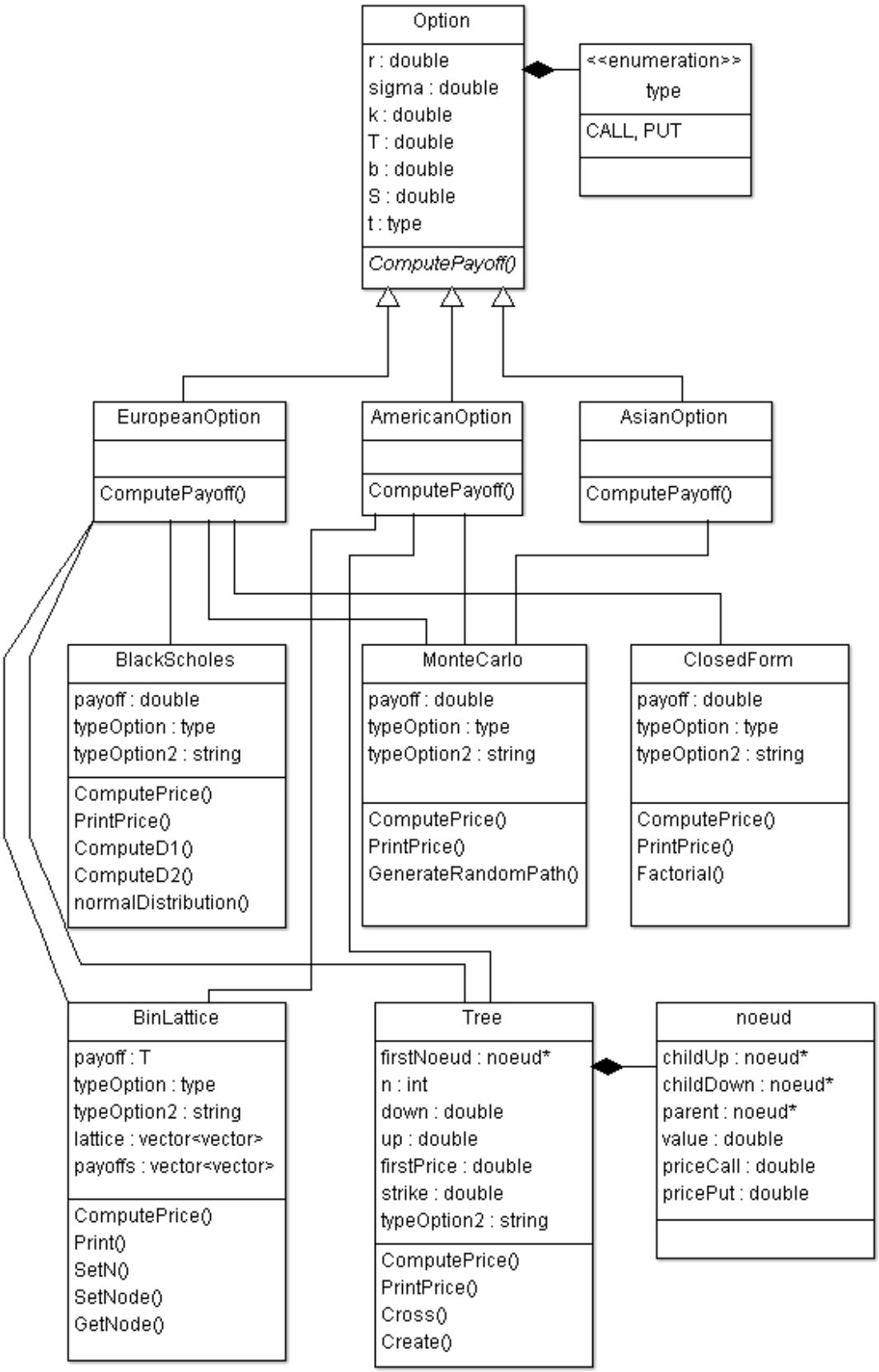
RAPPORT C++

I – UML CLASS DIAGRAM



II – ROLE AND PURPOSE OF CLASS

Class of Option type:

Option : It is a parent class who has as main function : ComputePayoff(). This method is Virtual, it aims to be redefined by its children classes, the attributes of the class are the option's parameters (r, sigma, k, T, S, b, and t which represents the option type ; Call or Put).

AmericanOption : Child class of the option, it redefines ComputePayoff() so that this one corresponds with the payoff of the american option.

AsianOption : Child class of the option, it redefines ComputePayoff() so that this one corresponds with the payoff of the asian option

EuropeanOption : Child class of the option, it redefined ComputePayoff() so that this one correspond at a payoff of asian option

Pricing Class :

Each following class have two common functions :

- ComputePrice() function which takes at least one parameter : a pointer on an option (American, Asian, European). This pointer only allows to pass the adress as a parameter, and not the whole object.
- PrintPrice() function who allows to display the result obtained.

BlackScholes : Being a closed form, blackscholes can only price the european options, this class owns the method normalDistribution() who allows to compute inverse normal law.

