

Bank

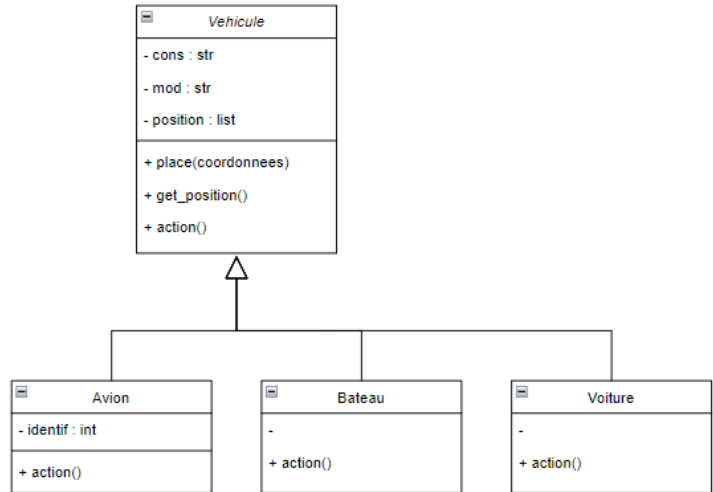
1. Présentation

Pour ce projet, vous devrez au préalable rechercher des informations sur les notions de classe, super classe, sous classe, héritage, surcharge, polymorphisme.

En résumé une super classe est une classe générale reliée à d'autres plus spécialisées (sous classes). Les sous classes héritent des propriétés (attributs, méthodes) de leur super classe. Elles peuvent comporter des propriétés spécifiques supplémentaires.

Par exemple :

Les voitures, les bateaux, et les avions sont des véhicules. Ils possèdent tous une marque et un modèle.



10.9_01 bank_P.drawio

1.1. Une seule classe

```

class Vehicule:
    # constructeur
    def __init__(self, constructeur, modele):
        self.cons = constructeur # le fabricant
        self.mod = modele # le modele
        self.position = [] # coordonnées GPS @44.8236488,-0.3279314 (N W)
    # méthodes
    def place(self, coordonnees):
        '''Place le véhicule sur des coordonnées GPS
        coordonnees tuple (+N/-S, +E/-W)
        '''
        self.position = list(coordonnees)
    def get_pos(self):
        return self.position
    def action(self):
        return 'je peux me deplacer'
if __name__ == "__main__":
    my_car = Vehicule('Citroen', 'C4')
    my_car.place((44.8236488,-0.3279314))
    
```

10.9.02 classe.py

1.2. La notion d'héritage

```

class Vehicule:
    # super classe
    # constructeur
    def __init__(self, constructeur, modele):
        self.cons = constructeur # le fabricant
        self.mod = modele # le modele
        self.position = [0, 0] # coordonnées GPS par exemple 44.8236488,-0.3279314 (N W)
    def place(self, coordonnees):
        '''Place le véhicule sur des coordonnées GPS
        coordonnees tuple (+N/-S, +E/-W)
        '''
        self.position = list(coordonnees)
    def get_pos(self):
        return self.position
    def action(self):
        return 'je peux me deplacer'
class Bateau(Vehicule):
    ''' Un bateau est un vehicule qui navigue'''
    pass
    
```

luc.vincent@ac-bordeaux.fr/
10.9 bank_P.docx

11. Projet

```
class Avion(Vehicule):
    ''' Un avion est un vehicule qui vole '''
    pass
class Voiture(Vehicule):
    '''Une voiture est un véhicule qui roule'''
if __name__ == "__main__":
    my_car = Voiture('Citroen', 'C4')
    my_car.place((44.8236488,-0.3279314))
    print(my_car.get_pos(), f'je suis {my_car.mod} {my_car.action()}')
#heritage
my_boat = Bateau('Couach', '2300 FLY')
my_boat.place((45.0101,+0.1234))
print(my_boat.get_pos(), f'je suis {my_boat.mod} {my_boat.action()}')
```

