Objectif Mars

Ou comment les compagnies aériennes du futur, financent les vols spatiaux

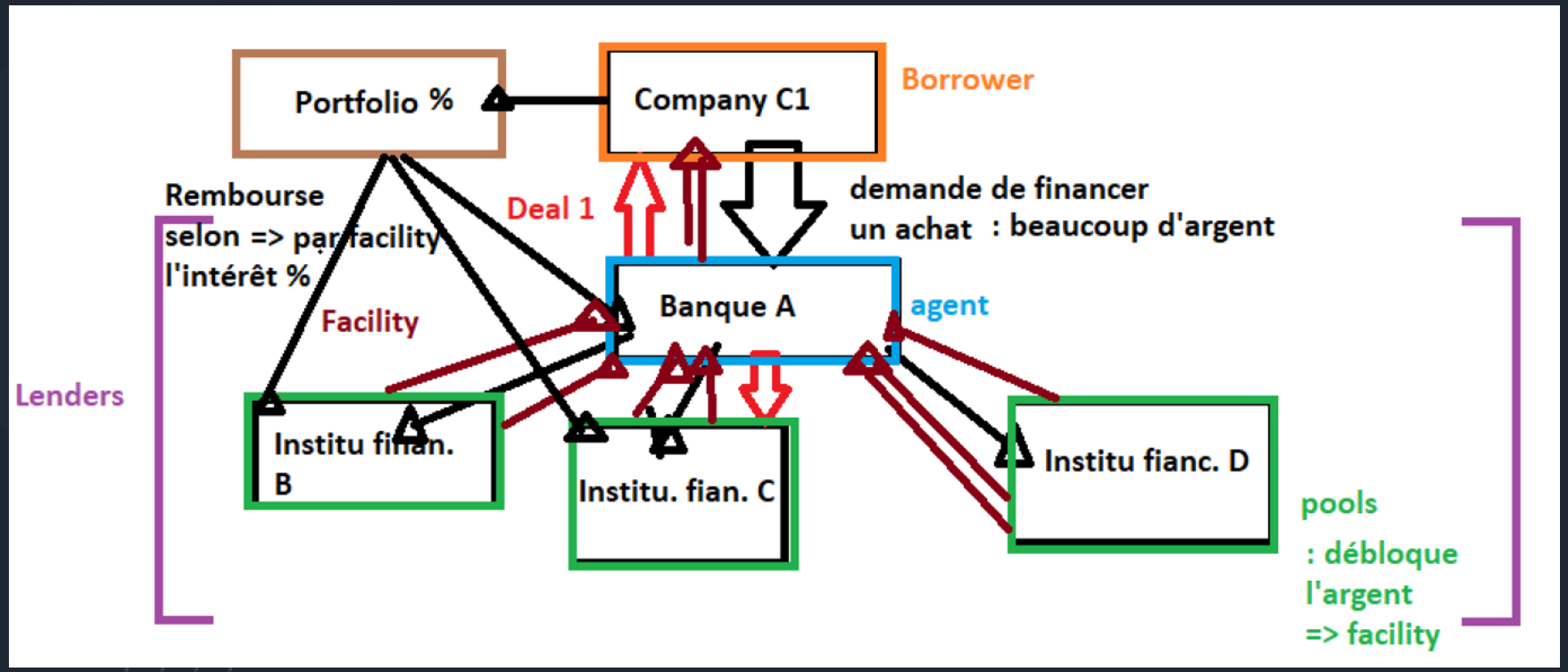
Alizé Baudin

Schéma du projet

Gestion du polymorphisme :

