.

Parameters

coach	

Returns

true If the Coach is removed.

false If the Coach is not in the team.

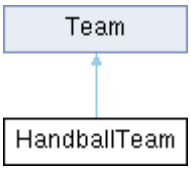
The documentation for this class was generated from the following files:

- headers/**teams.hpp**
- src/**teams.cpp**

HandballTeam Class Reference

`#include <teams.hpp>`

Inheritance diagram for HandballTeam:



Public Member Functions

- **HandballTeam** (std::string name, size_t value=0)
  *Construct a new Handball **Team** object.*

- void **changeSupportMoney** (size_t newValue)
  Changes the value of the Yearly Support money to a new given value.

- size_t **getValue** () const
  Get the Value.

- std::ostream & **print** (std::ostream &os)
  Writes the Handball team to the ostream.

- **~HandballTeam** ()
  *Destroy the Handball **Team** object.*

Public Member Functions inherited from Team

- **Team** (std::string name)
  *Construct a new **Team** object.*

- bool **addPlayer** (const **Person** &player)
  Adds a new player to the team.

- bool **removePlayer** (const **Person** &player)
  Removes a player from the team.

- bool **operator==** (const **Team** &team)
- **Person** & **getPlayer** (size_t idx)
  Get the Player object with the given index.

- std::string **getName** () const
  Get the Name.

- size_t **size** () const
  Get the Size.

- virtual std::ostream & **print** (std::ostream &os)
  Writes the team to the ostream.

- virtual **~Team** ()
  *Destroy the **Team** object.*

Constructor & Destructor Documentation

HandballTeam::HandballTeam (std::string *name*, size_t *value* = 0) [inline]

Construct a new Handball **Team** object.

Parameters

name	The name of the team.
value	The value of the Yearly Support money.

HandballTeam::~HandballTeam () [inline]

Destroy the Handball **Team** object.

Member Function Documentation

void HandballTeam::changeSupportMoney (size_t *newValue*)

Changes the value of the Yearly Support money to a new given value.

Parameters

newValue	The new value of the money.

size_t HandballTeam::getValue () const

Get the Value.

Returns

The money.

std::ostream & HandballTeam::print (std::ostream & *os*) [virtual]

Writes the Handball team to the ostream.

Reimplemented from **Team** (*p.28*).

The documentation for this class was generated from the following files:

- headers/**teams.hpp**
- src/**teams.cpp**

Menu Class Reference

`#include <menu.hpp>`

Public Member Functions

- **Menu** (std::vector< std::string > buttons)
  *Construct a new **Menu** object.*

- void **setState** (const **State** &n)
  Set the State.

- **State getState** () const
  Get the State.

- std::ostream & **print** (std::ostream &os)
  *Writes the **Menu** to the ostream.*

- size_t **select** (std::istream &is)
  Gets a number from the istream and checks if that number can push a button or not.

Constructor & Destructor Documentation

Menu::Menu (std::vector< std::string > *buttons*) `[inline]`

Construct a new **Menu** object.

Parameters

buttons	A vector that contaions all the Buttons in order.

Member Function Documentation

State Menu::getState () const

Get the State.

Returns

State

std::ostream & Menu::print (std::ostream & *os*)

Writes the **Menu** to the ostream.

size_t Menu::select (std::istream & *is*)

Gets a number from the istream and checks if that number can push a button or not.

Returns

size_t the number of the Selected button.

void Menu::setState (const State & *n*)

Set the State.

Parameters

n	The next state.

The documentation for this class was generated from the following files:

- headers/**menu.hpp**
- src/**menu.cpp**

MenuHandler Class Reference

This is a heterogeneous collection.
`#include <menu.hpp>`

Public Member Functions

- **MenuHandler** (std::vector< **Menu** > menus)
  *Construct a new **Menu** Handler object.*

- void **addMenu** (**Menu** newMenu)
  Adds a new menu to the vector.

- void **changeMenu** (**Menu** oldMenu, **Menu** newMenu)
  Deletes the old menu and replaces it with the new one.

- void **toPrevMenu** ()
  Moves the menu forwards.

- void **toNextMenu** ()
  Moves the menu backwards.

- size_t **getIndex** () const
  Get the Index.

- void **setState** (const **State** &n)
  Set the State.

- **State getState** () const
  Get the State.

