**Document Technique**

Dans ce document, le fonctionnement du programme et l’ensemble des classes seront décrites

**!! Pour le lancement du projet il faut changer le path du document ‘’ DataBloom.xlsx ‘’ qui se trouve dans la classe Utility et dans la fonction GetDatas !!**

**Fonctionnement du Programme :**

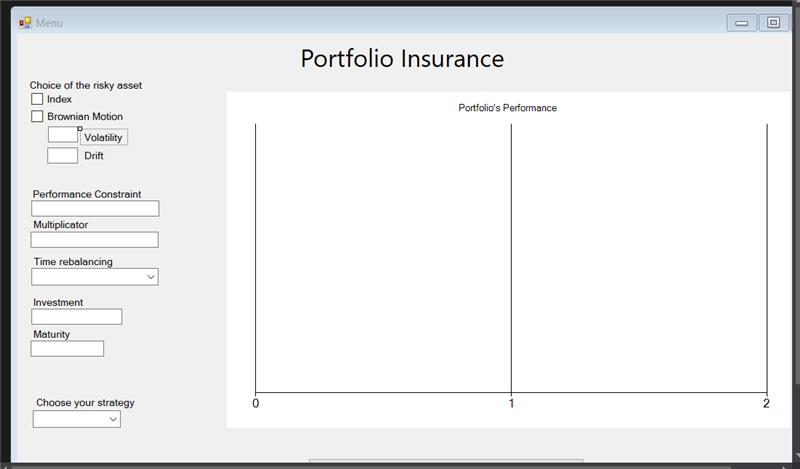
**Program**

C’est dans ce document que ce trouve la fonction Main() qui permet le lancement de l’application.

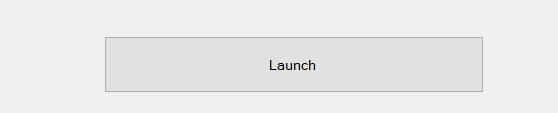
**Form1**

Lors du lancement du Main(), une application Windows Forms apparaît et permet :

* Le choix de l’actif risqué :
  + L’index LP02TREU
  + Mouvement brownien Géométrique : choix du drift et de la volatilité
* la sélection des paramètres des stratégies :
  + Performance Constraint
  + La Maturité
  + Le multiplicateur
  + La fréquence de rebalancement
  + Mise de départ
* Le choix de la stratégie



Après avoir sélectionné les paramètres, l’évolution du portefeuille est affichée sur la fenêtre de Chart en appuyant sur :



L’implémentation du code s’est basée sur la programmation orientée objets et notamment la relation d’héritage des classes.

**Dossier Stratgie**

Classe *Strategie*

Cette classe, abstraite, contient les attributs communs aux 3 stratégies :

|  |  |
| --- | --- |
| Attributs | Descriptions |
| Risky\_returns | La liste des rendements de l’actif risqué |
| *Riskless\_returns* | La liste des rendements de l’actif non-risqué |
| *Floor* | Le montant minimale à protéger |
| *Cushion* | Valeur en excès du floor |
| *PerfConstraint* | La contrainte de performance |
| *Mise\_de\_depart* |  |
| *Maturity* |  |
| *Period* |  |

Elle contient par ailleurs la méthode abstraite Weights qui permet le calcul des poids alloués à chaque actif à chaque date t durant l’implémentation des classes dérivés.

La stratégie d’allocation étant propre à chaque stratégie, c’est pourquoi cette méthode est abstraite.

Classe *CPPI*

Cette classe hérite de la classe stratégie et permet l’implémentation de la stratégie du CPPI

Cette classe possède l’attribut multiplicateur en plus.

Classe *TIPP*

Cette classe hérite de la classe stratégie et permet l’implémentation de la stratégie du TIPP

Cette classe possède l’attribut floor\_tipp et le multiplicateur en plus.

Classe *RD*

Cette classe hérite de la classe stratégie et permet l’implémentation de la stratégie du Relative Drawdown

Cette classe possède l’attribut multiplicateur en plus

**Dossier Datas**

Classe Data

Cette classe contient toutes les données d’un actif à chaque date t

|  |  |
| --- | --- |
| Attributs | Descriptions |
| Date | Date de la donnée |
| *Price* | Prix de la donnée |
| *Volatility* | Volatilité de l’asset |
| *Asset* | Permet de différencier les actifs |
| *Returns* | Le rendement de l’actif. |
| *Portfolio\_Value* | Valeur du portefeuille à la date t |
| *Weights* | Poids alloué à l’actif à la date t selon la stratégie |

Cette classe permet le stockage de toute l’information.

Elle permet la création d’une liste de cette classe pour la manipulation d’information.

Classe Utility

Cette classe contient toutes les méthodes nécessaires au bon fonctionnement du code

GetDatas

Cette méthode permet de la lecture/récupération des datas dans l’Excel

Elle permet de récupérer les données dans une liste de la classe Data.

Grâce à un linq selon le critère de l’attribut Asset (1 ou 2), on sépare cette liste en 2 listes distinctes : dataAsset1 et dataAsset2.

Calcul\_returns

Cette méthode permet de calculer le rendement à chaque date selon la formule suivante :