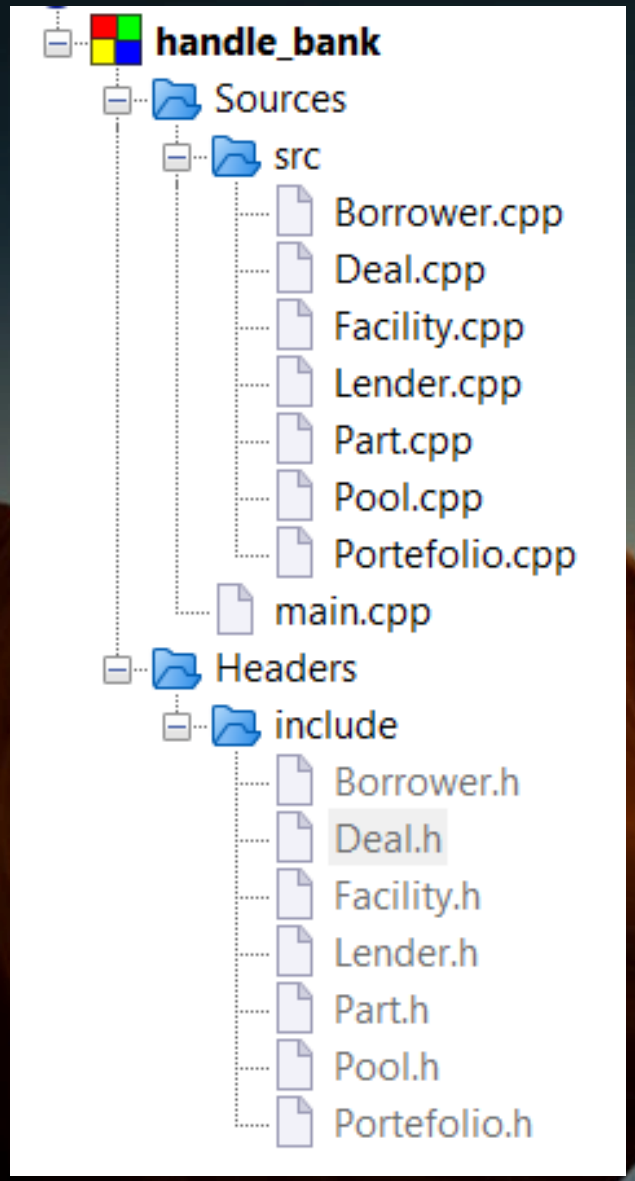
- Qui est l'héritié de qui ?
- Qui pointe vers qui ?

---> définition des besoins et des possibles



Company C1 : Ariane Space
Banque A : Internationale Space Bank
Institu finan. B : European Space Bank (ESA-Bank)
Institu financ. C : Musk Action
Institu Financ. D : Soviet Space Bank (Space Bank Russe)
Etc...

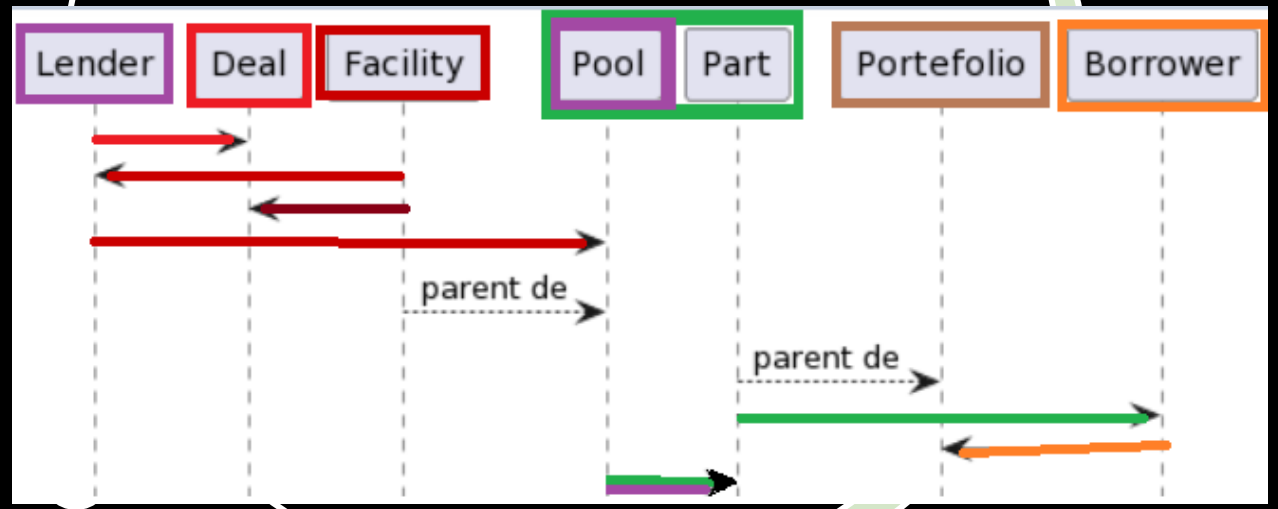
Définition des classes



- Afin de laisser transparaître le rôle de chaque classe, il fut utile de les classer une à une pour s'y retrouver.