- **Menu** & **operator[]** (size_t i)
- std::ostream & **print** (std::ostream &os)
  Writes the menu to the ostream.

- size_t **select** (std::istream &is)
  Gets a number from the istream and checks if that number can push a button or not.

Detailed Description

This is a heterogeneous collection.

Constructor & Destructor Documentation

MenuHandler::MenuHandler (std::vector< Menu > *menus*) [inline]

Construct a new **Menu** Handler object.

Parameters

menus	A vector that contains all the menus.

Member Function Documentation

void MenuHandler::addMenu (Menu *newMenu*)

Adds a new menu to the vector.

Parameters

newMenu	

void MenuHandler::changeMenu (Menu *oldMenu*, Menu *newMenu*)

Deletes the old menu and replaces it with the new one.

Parameters

oldMenu	
newMenu	

size_t MenuHandler::getIndex () const

Get the Index.

Returns

size_t The index of the current menu.

State MenuHandler::getState () const

Get the State.

Returns

The current State of the menu.

Menu & MenuHandler::operator[] (size_t *i*)

Parameters

i	index.

Returns

Menu& A reference to the current menu.

std::ostream & MenuHandler::print (std::ostream & *os*)

Writes the menu to the ostream.

size_t MenuHandler::select (std::istream & *is*)

Gets a number from the istream and checks if that number can push a button or not.

Returns

size_t the number of the Selected button.

void MenuHandler::setState (const State & *n*)

Set the State.

Parameters

n	The state of the new menu.

void MenuHandler::toNextMenu ()

Moves the menu backwards.

void MenuHandler::toPrevMenu ()

Moves the menu forwards.

The documentation for this class was generated from the following files:

- headers/**menu.hpp**
- src/**menu.cpp**

Person Class Reference

`#include <person.hpp>`

Public Member Functions

- **Person** (std::string name, size_t age)
  *Construct a new **Person** object.*

- std::string **getName** () const
  Get the Name.

- size_t **getAge** () const
  Get the Age.

- bool **operator==** (const **Person** &person) const
- bool **operator!=** (const **Person** &person) const

Constructor & Destructor Documentation

Person::Person (std::string *name*, size_t *age*) `[inline]`

Construct a new **Person** object.

Parameters

name	The name of the **Person**.
age	The age of the **Person**.

Member Function Documentation

size_t Person::getAge () const

Get the Age.

Returns

The age of the **Person**.

std::string Person::getName () const

Get the Name.

Returns

The name of the **Person**.

bool Person::operator!= (const Person & *person*) const

Parameters

person	the Pesron that the caller wants to compair the object with.

Returns

true The two People are different.

false The two People are the same.

bool Person::operator== (const Person & *person*) const

Parameters

person	the Pesron that the caller wants to compair the object with.

Returns

true The two People are the same.

false The two People are different.

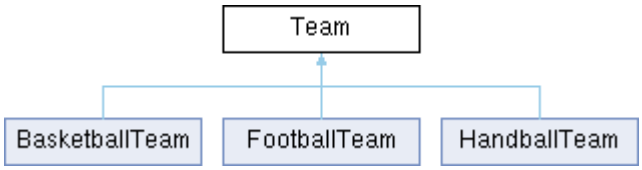
The documentation for this class was generated from the following files:

- headers/**person.hpp**
- src/**person.cpp**

Team Class Reference

`#include <teams.hpp>`

Inheritance diagram for Team:



Public Member Functions

- **Team** (std::string name)
  *Construct a new **Team** object.*

- bool **addPlayer** (const **Person** &player)
  Adds a new player to the team.

- bool **removePlayer** (const **Person** &player)
  Removes a player from the team.

- bool **operator==** (const **Team** &team)
- **Person** & **getPlayer** (size_t idx)
  Get the Player object with the given index.

- std::string **getName** () const
  Get the Name.

- size_t **size** () const
  Get the Size.

- virtual std::ostream & **print** (std::ostream &os)
  Writes the team to the ostream.

- virtual **~Team** ()
  *Destroy the **Team** object.*

Constructor & Destructor Documentation

Team::Team (std::string *name*) `[inline]`

Construct a new **Team** object.

Parameters

name	The name of the **Team**.

virtual Team::~Team () [inline], [virtual]

Destroy the **Team** object.