10.9.03 heritage.py

1.3. Polymorphisme et surcharge

```
class Vehicule:
    # constructeur
    def __init__(self, constructeur, modele):
        self.cons = constructeur # le fabricant
        self.mod = modele # le modele
        self.position = [0, 0] # coordonnées GPS par exemple 44.8236488,-0.3279314 (N W)

    def place(self, coordonnees):
        '''Place le véhicule sur des coordonnées GPS
        coordonnees tuple (+N/-S, +E/-W)
        '''
        self.position = list(coordonnees)

    def get_pos(self):
        return self.position

    def action(self):
        return 'je peux me déplacer'

class Bateau(Vehicule):
    ''' Un bateau est un vehicule qui navigue'''
    def action(self):
        return 'je peux naviguer'

class Avion(Vehicule):
    ''' Un avion est un vehicule qui vole
    Un avion dispose d'un numéro d'identification
    '''
    # On pourrait recopier
    # def __init__(self, constructeur, modele, identifiant):
    #     self.cons = constructeur # le fabricant
    #     self.mod = modele # le modele
    #     self.position = [0, 0] # coordonnées GPS par exemple 44.8236488,-0.3279314 (N W)
    #     self.id = identifiant
    # surcharge
    def __init__(self, cons, model, identif):
        super().__init__(cons, model)
        self.id = identif
    def action(self):
        return 'je peux voler'

class Voiture(Vehicule):
    '''Une voiture est un véhicule qui roule'''
    pass

if __name__ == "__main__":
    my_car = Voiture('Citroen', 'C4')
    my_car.place((44.8236488,-0.3279314))
    print(my_car.get_pos(), f'je suis {my_car.mod} {my_car.action()}')
#heritage
my_boat = Bateau('Couach', '2300 FLY')
my_boat.place((45.0101,+0.1234))
print(my_boat.get_pos(), f'je suis {my_boat.mod} {my_boat.action()}')
# surcharge
my_plane = Avion('Airbus', 'A320', 3827)
# polymorphisme
for v in (my_car, my_boat, my_plane):
    print(f'je suis {v.mod} et {v.action()}')
```

10.9.04 heritage.pysurcharge_polymorphisme.py

11. Projet

2. Le projet

2.1. Cahier des charges

Il s'agit d'écrire une application interactive qui permet de gérer les comptes bancaires d'une personne. Les fonctionnalités attendues sont les suivantes :

- Création (Suppression) d'un compte,
- Affichage d'un compte donné,
- Saisie d'une ligne comptable pour un compte donné,
- Sauvegarde des comptes
- Sauvegarde des lignes comptables

2.2. Afficher le menu principal et les options (niveau 1 Note 6)

L'exécution du programme affiche le menu suivant :

```
1. Creer un compte
2. Afficher un compte
3. Créer une ligne comptable
4. Sortir
5. De l'aide
6. Afficher tous les comptes
7. Afficher toutes les opérations
```

Votre choix de 1 à 5

L'utilisateur choisi une valeur pour exécuter l'opération souhaitée

Votre choix de 1 à 5 1

```
Type du compte [Types possibles :
C(ourant), J(oint), E(pargne)] : C
Numero du compte : FR123456
Premiere valeur créditée : 100
Compte FR123456 de type C créé Solde : 100.00
```

Votre choix de 1 à 5 1

```
Type du compte [Types possibles :
C(ourant), J(oint), E(pargne)] : E
Numero du compte : FR23456
Premiere valeur créditée : 200
Taux de placement : 5
Compte FR23456 de type E créé Solde : 200.00
```

Dans le cas d'un compte epargne l'utilisateur indique le taux en pourcentage.

Si l'utilisateur choisit l'option 2, le programme affiche les caractéristiques d'un compte.