- Au besoin nous utiliserons la ligne #include"nom_classe.h" dans le header de la classe.

Schéma définitif



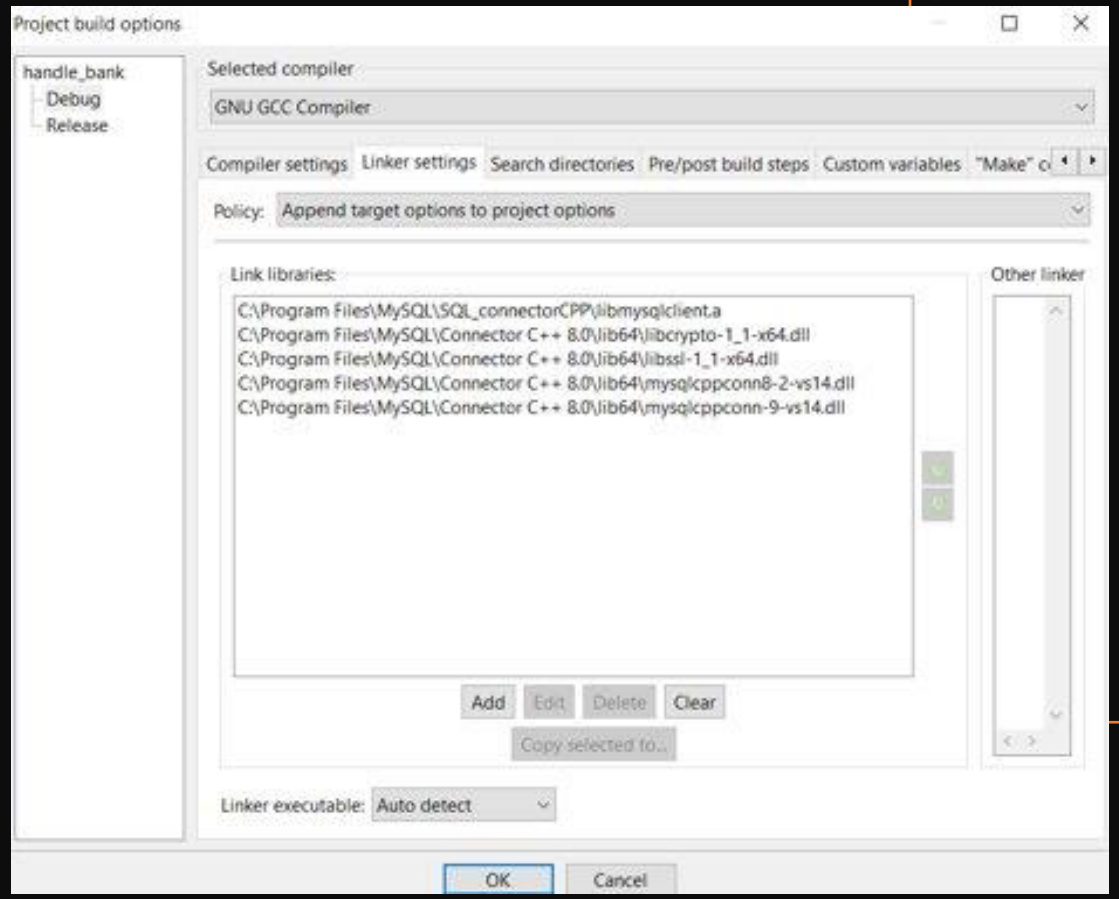
- Lender pointe sur Deal et Pool
- Deal est une classe autonome vu qu'elle ne pointe sur aucune autre classe.
- Facility pointe sur Lender et Deal . Facility est parent de la classe Pool
- Borrower pointe sur : Portfolio
- Pool : enfant de Facility, pointe sur Part
- Part : parent de Portfolio, pointe sur Borrower
- Portefolio : enfant de Part,

