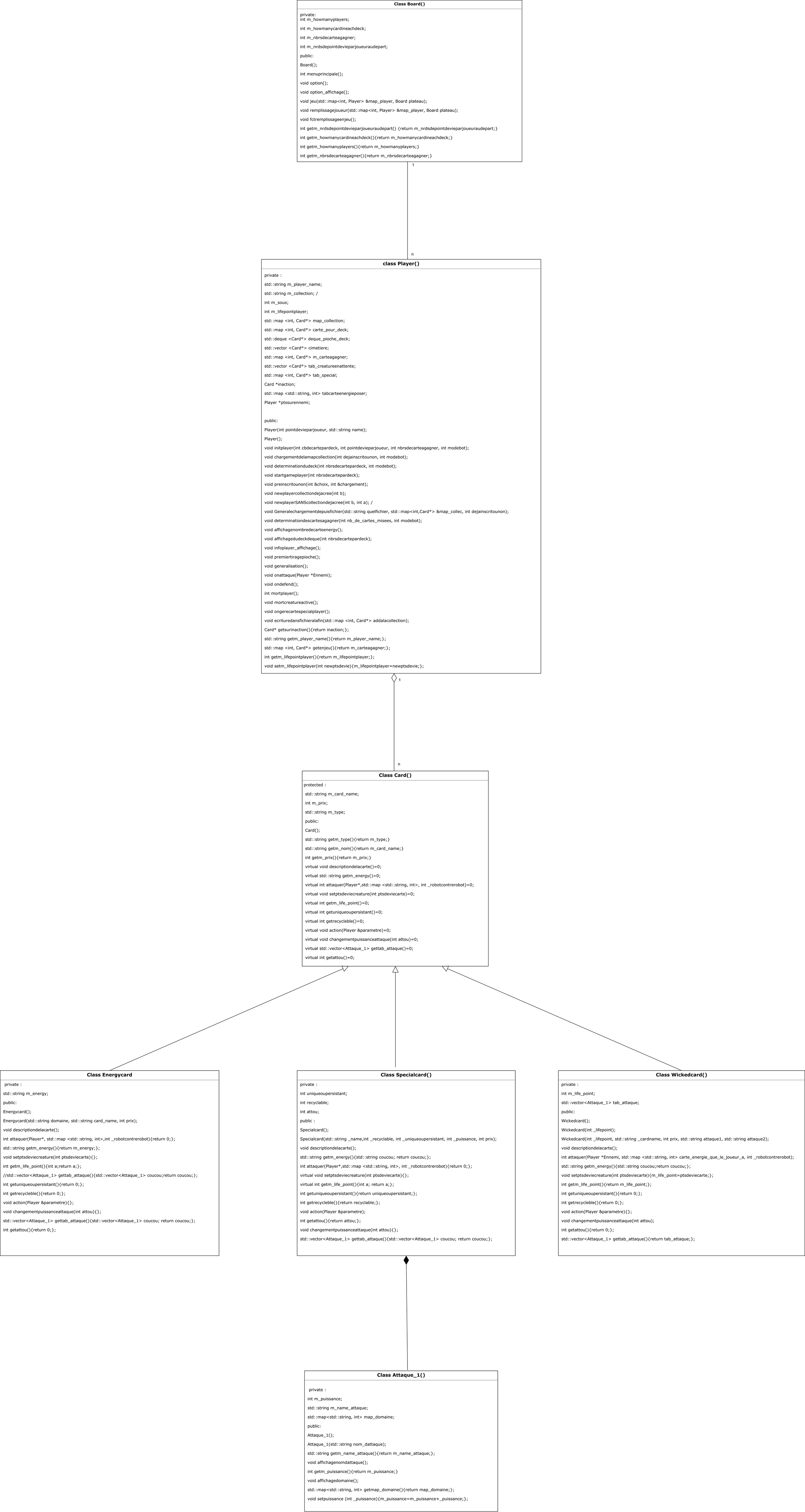
Class Board()
private:
int m_howmanyplayers;
int m_howmanycardineachdeck;
int m_nbrsdecarteagagner;
int m_nrdsdepointdevieparjoueuraudepart;
public:
Board();
int menuprincipale();
void option();
void option_affichage();
void jeu(std::map<int, Player> &map_player, Board plateau);
void remplissagejoueur(std::map<int, Player> &map_player, Board plateau);
void fctremplissageenjeu();
int getm_nrdsdepointdevieparjoueuraudepart() {return m_nrdsdepointdevieparjoueuraudepart;}
int getm_howmanycardineachdeck(){return m_howmanycardineachdeck;}
int getm_howmanyplayers(){return m_howmanyplayers;}
int getm_nbrsdecarteagagner(){return m_nbrsdecarteagagner;}
1
n
class Player()
private :
std::string m_player_name;
std::string m_collection; /
int m_sous;
int m_lifepointplayer;
std::map <int, Card*> map_collection;
std::map <int, Card*> carte_pour_deck;
std::deque <Card*> deque_pioche_deck;
std::vector <Card*> cimetiere;
std::map <int, Card*> m_carteagagner;
std::vector <Card*> tab_creatureenattente;
std::map <int, Card*> tab_special;
Card *inaction;
std::map <std::string, int> tabcarteenergieposer;
Player *ptssurennemi;
public:
Player(int pointdevieparjoueur, std::string name);
Player();
void initplayer(int cbdecartepardeck, int pointdevieparjoueur, int nbrsdecarteagagner, int modebot);
void chargementdelamapcollection(int dejainscritounon, int modebot);
void determinationdudeck(int nbrsdecartepardeck, int modebot);
void startgameplayer(int nbrsdecartepardeck);
void preinscritounon(int &choix, int &chargement);
void newplayercollectiondejacree(int b);
void newplayerSANScollectiondejacree(int b, int a); /
void Generalechargementdepuisfichier(std::string quelfichier, std::map<int,Card*> &map_collec, int dejainscritounon);
void determinationdescartesagagner(int nb_de_cartes_misees, int modebot);
void affichagenombredecarteenergy();
void affichagedudeckdeque(int nbrsdecartepardeck);
void infoplayer_affichage();
void premiertiragepioche();
void generalisation();
void onattaque(Player *Ennemi);
void ondefend();
int mortplayer();
void mortcreatureactive();
void ongerecartespecialplayer();
void ecrituredansfichieralafin(std::map <int, Card*> addalacollection);
Card* getsurinaction(){return inaction;};
std::string getm_player_name(){return m_player_name;};
std::map <int, Card*> getenjeu(){return m_carteagagner;};
int getm_lifepointplayer(){return m_lifepointplayer;};
void setm_lifepointplayer(int newptsdevie){m_lifepointplayer=newptsdevie;};
1
n
Class Card()
protected :
std::string m_card_name;
int m_prix;
std::string m_type;
public:
Card();
std::string getm_type(){return m_type;}
std::string getm_nom(){return m_card_name;}
int getm_prix(){return m_prix;}
virtual void descriptiondelacarte()=0;
virtual std::string getm_energy()=0;
virtual int attaquer(Player*,std::map <std::string, int>, int _robotcontrerobot)=0;
virtual void setptsdeviecreature(int ptsdeviecarte)=0;
virtual int getm_life_point()=0;
virtual int getuniqueoupersistant()=0;