Votre choix de 1 à 5 2

```
Quel compte souhaitez vous afficher ? : FR123456
Le compte n° : FR123456 est un compte C
Valeur actuelle : 100.0
```

Pour l'option 3 il s'agit de fournir toutes les informations pour créer un compte

Votre choix de 1 à 5 3

```
Sur quel compte souhaitez vous intervenir ? : FR123456
choisir la nature de l'opération
```

1. Pour effectuer un dépôt
2. Pour effectuer un retrait
3. Pour effectuer un transfert

Votre choix 1 à 3 :

11. Projet

```
Votre choix de 1 à 5 3
Sur quel compte souhaitez vous intervenir ? : FR123456
choisir la nature de l'opération
1. Pour effectuer un dépôt
2. Pour effectuer un retrait
3. Pour effectuer un transfert
Votre choix 1 à 3 : 1
Votre dépôt ? 50.50
Dépôt réalisé, le compte FR123456 a pour solde 150.50
```

```
Votre choix de 1 à 5 3
Sur quel compte souhaitez vous intervenir ? : FR123456
choisir la nature de l'opération
1. Pour effectuer un dépôt
2. Pour effectuer un retrait
3. Pour effectuer un transfert
Votre choix 1 à 3 : 2
Votre retrait ? 25
Retrait réalisé, le compte FR123456 a pour solde 125.50
```

```
Votre choix de 1 à 5 3
Sur quel compte souhaitez vous intervenir ? : FR1
choisir la nature de l'opération
1. Pour effectuer un dépôt
2. Pour effectuer un retrait
3. Pour effectuer un transfert
Votre choix 1 à 3 : 3
Vers quel compte souhaitez réaliser un transfert ? : FR2
Valeur du transfert ? 100
Commencer le transfert ✎
Tranfert réalisé ☒
le compte FR1 a maintenant pour solde 100.00
le compte FR2 a maintenant pour solde 205.00
```

Au niveau 1, seuls les affichages sont attendus, le calcul des valeurs se fera au niveau 2.

2.3. Déterminer les variables nécessaires

Un compte bancaire est défini par

Données	Exemple	Nom et Type
Numéro de compte	FR4010.205.530	account_nb : str
Type de compte	Courant	account_type : str
Valeur	-20.3	account_balance : float
Date	12 10 23	date : str
Taux de rémunération	0.05	Rate : float

La structure de mémorisation des comptes créés doit pouvoir accepter autant de compte que nécessaire.

Il faudra prévoir 2 fichiers de journaux, l'un pour les comptes l'autre pour les opérations.

11. Projet

3. Consignes à respecter

Vous devez travailler en équipe de 3.

Le programme principal commence obligatoirement par l'entête (exemple) :

```

1  __author__ = "One solo developer" # écrit le code
2  __authors__ = ["One developer", "And another one", "etc"]
3  __contact__ = "mail@example.com"
4  __copyright__ = "Copyright $YEAR, $COMPANY_NAME"
5  __credits__ = ["One developer", "And another one", "etc"] # signaler des corrections de bugs, des suggestions
6  __date__ = "2022/10/11" # "YYYY/MM/DD"
7  __email__ = "mail@example.com"
8  __license__ = "CC BY-SA"
9  __maintainer__ = "developer" # personne qui va corriger les bugs et améliorer
10 __status__ = "Production" # "Prototype", "Développement", "Production"
11 __version__ = "0.0.1"
12
13 '''
14     Unit_convert
15     L'ordinateur réalise la conversion d'une valeur exprimée dans une unité
16     et affiche le résultat dans une autre unité.
17
18     Validé : (sys.version)
19     Python '3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)]'
20     Modules natifs
21     math
22     Modules externes
23     Aucun
24 '''

```

Le programme est réalisé sous forme de modules, dont les noms sont imposés.

De même vous devrez respecter les noms des fichiers de log.csv

Th bank.py


Th choice.py

Th constant.py


Th count.py

Th count_utility.py

Th exception.py

 log_banking_transaction.csv

Th logbanking.py

 logbook_count.csv

Th logbooks.py

3.1. Le module bank

```

# Header

# Modules du projet

def main():
    '''Afficher le menu principal'''
    pass

if __name__ == "__main__":
    main()

```

3.2. Le module choice

luc.vincent@ac-bordeaux.fr/
10.9 bank_P.docx

11. Projet

Il contiendra les fonctions de choix

3.3. Le module constant

Il contient les constantes du programme, en particulier :

```
# Définir le fichier de log des comptes
FILENAME_ACCOUNT = "logbook_count.csv"

# Définir le fichier de log des transactions
FILENAME_TRANSACTION = "log_banking_transaction.csv"
```