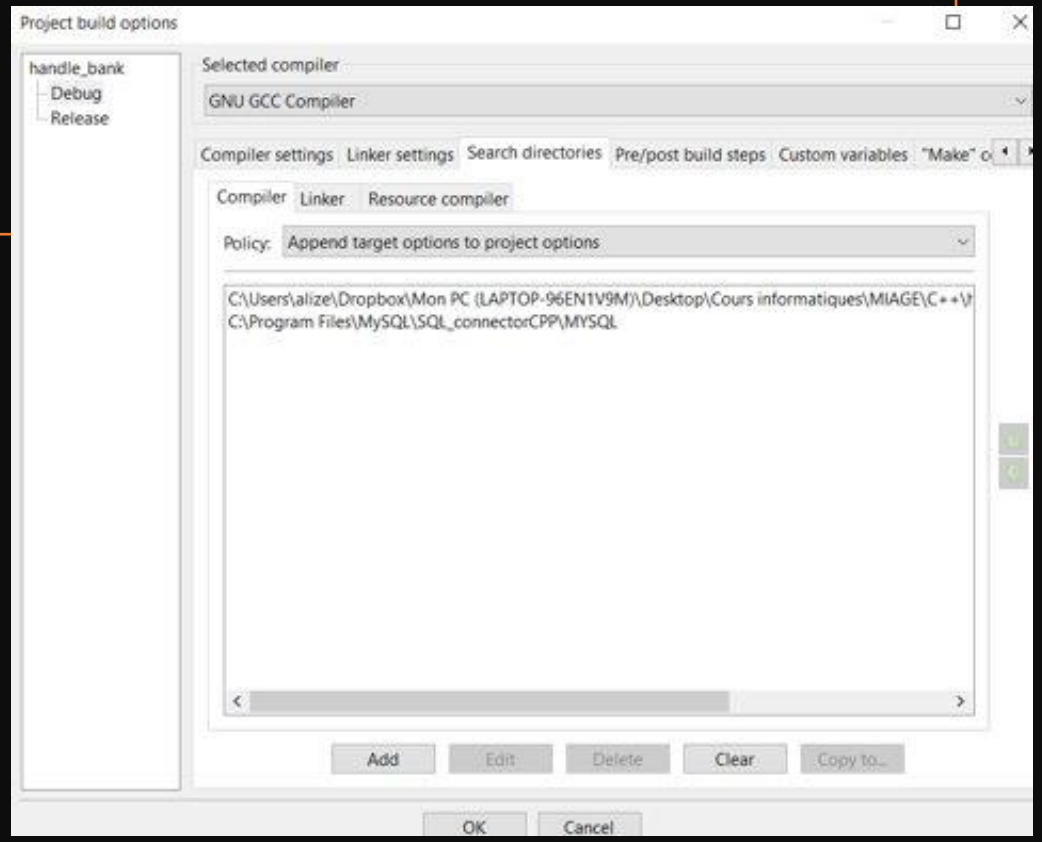
Utilisation des classes

```
// for stat the deal : we have borrower, Ariane space, and lender, Space-Bank, who sign the deal
Borrower *ArianeSpace = new Borrower("Ariane Space");
Lender *ISB = new Lender("International Space Bank");
Deal deal(1, ISB, "Ariane Space"); // id=1, Lender=A, borrower="company C1"

//The deal is for get many money to colonize Mars, for the 5th colonie (oh yeah : we are in 2184 years)
// It need many money : more than we can explain in 2023 (in our moment, it is represent 102 trillion dollar)
// Well the Space Bank get some pool in financial market : they call many Lender
Lender *ESABank = new Lender("Space Bank European", 3.8);
//ESABank->displayLender();
// And get some pool by bank : we need facilities by lenders
Facility* facilityEuro = new Facility(1, "2184-01-01", "2184-12-31", 10000000000.0, "EUR");
// a Lender could be have different facilities. for tha we stock facilities in pointer vector of Facility
std::vector<Facility*> vectESA;
vectESA = ESABank->vectorFacilitie(facilityEuro);
//ESABank->displayLender();
//ESABank->displayLender();
//ESABank->displayFacilities(vectESA);
//we got of pool this :
Pool* poolESA = new Pool(ESABank, vectESA);
// We creat differente part emits by facility and get by pool:
Part part1(poolESA, 50000.0, 5.0);
//part1.displayPart();
//borrower get all part
ArianeSpace->addPart(part1);
// and deal add

// Now, we creat the Ariane Space portfolio for handle the payement what due borrower to lenders
// We suppose the portefolio could be invest in the financial market for deal
Portefolio* portefolio = new Portefolio(5000000000.0, 1.8);
portefolio->addPartPF(part1, ArianeSpace);
//portefolio->calculPrice(ArianeSpace);
portefolio->displayPart();
```





