# Modélisation informatique d’un portefeuille

La conception de ce projet nécessite une modélisation informatisée. Celle-ci a suivi la représentation naturelle que nous avions d’un portefeuille. Nous avons donc recouru l’utilisation de la programmation orientée objet. La programmation orientée objet apporte des avantages face à la programmation procédurale.

Tout d’abord elle permet d’obtenir un code évolutif. Les types d’objets créés peuvent servir de base pour d’autres objets. Les objets ont chacun leur propriétés et méthodes. Le code est plus compréhensible. Il est donc aisé de voir ce qu’une fonction manipule et à quoi correspondent les variables disponibles.

A des fins de confidentialités, les attributs des classes sont privés. Leurs valeurs sont accédées et modifiées via des accesseurs (getter & setter). De plus toute les classes sont équipé de la fonction \_\_repr\_\_ permettant la représentation des objets imbriquées.

Voyons donc la structure de nos classes.

## La classe Broker :

Elle contient les attributs suivants :

\_\_broker\_name : permet d’identifier le broker avec son nom

\_\_broker\_fees : permet de fixer les frais du broker

Ces frais sont sous forme suivante :

list(borne inférieur 1, borne supérieur 1, taux fixe, taux variable, borne inférieur 2, borne supérieur 2, taux fixe, taux variable […] )

*Exemple : ﻿*

*B1 =Broker(‘Broker\_1’, (0,500,1.99,0,500,10\*\*10,0,0.006))*

*print(B1)*

*﻿{broker\_name: Broker\_1’, broker fees: (0, 500, 1.99, 0, 500, 10000000000, 0, 0.006)}*

Elle contient les méthodes suivantes :

comp\_broker\_fees(asset\_quantity, asset\_price)

Cette fonction permet de calculer les frais de courtage correspondants à une position (quantités d’actif \* prix de l’actif). Cette fonction est vitale pour l’ensemble du projet, elle est réutilisée dans différentes classes.

## La classe User :

Elle contient les attributs suivants :

\_\_user\_lastname : permet de définir le nom de l’utilisateur

\_\_user\_firstname : permet de définir le prénom de l’utilisateur

\_\_user\_risk : le risque cumulatif des différents portefeuilles

\_\_user\_list\_portfolio : contient les portefeuilles associés à un utilisateur

Elle contient les méthodes suivantes :

﻿add\_user\_list\_portfolio (self, user\_portfolio)

Cette fonction permet d’ajouter des portefeuilles à l’utilisateur.

*Exemple :*

*﻿ u1=User("Mace","Nicolas")*

*Print(u1)*

*{user\_lastname: Mace, user\_firstname: Nicolas, user\_risk: None, user\_list\_portfolio:[]}*

*u1.* add\_user\_list\_portfolio (portefeuille\_1)

*u1.* add\_user\_list\_portfolio (portefeuille\_2)

*Print(u1)*

*{user\_lastname: Mace, user\_firstname: Nicolas, user\_risk: None, user\_list\_portfolio:[portefeuille\_1, portefeuille\_2]}*

## La classe Asset :

La classe asset est une classe mère à hériter. Elle représente les « actifs ». Elle sera utilisée pour les classes filles tels que les actions, obligations…

Elle contient les attributs suivants :

﻿ \_\_asset\_price : le prix de l’actif

\_\_asset\_ISIN : le nom code de l’actif

Elle contient les méthodes suivantes :

﻿comp\_asset\_cost(self, asset\_quantity, broker)

Cette fonction permet de calculer le prix d’un actif selon sa quantité et son broker.

## La classe Stock :

Cette classe hérite de la classe Asset.

Elle contient les attributs suivants :

﻿\_\_stock\_currency : l’unité de l’action (la monnaie)

Mais aussi issue de la classe mère :

\_\_asset\_price : le prix de l’actif

\_\_asset\_ISIN : le nom code de l’actif

Elle contient les méthodes suivantes :

comp\_asset\_cost(self, asset\_quantity, broker)

*Exemple:*

*﻿s1=Stock("Stock\_1","€",6.95)*

*s1*

*﻿{asset\_ISIN: Stock\_1, asset\_price: 6.95, stock\_currency:€}*

## La classe Investment :

Cette classe permet de créer l’équivalent numérique des lignes d’un portefeuille.

Elle contient les attributs suivants :

﻿ \_\_investment\_asset : permet de préciser l’actif en question

\_\_investment\_quantity : permet de définir la quantité d’actif investit

\_\_investment\_date : permet d’enregistrer la date d’achat

\_\_investment\_cost : permet de définir le ﻿cout unitaire à l'achat sans les frais de courtages

Elle contient les méthodes suivantes :

﻿comp\_investment\_broker\_fees(self,broker): calcule les frais de broker associés

﻿comp\_investment\_price(self,broker): calcule le prix de l’investissement

﻿comp\_investment\_cost(self,broker): calcule le cout de l’investissement

﻿comp\_investment\_PnL(self, broker): calcule le PnL de l’investissement à la date d’achat à date de sortie

*Exemple :*

*﻿ i1=Investment(s1,100,"10/10/2017",6.95)*

*{investment\_asset: {asset\_ISIN: Stock\_1, asset\_price: 6.95, stock\_currency:€}, investment\_quantity: 100, investment\_date: 10/10/2017, investment\_cost: 6.95}*

On voit ici le début de l’emboitement des objets. En effet, notre objet investissment i1 inclus l’objet Stock(Actif) s1.

La classe User :

Elle contient les attributs suivants :

Elle contient les méthodes suivantes :

La classe User :

Elle contient les attributs suivants :

Elle contient les méthodes suivantes :