GOLFIER
MAXIME
GOLM16039801

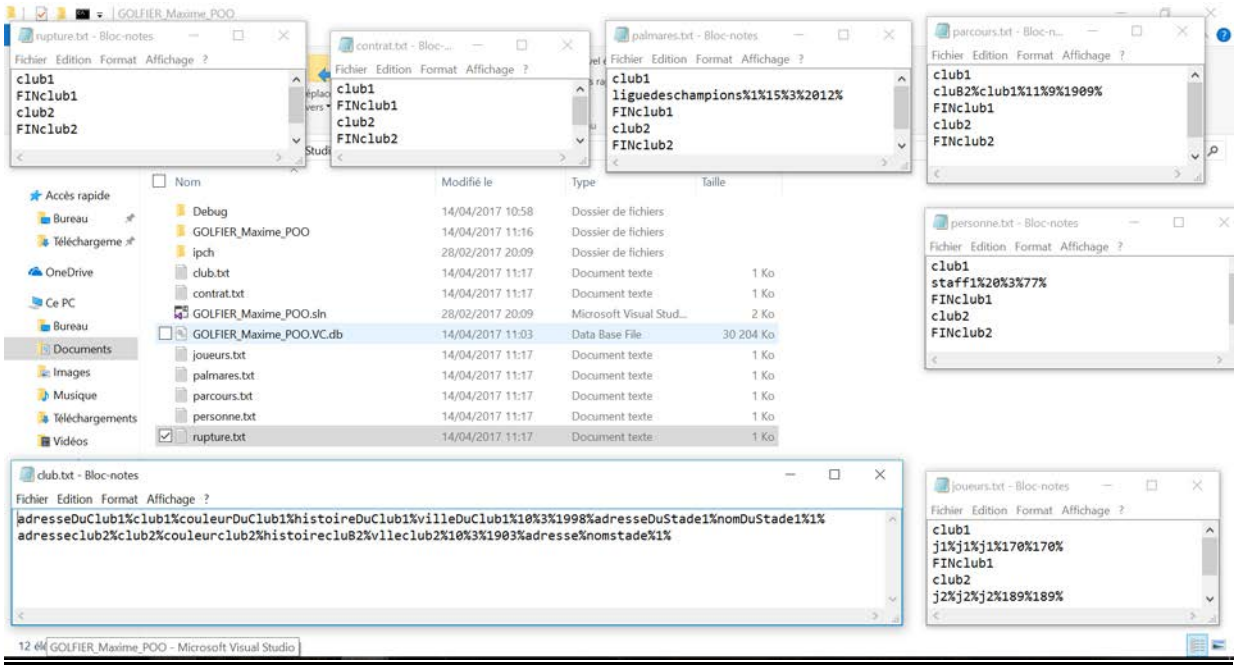
MINI PROJET- Programmation Orientée Objet

Projet de Mr Abdenour Bouzouane pour la Session d'hiver de 2017 à UQAC.

1 - Ouverture du programme

```
---Menu principal:----
        A-Menu du TP1
        B-Menu du TP2
        C-Creation d'une negociation
        D-Charger
        E-Sauvegarder
        X-Quitter
                Votre choix:
```