4. Poursuivre le projet

4.1. Le module count (Niveau 2 note 12)

Le module count contient la définition des classes Count, pour les comptes simples et InterestAccount(Count) pour les comptes rémunérés.

```
class Count():
    ''' un compte en banque '''
    pass

class InterestAccount(Count):
    '''Comptes récompensés les dépôts sont rémunérés au taux indiqué'''
    pass

if __name__ == "__main__":
    #jeu de test à compléter
    account_one = Count("FR1", 'C', 100.50)
```

Le calcul des valeurs doit être maintenant exact. Par exemple si

```
Votre choix de 1 à 5 2
  Quel compte souhaitez vous afficher ? : FR2
Le compte n° : FR2 est un compte E
dont le taux est de 5.0%
Valeur actuelle : 200.0
```

Alors un dépôt devrait afficher

```
Votre choix de 1 à 5 3
  Sur quel compte souhaitez vous intervenir ? : FR2
choisir la nature de l'opération
    1. Pour effectuer un dépôt
    2. Pour effectuer un retrait
    3. Pour effectuer un transfert
Votre choix 1 à 3 : 1
Votre dépôt ? 100
Dépôt réalisé, le compte FR2 a pour solde 305.00
```

On ajoutera dans le module count_utility une fonction qui permettra de valider l'existence d'un compte ou pas

Afin de pouvoir à minima gérer des situations comme :

luc.vincent@ac-bordeaux.fr/
10.9 bank_P.docx

```

1. Creer un compte
2. Afficher un compte
3. Créer une ligne comptable
4. Sortir
5. De l'aide
6. Afficher tous les comptes
7. Afficher toutes les opérations

```

```

Votre choix de 1 à 5 2
  Quel compte souhaitez vous afficher ? : FGH
Le systeme ne connait pas le compte FGH

```

4.2. Le module count et le module exception (niveau 3 note 14)

On ajoutera un mécanisme permettant de s'assurer qu'une opération peut avoir lieu (retrait ou transfert) uniquement si le solde du compte le permet.

On utilisera une levée d'erreur personnalisée dans le module exception qui est le suivant :

```

class BalanceException(Exception):
    ''' Un objet BalanceException héritant de la classe prédéfinie Exception
    On lèvera une erreur en cas de problème de solde sur le compte'''
    pass

```

Exemple :

```

Votre choix de 1 à 5 2
  Quel compte souhaitez vous afficher ? : FR1
Le compte n° : FR1 est un compte C
Valeur actuelle : 100.0

1. Creer un compte
2. Afficher un compte
3. Créer une ligne comptable
4. Sortir
5. De l'aide
6. Afficher tous les comptes
7. Afficher toutes les opérations

Votre choix de 1 à 5 3
  Sur quel compte souhaitez vous intervenir ? : FR1
  choisir la nature de l'opération
    1. Pour effectuer un dépôt
    2. Pour effectuer un retrait
    3. Pour effectuer un transfert
  Votre choix 1 à 3 : 2
  Votre retrait ? 200
  Retrait impossible, le compte FR1 a pour solde 100.00

```

Il en sera de même pour une tentative de transfert impossible :

```

1. Creer un compte
2. Afficher un compte
3. Créer une ligne comptable
4. Sortir
5. De l'aide
6. Afficher tous les comptes
7. Afficher toutes les opérations

Votre choix de 1 à 5 3
  Sur quel compte souhaitez vous intervenir ? : FR1
  choisir la nature de l'opération
    1. Pour effectuer un dépôt
    2. Pour effectuer un retrait
    3. Pour effectuer un transfert
  Votre choix 1 à 3 : 3
  Vers quel compte souhaitez réaliser un transfert ? : FR2
  Valeur du transfert ? 200
  Commencer le transfert
  Transfert impossible
  le compte FR1 a seulement pour solde 100.00
  
```

4.3. Le module logbooks et le fichier logbook_count.csv (niveau 4 note 17)

Dans les versions précédentes c'est une structure mémoire qui conservait les informations. Nous ajoutons maintenant en plus le stockage de cette structure dans un fichier au format .csv