Member Function Documentation

bool Team::addPlayer (const Person & *player*)

Adds a new player to the team.

Parameters

player	

Returns

true If the player is added.

false If the player is already a team member.

std::string Team::getName () const

Get the Name.

Returns

The name of the team.

Person & Team::getPlayer (size_t *idx*)

Get the Player object with the given index.

Parameters

idx	the index of the needed player.

Returns

Person& A refenrece to the player with the given index.

bool Team::operator== (const Team & *team*)

Parameters

team	the **Team** that the caller wants to compair the object with.

Returns

true The two Teams are the same.

false The two Teams are different.

std::ostream & Team::print (std::ostream & *os*) [virtual]

Writes the team to the ostream.

Reimplemented in **FootballTeam** (*p.16*), **BasketballTeam** (*p.7*), and **HandballTeam** (*p.18*).

bool Team::removePlayer (const Person & *player*)

Removes a player from the team.

Parameters

player	

Returns

true If the player is removed.

false If the player is not a team member.

size_t Team::size () const

Get the Size.

Returns

The size of the team.

The documentation for this class was generated from the following files:

- headers/**teams.hpp**
- src/**teams.cpp**

File Documentation

headers/facility.hpp File Reference

Declaration of the class **Facility**.

```
#include <fstream>
#include "teams.hpp"
#include "json.hpp"
#include "memtrace.h"
```

Classes

class **Facility***This is a heterogeneous collection.*

Detailed Description

Declaration of the class **Facility**.

facility.hpp

Go to the documentation of this file.

```
7 #ifndef FACILITY_HPP
8 #define FACILITY_HPP
9
10 #include <fstream>
11 #include "teams.hpp"
12 #include "json.hpp"
13 #include "memtrace.h"
14
18 class Facility
19 {
20 private:
21     std::vector<Team*> teams;
22     size_t currentTeam;
23 public:
29     bool addTeam(Team* team);
35     bool removeTeam(Team team);
41     bool setCurrentTeam(const size_t& n);
45     size_t getCurrentTeam() const;
49     size_t size() const;
55     Team* operator[](size_t idx);
63     template<typename C>
64     C* getTeam(size_t idx)
65     {
66         size_t n = 0;
67         for (size_t i = 0; i < teams.size(); i++)
68         {
69             C* t;
70             if ((t = dynamic_cast<C*>(teams[i])))
71             {
72                 n++;
73             }
74             if (idx == n)
75             {
76                 return t;
77             }
78         }
79         return nullptr;
80     }
81
85     std::ostream& listFootballTeams(std::ostream& os);
89     std::ostream& listBasketballTeams(std::ostream& os);
93     std::ostream& listHandballTeams(std::ostream& os);
98     ~Facility()
99     {
100         for (size_t i = 0; i < teams.size(); i++)
101         {
102             delete teams[i];
103         }
104     }
105 };
106
107 #endif
```

headers/file.hpp File Reference

Declaration of the class **FileHandler**.

```
#include "facility.hpp"
#include "json.hpp"
#include "memtrace.h"
```

Classes

class **FileHandler**

Detailed Description

Declaration of the class **FileHandler**.

file.hpp

Go to the documentation of this file.

```
7 #ifndef FILE_HPP
8 #define FILE_HPP
9
10 #include "facility.hpp"
11 #include "json.hpp"
12 #include "memtrace.h"
13
14 class FileHandler
15 {
16 private:
17     std::string fPath;
18     std::string bPath;
19     std::string hPath;
20
21 public:
29     FileHandler(std::string fileF, std::string fileB, std::string fileH):
fPath(fileF), bPath(fileB) , hPath(fileH) {}
30
36     std::vector<FootballTeam*> readFootballTeams();
42     std::vector<BasketballTeam*> readBasketballTeams();
48     std::vector<HandballTeam*> readHandballTeams();
49
55     void writeFootballTeam(std::vector<FootballTeam> ft);
61     void writeBasketballTeam(std::vector<BasketballTeam> ft);
67     void writeHandballTeam(std::vector<HandballTeam> ft);
68
74     Facility readFacility();
80     void writeFacility(Facility changedFacility);
81 };
82
83
84 #endif
```

headers/menu.hpp File Reference

Declaration of the classes **Menu** and Menuhandler.

```
#include "facility.hpp"
#include "memtrace.h"
```

Classes

- class **Menu**class **MenuHandler**
  This is a heterogeneous collection.