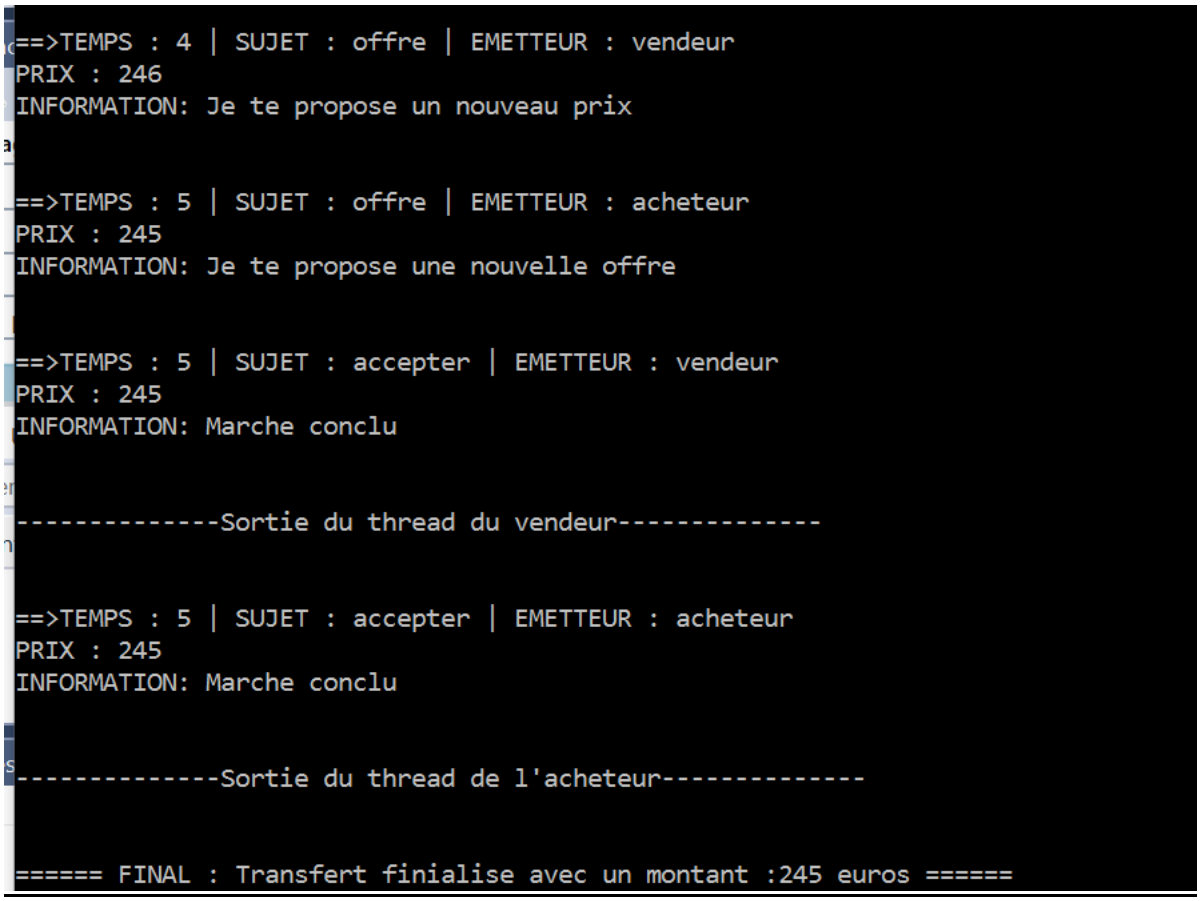
1.5-Initialisation : Fonction de Chargement/Sauvegarde



PS : Voici les fichiers qui sont générées suite à la sauvegarde. La fonction de chargement utilisera cela pour initialiser le programme. La fonction sauvegarde est la réciproque de la fonction chargement.

2-Création d'une transaction

```
        X-Quitter
                Votre choix:C
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
200

---ACHETEUR---
Donner le montant desire pour l'acheteur
240
Ainsi que le montant maximum qu'il peut faire
290
Donner le nom du club
club1
Element trouve dans mon vecteur:
Le club : club1 existe

---VENDEUR---
Donner le montant desire pour le vendeur
250
Ainsi que le montant maximum qu'il peut faire
100
Donner le nom du club
club2
Element trouve dans mon vecteur:
Le club : club2 existe


--------------Entree dans le thread de l'acheteur--------------

--------------Entree dans le thread du vendeur--------------

==>TEMPS : 0 | SUJET : offre | EMETTEUR : acheteur
PRIX : 240
INFORMATION: Je te propose une nouvelle offre


==>TEMPS : 0 | SUJET : offre | EMETTEUR : vendeur
PRIX : 250
INFORMATION: Je te propose un nouveau prix
```



PS : Suite à la réussite de la transaction, l'utilisateur est invité à créer un contrat grâce à la fonction que nous avons codé au TP2.