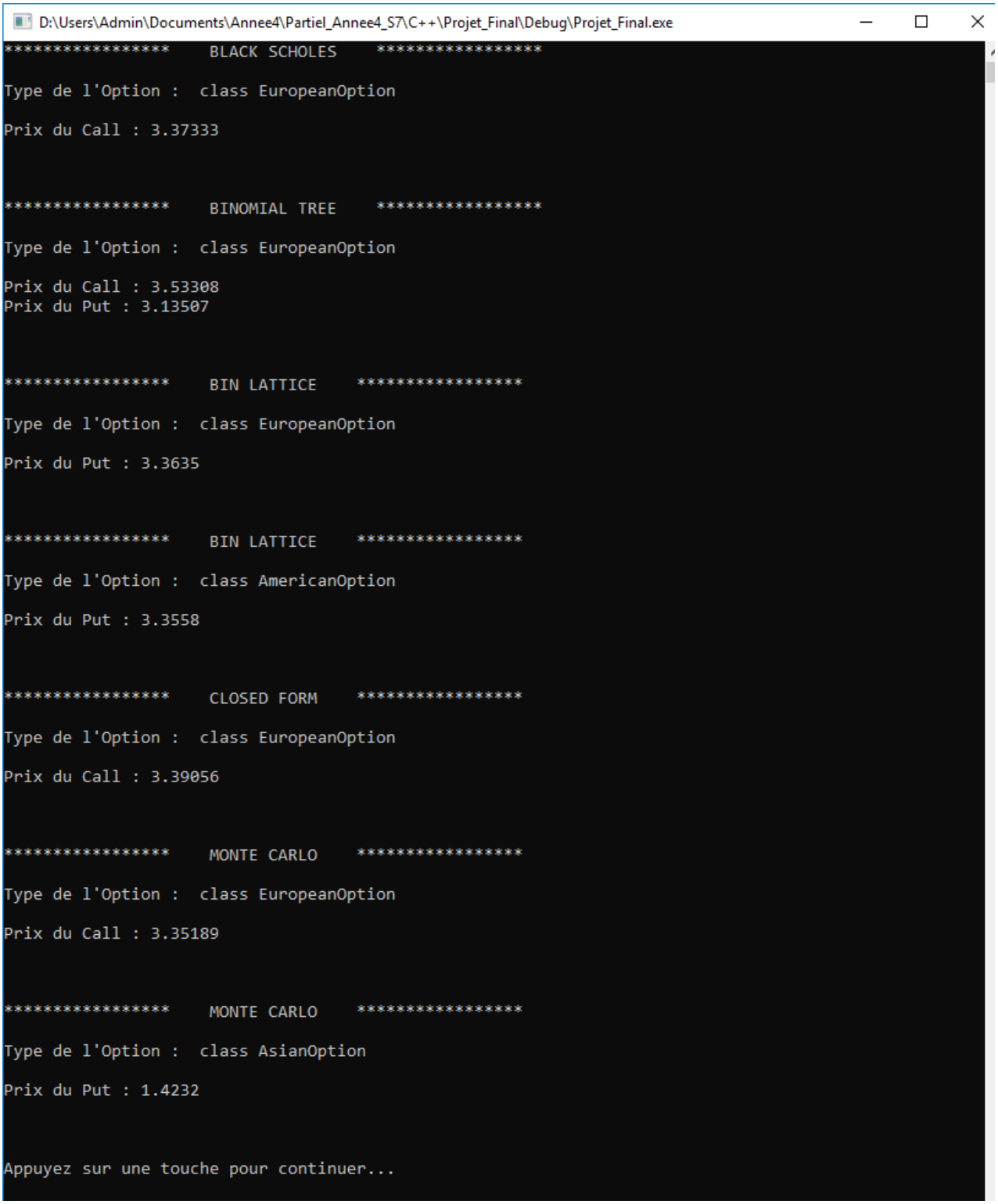
Monte Carlo : It allows to price european and asian options. This class uses vectors to stock the simulated prices. The method ComputePrice() takes two additionals parameters, double step and double generationNumber which represents the number of simulations generated.

ClosedForm : Being a closed form, Closed form can only price the european options

BinLattice is type generic<T>, it's the only class allowed to price the american options, it uses the vectors of vectors to be able to do a recombinant tree. The method ComputePrice() takes one additional parameter length which represents the depth of the recombinant tree.

Tree, Nœud : This two classes allow to create a non-recombinant binomial tree, each node owns two pointers on two others node classes, that permit to keep the history of prices paths but request a lot of calculations (2^n, n being the depth).

The **Projet_Final** class is the one where the main is coded, it instantiates the class and calls the method ComputePrice() then Print() which gives the following result :


D:\Users\Admin\Documents\Annee4\Partiel_Annee4_S7\C++\Projet_Final\Debug\Projet_Final.exe
********** BLACK SCHOLES **********
Type de l'Option : class EuropeanOption
Prix du Call : 3.37333
********** BINOMIAL TREE **********
Type de l'Option : class EuropeanOption
Prix du Call : 3.53308
Prix du Put : 3.13507
********** BIN LATTICE **********
Type de l'Option : class EuropeanOption
Prix du Put : 3.3635
********** BIN LATTICE **********
Type de l'Option : class AmericanOption
Prix du Put : 3.3558
********** CLOSED FORM **********
Type de l'Option : class EuropeanOption
Prix du Call : 3.39056
********** MONTE CARLO **********
Type de l'Option : class EuropeanOption
Prix du Call : 3.35189
********** MONTE CARLO **********
Type de l'Option : class AsianOption
Prix du Put : 1.4232
Appuyez sur une touche pour continuer...

III – DESIGN CHOICES

My code is composed of two family classes : **options** and **pricers**.

Option classes represent a type of option.

Pricer classes allow to compute the price of an option, they take as parameter an option, and return a price.

The code is as follows :

Instantiation of an **option** class (exemple : american option), it is here that one defines the parameter of the option (exemple : strike, Call or Put, ...)

Instantiation of a **pricer** class (exemple : monte carlo), here one chooses the type of pricer as well as its parameters (exemple : the number of generation, ...)

We call the method ComputePrice () of the class pricer with the option in parameter, thanks to the inheritance principle, regardless of the type of option.

The computePrice () method of the pricer will call the method ComputePayoff () of the option, which will return the payoff of the option according to its type (American, Asian, European), this is the principle of polymorphism. Finally, the Print () method will display the result on the screen.