Connecteur C++ pour MySQL : configuration sous code Block

Test pour la connexion à MySQL sur un autre projet :

```
4   #include <mysql.h>
5   using namespace std;
6
7   //for demonstration only. never save your password in the code!
8
9   int main()
10  {
11      MYSQL *sock;
12
13      //sock = mysql_init(0);
14      mysql::MySQL_Driver* driver;
15      mysql::get_mysql_driver_instance();
16
17      if(driver){
18          cout<<"OK"<<endl;
19      }else{
20          cout<<"NOK"<<endl;
21      }
22      return 0;
23  }
```

Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck/Vera+

File	Line	Message
		=== Build: Debug in firstConnectionSQL (compiler: GNU GCC Compiler) ===
C:\Users\al...		In function 'int main()':
C:\Users\al...	14	error: 'mysql' has not been declared
C:\Users\al...	14	error: 'driver' was not declared in this scope
C:\Users\al...	14	note: suggested alternative: 'div'
C:\Users\al...	15	error: 'mysql' has not been declared

```
2   #include <iostream>
3   #include <windows.h>
4   #include <mysql.h>
5   using namespace std;
6
7   //for demonstration only. never save your password in the code!
8
9   int main()
10  {
11      MYSQL *sock;
12
13      sock = mysql_init(0);
14
15      if(sock){
16          cout<<"OK"<<endl;
17      }else{
18          cout<<"NOK"<<endl;
19      }
20      return 0;
21  }
```

File	Line	Message
		=== Build: Debug in firstConnectionSQL (compiler: GNU GCC Compiler) ===
obj\Debug\m...		In function `main':
C:\Users\al...	13	undefined reference to `mysql_init'
		error: ld returned 1 exit status
		=== Build failed: 2 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===

Connector C++ MySql : Exemple d'implémentation à l'aide de ChatGPT

- L'implémentation est bonne puisque'elle reprend un format simple disponible sur le lien : [Microsoft-Azur]
- Cependant cela ne fonctionne pas, malgré la mise en place du connecteur C++ 8.0 pour MySQL

```
#include <mysql_driver.h>
#include <mysql_connection.h>
#include <cppconn/driver.h>
#include <cppconn/resultset.h>
#include <cppconn/statement.h>
#include <cppconn/prepared_statement.h>

using namespace sql;
```

```
void Deal::saveToDatabase(Connection* con) {
    PreparedStatement* pstmt = con->prepareStatement(
        "INSERT INTO deals (contract_number, agent, pool, borrower, project_amount, currency, "
        "contract_start_date, contract_end_date, deal_status) "
        "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
    );
    pstmt->setString(1, m_contractNumber);
    pstmt->setString(2, m_agent);
    pstmt->setString(3, m_pool);
    pstmt->setString(4, m_borrower);
    pstmt->setDouble(5, m_projectAmount);
    pstmt->setString(6, m_currency);
    pstmt->setString(7, m_contractStartDate);
    pstmt->setString(8, m_contractEndDate);
    pstmt->setString(9, m_dealStatus);

    pstmt->executeUpdate();

    delete pstmt;
}
```

Merci pour votre attention