2.1 – Echec dû au temps qui est trop court….

```
votre choix:c
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
5

---ACHETEUR---
Donner le montant desire pour l'acheteur
200
Ainsi que le montant maximum qu'il peut faire
2000
Donner le nom du club
club1
Element trouve dans mon vecteur:
Le club : club1 existe

---VENDEUR---
Donner le montant desire pour le vendeur
1600
Ainsi que le montant maximum qu'il peut faire
300
Donner le nom du club
club2
Element trouve dans mon vecteur:
Le club : club2 existe


--------------Entree dans le thread de l'acheteur--------------

--------------Entree dans le thread du vendeur--------------

==>TEMPS : 0 | SUJET : offre | EMETTEUR : acheteur
PRIX : 200
INFORMATION: Je te propose une nouvelle offre


==>TEMPS : 0 | SUJET : offre | EMETTEUR : vendeur
PRIX : 1600
INFORMATION: Je te propose un nouveau prix
```

```
==>TEMPS : 4 | SUJET : offre | EMETTEUR : vendeur
PRIX : 1596
INFORMATION: Je te propose un nouveau prix


==>TEMPS : 4 | SUJET : offre | EMETTEUR : acheteur
PRIX : 204
INFORMATION: Je te propose une nouvelle offre


--------------Sortie du thread de l'acheteur--------------

==>TEMPS : 5 | SUJET : offre | EMETTEUR : vendeur
PRIX : 1595
INFORMATION: Je te propose un nouveau prix



--------------Sortie du thread du vendeur--------------


====== FINAL : Le transfert fut un echec... ======
```

2.2 – Echec dû à un prix trop bas de l'acheteur…

```
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
2000

---ACHETEUR---
Donner le montant desire pour l'acheteur
2000
Ainsi que le montant maximum qu'il peut faire
2020
Donner le nom du club
club1
Element trouve dans mon vecteur:
Le club : club1 existe

---VENDEUR---
Donner le montant desire pour le vendeur
2100
Ainsi que le montant maximum qu'il peut faire
2040
Donner le nom du club
club2
```

```
==>TEMPS : 20 | SUJET : offre | EMETTEUR : acheteur
PRIX : 2020
INFORMATION: Je te propose une nouvelle offre


==>TEMPS : 20 | SUJET : offre | EMETTEUR : vendeur
PRIX : 2080
INFORMATION: Je te propose un nouveau prix


==>TEMPS : 21 | SUJET : refuser | EMETTEUR : acheteur
PRIX : 2021
INFORMATION: Je peux pas aller plus haut....


--------------Sortie du thread de l'acheteur--------------


==>TEMPS : 21 | SUJET : refuser | EMETTEUR : vendeur
PRIX : 2079
INFORMATION: Bah si t'as pas assez de sous, tu peux pas l'acheter mon pote !


--------------Sortie du thread du vendeur--------------


====== FINAL : Le transfert fut un echec... ======
```

2.3 – Echec dû à un prix trop haut du vendeur…

```
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
2000

---ACHETEUR---
Donner le montant desire pour l'acheteur
2000
Ainsi que le montant maximum qu'il peut faire
30000
Donner le nom du club
club1
Element trouve dans mon vecteur:
Le club : club1 existe

---VENDEUR---
Donner le montant desire pour le vendeur
40000
Ainsi que le montant maximum qu'il peut faire
39850
Donner le nom du club
club2
```

```
==>TEMPS : 149 | SUJET : offre | EMETTEUR : acheteur
PRIX : 2149
INFORMATION: Je te propose une nouvelle offre


==>TEMPS : 149 | SUJET : offre | EMETTEUR : vendeur
PRIX : 39851
INFORMATION: Je te propose un nouveau prix


==>TEMPS : 149 | SUJET : offre | EMETTEUR : acheteur
PRIX : 2149
INFORMATION: Je te propose une nouvelle offre


==>TEMPS : 150 | SUJET : refuser | EMETTEUR : vendeur
PRIX : 39850
INFORMATION: Je peux pas descendre plus bas...


--------------Sortie du thread du vendeur--------------


==>TEMPS : 150 | SUJET : refuser | EMETTEUR : acheteur
PRIX : 2150
INFORMATION: Tu demandes trop de sous la ! Je suis desole mais le transfert est fini !


--------------Sortie du thread de l'acheteur--------------


====== FINAL : Le transfert fut un echec... ======
```

3 – Principe de Polymorphisme : Méthode virtuelle pure

```
#include "Sportif.h"
#include "Contrat.h"

class JoueurContract : public virtual Sportif
{
protected:
    Contrat *contratActuel;

public:
    JoueurContract();
    JoueurContract(Contrat*, string, string);
    virtual ~JoueurContract();

#pragma region Getter
    Contrat* getContrat();
#pragma endregion Getter

#pragma region Setter
    void setContrat(Contrat*);
#pragma endregion Setter

#pragma region fonction virtuelle
    virtual void indiquerSonEtat() = 0;
#pragma endregion fonction virtuelle
};
#endif
```

```
#ifndef HETERONOME_H
#define HETERONOME_H

#include "Joueur.h"
#include "JoueurContract.h"

class Heteronome : public JoueurContract, public Joueur
{

private:
    int nbAnneeRestante;

public:
#pragma region Constructeur
    Heteronome();
    Heteronome(string, string, float, float, string, vector<Parcours*>, int, Contrat*);
    ~Heteronome();
#pragma endregion Constructeur

#pragma region Getter
    int getNbAnRest();
#pragma endregion Getter

#pragma region Setter
    void setNbAnRest(int);
#pragma endregion Setter

    void indiquerSonEtat();
};
#endif
```