Enumerations

- enum **State** { **Football**, **Basketball**, **Handball**, **Quit** }

Detailed Description

Declaration of the classes **Menu** and Menuhandler.

Enumeration Type Documentation

enum State

Enumerator:

Football	
Basketball	
Handball	
Quit	

menu.hpp

Go to the documentation of this file.

```
7 #ifndef MENU_HPP
8 #define MENU_HPP
9
10 #include "facility.hpp"
11 #include "memtrace.h"
12
13 enum State
14 {
15     Football,
16     Basketball,
17     Handball,
18     Quit
19 };
20
21 class Menu
22 {
23 private:
24     std::vector<std::string> buttons;
25     static State state;
26 public:
32     Menu(std::vector<std::string> buttons): buttons(buttons) {}
33
39     void setState(const State& n);
45     State getState() const;
46
50     std::ostream& print(std::ostream& os);
56     size_t select(std::istream& is);
57 };
58
62 class MenuHandler
63 {
64 private:
65     std::vector<Menu> menus;
66     size_t currentMenuIndex;
67 public:
73     MenuHandler(std::vector<Menu> menus): menus(menus), currentMenuIndex(0) {}
74
80     void addMenu(Menu newMenu);
87     void changeMenu(Menu oldMenu, Menu newMenu);
91     void toPrevMenu();
95     void toNextMenu();
101     size_t getIndex() const;
107     void setState(const State& n);
113     State getState() const;
114
119     Menu& operator[](size_t i);
120
124     std::ostream& print(std::ostream& os);
130     size t select(std::istream& is);
131 };
132
133 #endif
```

headers/person.hpp File Reference

Declaration of the class **Person**.

```
#include <iostream>
#include <vector>
#include "memtrace.h"
```

Classes

class **Person**

Detailed Description

Declaration of the class **Person**.

person.hpp

Go to the documentation of this file.

```
7 #ifndef PERSON_HPP
8 #define PERSON_HPP
9
10 #include <iostream>
11 #include <vector>
12 #include "memtrace.h"
13
14 class Person
15 {
16 private:
17     std::string name;
18     size_t age;
19 public:
26     Person(std::string name, size_t age): name(name), age(age) {}
27
33     std::string getName() const;
39     size_t getAge() const;
40
48     bool operator==(const Person& person) const;
56     bool operator!=(const Person& person) const;
57 };
58
59 #endif
```

headers/teams.hpp File Reference

Declaration of the classes **Team** and the three different teams.

```
#include "person.hpp"
#include "memtrace.h"
```

Classes

- class **Team**class **FootballTeam**
- class **BasketballTeam**
- class **HandballTeam**

Detailed Description

Declaration of the classes **Team** and the three different teams.

teams.hpp

Go to the documentation of this file.1

```
7 #ifndef TEAMS_HPP
8 #define TEAMS_HPP
9
10 #include "person.hpp"
11 #include "memtrace.h"
12
13 class Team
14 {
15 private:
16     std::string name;
17     std::vector<Person> players;
18 public:
24     Team(std::string name): name(name) {}
25
33     bool addPlayer(const Person& player);
41     bool removePlayer(const Person& player);
49     bool operator==(const Team& team);
56     Person& getPlayer(size_t idx);
62     std::string getName() const;
68     size_t size() const;
69
73     virtual std::ostream& print(std::ostream& os);
74
79     virtual ~Team() {}
80 };
81
82 class FootballTeam : public Team
83 {
84 private:
85     std::vector<Person> coaches;
86 public:
92     FootballTeam(std::string name): Team(name) {}
93
101     bool addCoach(const Person& coach);
109     bool removeCoach(const Person& coach);
116     Person& getCoach(size_t idx);
122     size t numberOfCoaches() const;
123
127     std::ostream& print(std::ostream& os);
128
133     ~FootballTeam() {}
134 };
135
136 class BasketballTeam : public Team
137 {
138 private:
139     std::vector<Person> cheerleaders;
140 public:
146     BasketballTeam(std::string name): Team(name) {}
147
155     bool addCheerleader(const Person& cheerleader);
163     bool removeCheerleader(const Person& cheerleader);
170     Person& getCheerleader(size_t idx);
176     size t numberOfCheerleaders();
177
181     std::ostream& print(std::ostream& os);
182
187     ~BasketballTeam() {}
188 };
189
190 class HandballTeam : public Team
191 {
192 private:
193     size_t supportMoney;
194 public:
201     HandballTeam(std::string name, size t value = 0): Team(name), supportMoney(value)
{}
202
208     void changeSupportMoney(size_t newValue);
214     size_t getValue() const;
215
219     std::ostream& print(std::ostream& os);
```

```
220
225     ~HandballTeam() {}
226 };
227
228 #endif
```