virtual int getrecycleble()=0;
virtual void action(Player &parametre)=0;
virtual void changementpuissanceattaque(int attou)=0;
virtual std::vector<Attaque_1> gettab_attaque()=0;
virtual int getattou()=0;
Class Energycard
private :
std::string m_energy;
public:
Energycard();
Energycard(std::string domaine, std::string card_name, int prix);
void descriptiondelacarte();
int attaquer(Player*, std::map <std::string, int>,int _robotcontrerobot){return 0;};
std::string getm_energy(){return m_energy;};
void setptsdeviecreature(int ptsdeviecarte){};
int getm_life_point(){int a;return a;};
//std::vector<Attaque_1> gettab_attaque(){std::vector<Attaque_1> coucou;return coucou;};
int getuniqueoupersistant(){return 0;};
int getrecycleble(){return 0;};
void action(Player &parametre){};
void changementpuissanceattaque(int attou){};
std::vector<Attaque_1> gettab_attaque(){std::vector<Attaque_1> coucou; return coucou;};
int getattou(){return 0;};
Class Specialcard()
private :
int uniqueoupersistant;
int recyclable;
int attou;
public :
Specialcard();
Specialcard(std::string _name,int _recyclable, int _uniqueoupersistant, int _puissance, int prix);
void descriptiondelacarte();
std::string getm_energy(){std::string coucou; return coucou;};
int attaquer(Player*,std::map <std::string, int>, int _robotcontrerobot){return 0;};
virtual void setptsdeviecreature(int ptsdeviecarte){};
virtual int getm_life_point(){int a; return a;};
int getuniqueoupersistant(){return uniqueoupersistant;};
int getrecycleble(){return recyclable;};
void action(Player &parametre);
int getattou(){return attou;};
void changementpuissanceattaque(int attou){};
std::vector<Attaque_1> gettab_attaque(){std::vector<Attaque_1> coucou; return coucou;};
Class Wickedcard()
private :
int m_life_point;
std::vector<Attaque_1> tab_attaque;
public:
Wickedcard();
Wickedcard(int _lifepoint);
Wickedcard(int _lifepoint, std::string _cardname, int prix, std::string attaque1, std::string attaque2);
void descriptiondelacarte();
int attaquer(Player *Ennemi, std::map <std::string, int> carte_energie_que_le_joueur_a, int _robotcontrerobot);
std::string getm_energy(){std::string coucou;return coucou;};
void setptsdeviecreature(int ptsdeviecarte){m_life_point=ptsdeviecarte;};
int getm_life_point(){return m_life_point;};
int getuniqueoupersistant(){return 0;};
int getrecycleble(){return 0;};
void action(Player &parametre){};
void changementpuissanceattaque(int attou);
int getattou(){return 0;};
std::vector<Attaque_1> gettab_attaque(){return tab_attaque;};
Class Attaque_1()
private :
int m_puissance;
std::string m_name_attaque;
std::map<std::string, int> map_domaine;
public:
Attaque_1();
Attaque_1(std::string nom_dattaque);
std::string getm_name_attaque(){return m_name_attaque;};
void affichagenomdattaque();
int getm_puissance(){return m_puissance;}
void affichagedomaine();
std::map<std::string, int> getmap_domaine(){return map_domaine;};
void setpuissance (int _puissance){m_puissance=m_puissance+_puissance;};