Heteronome.cpp | Heteronome.h | JoueurContract.h | NegoAcheteur.cpp | Nego

TP_GOLFIER_Maxime | (Portée globale)

```cpp
#pragma region Getter
int Heteronome::getNbAnRest()
{
    return nbAnneeRestante;
}
#pragma endregion Getter

#pragma region Setter
void Heteronome::setNbAnRest(int n)
{
    nbAnneeRestante = n;
}
#pragma endregion Setter

void Heteronome::indiquerSonEtat()
{
    cout << "Je suis un joueur heteronome et il me reste " << getNbAnRest() << " annee restante...\n";
}
```

Autonome.cpp | Heteronome.cpp | Heteronome.h | JoueurC

TP_GOLFIER_Maxime | (Portée globale)

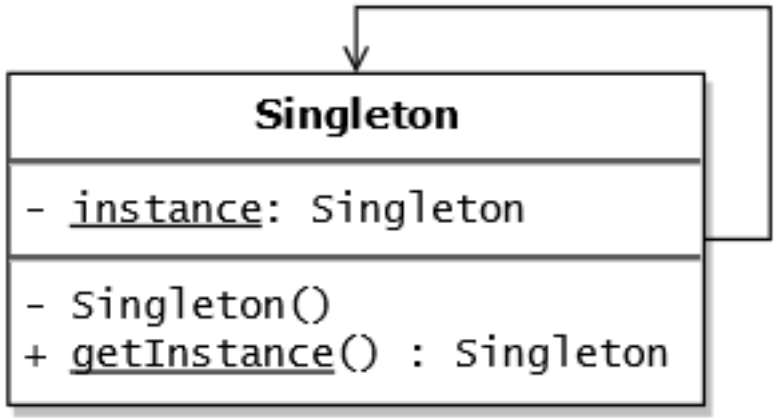
```cpp
}
#pragma endregion Constructeur

#pragma region Getter
Rupture * Autonome::getRupture()
{
    return rupture;
}
#pragma endregion Getter

#pragma region Setter
void Autonome::setRupture(Rupture *r)
{
    rupture = r;
}
#pragma endregion Setter

void Autonome::indiquerSonEtat()
{
    cout << "Je suis un joueur autonome ;) \n";
}
```

PS : Ici, nous voyons que la même méthode est définie de deux manière différentes en fonction de la classe fille, cela s'explique parce que la méthode a été définie en virtuelle dans la classe mère

4– Principe de Généricité : Template class

```cpp
template <class T>
class Singleton
{
public:
    template <typename... Args>
    static
        T* get_instance(Args... args)
    {
        if (!instance_)
        {
            instance_ = new T(std::forward<Args>(args)...);
        }

        return instance_;
    }

    static
        void destroy_instance()
    {
        delete instance_;
        instance_ = nullptr;
    }

private:
    static T* instance_;
};

template <class T> T*  Singleton<T>::instance_ = nullptr;
#endif
```

PS J'ai créé une template class singleton, ainsi cela m'évite de la redéfinir plusieurs fois si je l'utilise à différents endroits dans mon code. Pour rappel, le singleton permet d'éviter la prolifération d'instance afin d'améliorer la performance du programme.



Concrètement, cela m'a permis de l'utiliser dans deux classes assez « lourde », à savoir la classe Chargement et Sauvegarde. Voici la manière dont on doit l'implémenter :