main.cpp File Reference

Creating the **Menu** and handling it with a working State Machine.

```
#include "headers/menu.hpp"
#include "headers/file.hpp"
#include "headers/memtrace.h"
```

Functions

- int **main** ()

Detailed Description

Creating the **Menu** and handling it with a working State Machine.

Function Documentation

int main ()

src/facility.cpp File Reference

Implementation of the class **Facility**.

```
#include "../headers/facility.hpp"
```

Detailed Description

Implementation of the class **Facility**.

src/file.cpp File Reference

Implementation of the class **FileHandler**.
`#include "../headers/file.hpp"`

Typedefs

- using **json** = nlohmann::json

Detailed Description

Implementation of the class **FileHandler**.

Typedef Documentation

using json = nlohmann::json

src/menu.cpp File Reference

Implementation of the classes **Menu** and **MenuHandler**.

```
#include "../headers/menu.hpp"
```

Detailed Description

Implementation of the classes **Menu** and **MenuHandler**.

src/person.cpp File Reference

Implementation of the class **Person**.

```
#include "../headers/person.hpp"
```

Detailed Description

Implementation of the class **Person**.

src/teams.cpp File Reference

Implementation of the classes **Team** and the three different teams.
`#include "../headers/teams.hpp"`

Detailed Description

Implementation of the classes **Team** and the three different teams.

Specifikáció

Házi feladat – Sportegyesület

Programozás alapjai 2.

A feladatban megvalósítok egy sportegyesületet. A program segítségével 3 különböző sportágból lehet csapatokat nyilvántartani.
Egy menürendszer segítségével lehet a konzolban navigálni, ezt egy állapotgéppel valósítom meg
A menürendszer a következőképpen fog kinézni:
A program elindításakor:
1) Focicsapatok
2) Kosárcsapatok
3) Kézicsapatok
4) Kilépés (ez a menüpont mindig elérhető lesz)

Ezek közül bármelyiket kiválasztva (a kilépést kivéve) a következők jelennek meg:
1) Csapatok listázása
2) Új csapat felvétele
3) Csapat törlése
4) Vissza (az első menüsor kivételével ez a gomb szintén mindig elérhető lesz)

Csapat törlése és felvétele esetén meg kell adni a csapat nevét, (az egyesületen belül minden csapat különböző nevű kell, hogy legyen) ha ezt megtettük, akkor automatikusan visszatérünk ehhez a menühöz.
Vissza gombnál pedig vissza tudunk menni eggyel a menüben.
Csapatok listázásánál a csapatok megjelennek egyesével felsorolva. Először a nevük, majd a játékosok és legutoljára, focicsapatnál az edző(k), kosárcsapatnál a pom-pom lányok, kézicsapatnál az évi támogatás.
Ha kilistáztuk a csapatokat, akkor ki tudunk választani egyet, majd a következők jelennek meg:
1) Játékos felvétele
2) Játékos törlése

