La conception de ce projet nécessite 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 peuvent être accéder et modifier via des accesseurs (getter & setter). De plus toute les classes sont équipé de \_\_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 de définir d’identifier le broker

\_\_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 […] )

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 (quantité 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 l’utilisateur

Elle contient les méthodes suivantes :

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

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

La classe Asset:

La classe asset est une classe mère à hériter. Elle sera utilisée pour les classes filles tels que 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 cout d’un actif selon sa quantité et son broker.

La classe Stock :

Elle contient les attributs suivants :

﻿\_\_stock\_currency

La classe Investment : est une classe permettant de créer les lignes d’un portefeuille.

Elle contient les attributs suivants :

﻿ \_\_investment\_asset : permet d’ajouter l’actif investit

\_\_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 broker fees

Elle contient les méthodes suivantes :

﻿comp\_investment\_broker\_fees(self,broker):

La function calcule les frais de broker correspondant à l’investissement

﻿def comp\_investment\_price(self,broker):

La fonction calcule le prix de l’investissement ( frais de broker compris)

﻿def comp\_investment\_cost(self,broker):

La fonction calculi le cout de l’investissement ( frais de broker compris)

﻿def comp\_investment\_PnL(self, broker):

La fonction calcule le PnL de l’investissement à la date d’achat à date de sortie

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 :