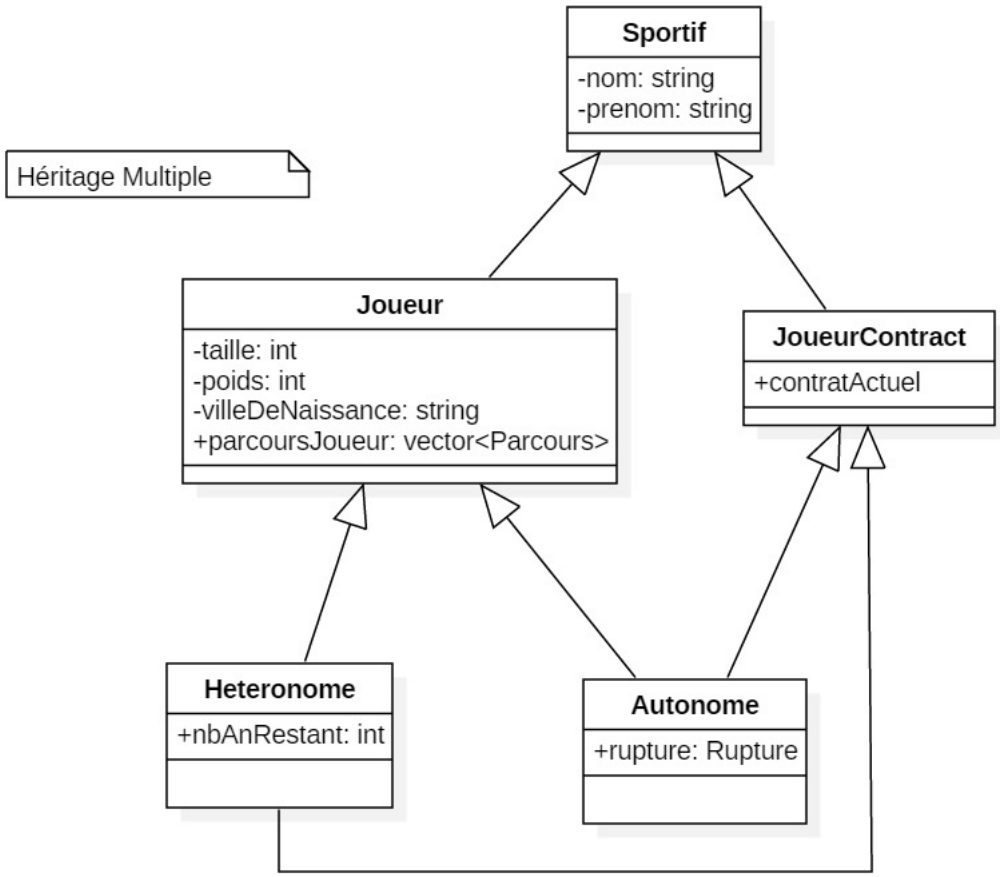
```
class Chargement : public Singleton<Chargement>
{

    friend class Singleton<Chargement>;

private:
    Chargement() {}

public:

    vector<Joueur*> ChargerJoueur(string);

    vector<Parcours*> ChargerParcours(string);

    vector<Personne*> ChargerPersonne(string);
```

```
class Sauvegarde : public Singleton<Sauvegarde>
{

    friend class Singleton<Sauvegarde>;

private:
    Sauvegarde() {}

public:
    void EnregistrerJoueur(vector<Joueur*>,string);
```

Puis lorsqu'on utilise ce pattern, on agit de cette manière :

```
void Affichage::CreerNegociation(LigueSoccer * lg, Calendrier *ca
#pragma endregion TP3

#pragma region Chargement + Sauvegarde
void Affichage::chargeAll(LigueSoccer *lg, Calendrier *cal)
{
    //singleton
    Chargement* c = Chargement::get_instance();
    lg->setListeClub(c->ChargerClub());
    Chargement::destroy_instance();
}

void Affichage::sauvAll(LigueSoccer *lg, Calendrier *cal)
{
    //singleton
    Sauvegarde* s = Sauvegarde::get_instance();
    s->EnregistrerClub(lg->getListeClub());
    Sauvegarde::destroy_instance();
}
#pragma endregion Chargement + Sauvegarde
```

5- Principe de réutilisation : Le double héritage



```
#include "JoueurContract.h"
#include "Rupture.h"

class Autonome :  public JoueurContract, public Joueur
{

private:
    Rupture *rupture;

public:

#pragma region Constructeur
    Autonome();
    Autonome(string, string, float, float, string, vector<Parcours*>, Rupture*, Contrat*);
    ~Autonome();
#pragma endregion Constructeur

#pragma region Getter
    Rupture* getRupture();
```

PS : Reportez-vous au point 3 pour voir la classe JoueurContract et Heteronome. Afin d'éviter la duplication de membre de la classe mère, j'ai fait de l'héritage virtuelle dans la classe Joueur et JoueurContract.