Vous devrez utiliser le module pandas pour gérer le fichier logbook_count.csv

Exemple de contenu du fichier

	A	B	C	D	E	F	
1	Date	Heure	Compte	Type	Taux	Solde	
2	10/17/23	19:03:38	FR1	C		100.0	
3	10/17/23	19:04:36	FR2	E	5.0	210.0	
4	10/18/23	08:17:20	FR123456	C		125.5	
5	10/18/23	08:31:45	FR23456	E	5.0	200.0	
6	10/18/23	14:26:55	FRE	C		100.0	

Toutes les opérations bancaires doivent mettre à jour la structure ET ce nouveau fichier

Le choix 6 du menu principal doit maintenant conduire à l'affichage du fichier dans la console.

```

Votre choix de 1 à 5 6
      Date      Heure      Compte      Type      Taux      Solde
0      10/17/23  19:03:38      FR1      C      100.00
1      10/17/23  19:04:36      FR2      E      5.00    210.00
2      10/18/23  08:17:20  FR123456  C      125.50
3      10/18/23  08:31:45  FR23456  E      5.00    200.00
4      10/18/23  14:26:55      FRE      C      100.00
  
```


11. Projet

4.4. Le module logbanking et le fichier log_banking_transaction.csv (niveau 5 note 20)

Nous ajoutons un fichier log_banking_transaction.csv journal de toutes les lignes comptables réalisées.

	A	B	C	D	E	F
1	Date	Heure	Compte	Nature	Valeur	Solde
2	10/18/23	15:17:58	FR1	Crédit	50.0	150.0
3	10/18/23	15:18:44	FR1	Débit	25.0	125.0
4	10/18/23	15:19:48	FR1	T-Débit	50.0	75.0
5	10/18/23	15:19:48	FR2	T-Crédit	50.0	215.0
6	10/18/23	15:22:09	FR2	Crédit	50.0	217.5
7	10/18/23	15:26:08	FR2	Débit	17.5	200.0
8	10/18/23	15:26:58	FR2	Crédit	100.0	305.0
9	10/18/23	15:30:08	FR1	T-Débit	10.0	65.0
10	10/18/23	15:30:08	FR2	T-Crédit	10.0	315.5

Le choix 7 du menu principal doit maintenant conduire à l’affichage du fichier dans la console

Votre choix de 1 à 5 7

	Date	Heure	Compte	Nature	Valeur	Solde
0	10/18/23	15:17:58	FR1	Crédit	50.00	150.00
1	10/18/23	15:18:44	FR1	Débit	25.00	125.00
2	10/18/23	15:19:48	FR1	T-Débit	50.00	75.00
3	10/18/23	15:19:48	FR2	T-Crédit	50.00	215.00
4	10/18/23	15:22:09	FR2	Crédit	50.00	217.50
5	10/18/23	15:26:08	FR2	Débit	17.50	200.00
6	10/18/23	15:26:58	FR2	Crédit	100.00	305.00

5. Compétences évaluées

- ✓ Analyser et modéliser un problème
- ✓ Décomposer un problème en sous problèmes
- ✓ Concevoir des solutions algorithmiques
- ✗ Mobiliser les concepts et les technologies
- ✓ Traduire un algorithme dans un langage de programmation
- ✗ Développer des capacités d’abstraction et de généralisation

	D	C	B	A
APP Rechercher l'information utile à l'aide de sources fiables	Je copie des solutions dans des sources sans liens apparents entre elles.	Je trie les éléments intéressant dans les sources rencontrées pour les utiliser dans ma solution.	J'identifie et trie les éléments intéressant dans les sources rencontrées pour les utiliser dans ma solution. Je les documente dans mon code.	Je m'inspire de différentes sources pour créer ma solution, en les comparant entre elles pour trouver la plus adaptée. Je cite et documente ces sources.
REA Imaginer et concevoir une solution modulaire : décomposer en blocs, se ramener à des sous problèmes simples et indépendants	J'écris quelques fonctions.	J'écris et utilise des fonctions que je documente.	J'écris et utilise des fonctions, des classes et des modules adaptés que je documente.	J'écris et utilise les fonctions, classes et modules les plus adaptés au problème. Je documente et explique mes choix.