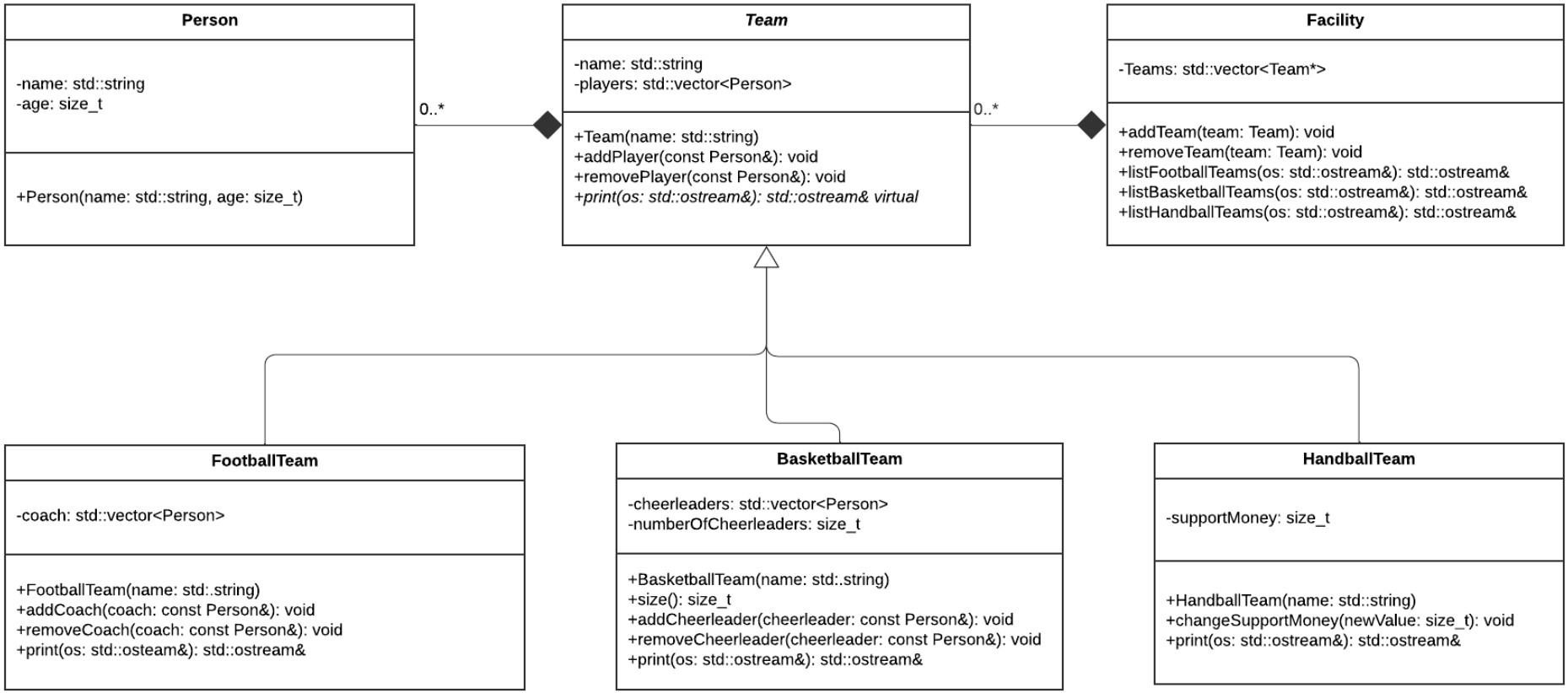
Valamint
Focicsapatnál: Edző felvétele, törlése
Kosárcsapatnál: Pom-pom lány felvétele, törlése.
Kézicsapatnál: Támogatás összeg változtatása.

Itt bármelyik változtatás után visszatérünk ehhez a menüsorhoz.

A játékosoknak lesz neve és életkora, ezeket a felvételkor kell majd megadni.

A programomban lesz egy csapat ősosztály, ennek a leszármazottjai lesznek a különféle csapatok. Minden csapatnak lesznek játékosai és egy egyedi neve. A játékosok emberek lesznek, amit szintén egy osztályként valósítok meg. Magát az egyesület egy heterogén kollekció lesz, ami csapat pointereket tárol.
A programot angol nyelven írom meg. A csapatokat egy fájlból olvasom be, majd a program végén a változtatott egyesületet ugyanebbe a fájlba írom vissza.

Sport Facility



Tesztelés

A programhoz tartozik 3 json fájl, amit a tesztelésben segít. Ezek a fájlok tartalmazzák a Teszt csapatokat, valamit ezekbe a fájlokba történik a fájlkezelés is.

Kezdetben a program a fájlokból kiolvassa az adatokat és létrehoz ezekből az adatokból egy Egyesületet. A program futása közben ezt az egyesületet kedvünk szerint változtathatjuk, ha már nem kívánunk semmin sem változtatni, akkor kilépünk a programból, ekkor a program a megváltoztatott Egyesületet egyesével visszaírja a megfelelő fájlba.

Memóriaszivárgás

A programban egyedül az egyesületben foglalunk memóriát az új csapatoknak. A csapat, valamit az egyesület törlésénél is kezelem ezt.