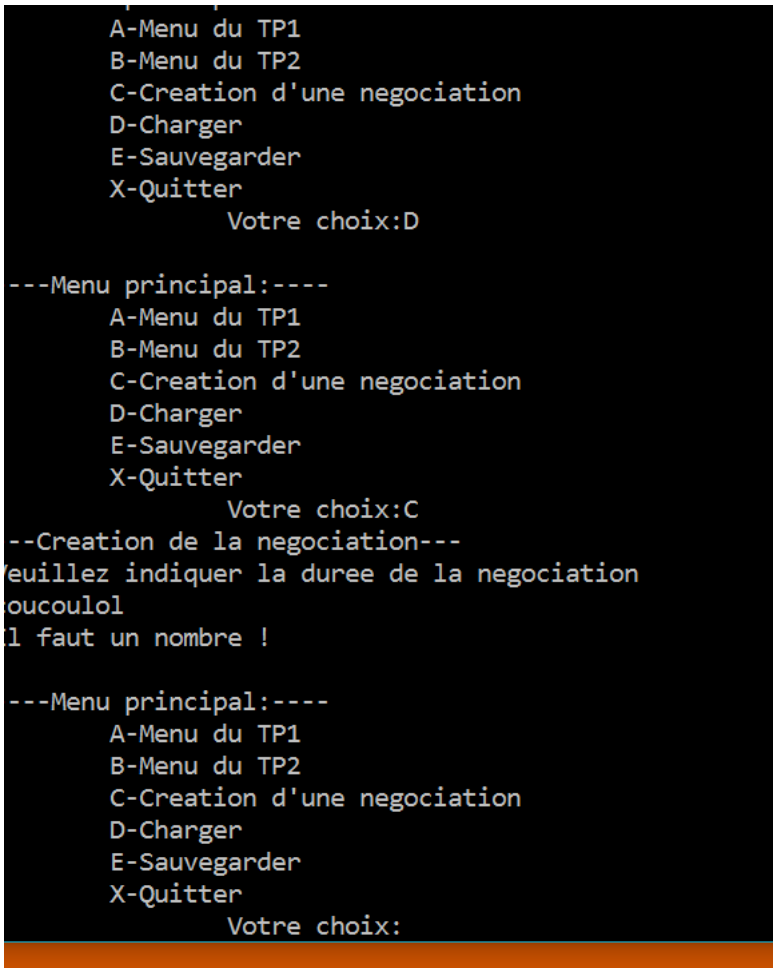
6- Principe de robustesse : La gestion des exceptions

1.Mon exception :

```
        B-Menu du TP2
        C-Creation d'une negociation
        D-Charger
        E-Sauvegarder
        X-Quitter
                Votre choix:C
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
1000000
Exception : La duree est trop longue...

----Menu principal:----
        A-Menu du TP1
        B-Menu du TP2
        C-Creation d'une negociation
        D-Charger
        E-Sauvegarder
        X-Quitter
                Votre choix:C
---Creation de la negociation---
Veuillez indiquer la duree de la negociation
200

---ACHETEUR---
Donner le montant desire pour l'acheteur
```

```cpp
#ifndef EXCEPTION_CHIFFRE_H
#define EXCEPTION_CHIFFRE_H

#include <iostream>
#include <exception>

using namespace std;


class Exception_Chiffre : public exception {
private:
    char * ad_texte;
public:
    Exception_Chiffre(char * texte) { ad_texte = texte; }
    const char * what() const { return ad_texte; }

};
#endif
```

PS : Ici, j'ai créé ma propre exception que j'ai mise dans le package exception. Elle me permet de gérer les paramètres de l'application du genre : durée d'une transaction.



PS : Ici j'utilise les excpetions du c++ pour gérer ma saisie. Un utilisateur est obligé de saisir un nombre sinon une exception est déclenchée. Très utile pour éviter que le programme plante !

```
cin.exceptions(istream::failbit | istream::badbit);

try
{
    cin >> duree;
}
catch (istream::failure e)
{
    cerr << "Il faut un nombre !\n";
    cin.clear();
    cin.ignore((numeric_limits<streamsize>::max)(), '\n');
    return;

}

try {
    if (duree>10000) {
        throw Exception_Chiffre("La duree est trop longue...");
    }
}
catch (exception & e) {
    cout << "Exception : " << e.what() << "\n";
    return;
```

PS : Voici dans le main, la manière dans je try et catch mes exceptions que ce soit celle du système (que je n'ai pas codé) et les miennes.