**BLOC2 ALGO.4 : L'héritage en programmation objet**

|  |
| --- |
| Objets d’un traitement informatique  |

Introduction

L'héritage est l'un des mécanismes fondamentaux de la POO (Programmation orientée objet). Il permet de créer des classes à partir de classes existantes. L'objectif de ce chapitre est de découvrir son fonctionnement.

**Question 1** : donner un exemple d'héritage

# Cas d'utilisation

Une banque souhaite gérer ses clients dans une application. Pour chaque client elle souhaite connaitre :

* nom
* prénom
* genre (1 pour homme ; 2 pour femme)
* date de naissance
* catégorie socio professionnelle
* adresse complète
* téléphone
* email

Bien sur, les comptes seront aussi gérés dans une application. Un compte doit être rattaché à une seule personne. Nous souhaitons connaître pour un compte :

* le numéro
* le titulaire
* le solde
* devise

**Question 2** : faire le diagramme de classe de ce contexte

**Question 3** : Faire le constructeur, accesseurs (getteur) et mutateur (setteur) de la classe compte

On supposera que ces différentes méthodes "standards" seront faites pour les autres classes.

**Question 4** : Ecrire la méthode débiter et créditer de la classe compte

**Question 5** : Ecrire la méthode Decrire de la classe compte qui retourne une chaine de caractère avec la description d'un compte.

Exemple de la chaine de caractère retournée :

n° : 13245411324 - solde : 548200 €

**Question 6** : Ecrire la méthode info\_comptes qui permet d'afficher l'ensemble des comptes d'un adhérent.

Liste des comptes de M. Gravouil Benjamin

n° : 13245411324 - solde : 2548200 €

n° : 22544771289 - solde : -214 €

Exemple d'affichage :

**Question 7** : créer le programme principal permettant d'effectuer ceci :

Créer le client ci-dessous :

* nom : Gravouil
	+ prénom : Benjamin
	+ genre : 1
	+ date de naissance : 02/05/1977
	+ catégorie : enseignant
	+ adresse : lycée Paul Lapie, 5 boulevard AB, 92400 COURBEVOIE
	+ téléphone : 01.01.01.01.01
	+ email : prof.gravouil@gmail.com
* Créer ses comptes (à gérer dans une liste) :
	+ n° : 13245411324 - solde : 2548200 €
	+ n° : 32544771289 - solde : 1000 €
	+ n° : 22544771289 - solde : -214 €
* Afficher la liste des comptes du client

# L'héritage

Supposons maintenant que nous ayons à gérer un nouveau type de compte : le compte épargne. Comme un compte classique, un compte épargne possède un numéro, un titulaire, un solde et une devise. Sa spécificité est qu'il permet d'appliquer des intérêts à l'argent déposé sur le compte.

**Question 8 :** remplir le tableau ci-dessous :

|  |  |
| --- | --- |
| Liste attribut(s) identique(s) entre le compte épargne et un compte "classique" | Liste attribut(s) différent(s) entre le compte épargne et un compte "classique" |
|  |  |



## Formalisme héritage en algorithme

* écriture d'une classe "fille" :

CLASSE fille **hérite de** mère

début

…

fin

* Pour revenir à la classe mère : utilisez le préfixe **super**

## Formalisme héritage en java

* écriture d'une classe "fille" :

public class fille **extends** mère

{

…

}

* Pour revenir à la classe mère : utilisez le préfixe **super**

**Question 9 :** Modifier votre diagramme de classe

**Question 10 :** Ecrire les méthodes "classiques" (constructeur, getteur, setteur) de la nouvelle classe

**Question 11 :** Ecrire la fonction Decrire de la classe compte\_epargne qui retourne une chaine de caractère avec la description d'un compte :

Exemple de la chaine de caractère retournée :

Compte Epargne n° : 32544771289 - solde : 1000 € - taux = 5%

**Attention le taux est un réel. Par exemple 0.05 correspond à 5%. Il ne faut pas afficher 0.05 mais bien 5%.**

**Question 12 :** Ecrire la méthode ajouter\_intéret de la classe EPARGNE qui permet de rajouter les intérêts au solde du compte. Si le solde est négatif, alors rien ne sera crédité (ni débité).

Le tableau ci-dessous rassemble les trois niveaux de visibilité utilisables.



# Le polymorphisme

Afin de gagner en généricité, on peut appliquer le même code à des objets de types différents, lorsque les classes de ces objets sont liées par héritage. C'est le principe du polymorphisme.

**Question 13 :** créer le programme principal permettant d'effectuer ceci :

Créer le client ci-dessous :

* nom : Gravouil
	+ prénom : Benjamin
	+ genre : 1
	+ date de naissance : 02/05/1977
	+ catégorie : enseignant
	+ adresse : lycée Paul Lapie, 5 boulevard AB, 92400 COURBEVOIE
	+ téléphone : 01.01.01.01.01
	+ email : prof.gravouil@gmail.com
* Créer ses comptes (à gérer dans une liste) :
	+ n° : 13245411324 - solde : 2548200 €
	+ **Compte Epargne n° : 32544771289 - solde : 1000 € - taux 5%**
	+ n° : 22544771289 - solde : -214 €
* Afficher la liste des comptes du client

**Question 14 :** dans le programme principal, ajouter les intérêts du compte Epargne.

# Les méthodes abstraites

Nous obtenons les précisions suivantes sur notre domaine d'étude :+

* un compte bancaire est soit un compte courant, soit un compte épargne.
* un compte courant se caractérise par le numéro de la carte bancaire qui lui est associée, ainsi que par un découvert maximal autorisé. Tout retrait qui ferait passer le nouveau solde en dessous du découvert maximal est interdit et non effectué.
* on ne peut retirer en une seule fois plus de la moitié du solde d'un compte épargne.

Un compte bancaire est soit un compte courant, soit un compte épargne. Un compte bancaire en général n'a pas d'existence concrète. La classe CompteBancaire est une abstraction destinée à factoriser ce qui est commun à tous les comptes, mais pas à être instanciée.

en ALGO

**ABSTRACT** CLASS CompteBancaire

{

…

}

en JAVA

CLASSE **ABSTRAITE** CompteBancaire

début

…

fin

**Question 15 :** Faire le nouveau diagramme de classe, avec la classe CompteCourant

**TD7 : Cas Banque (Héritage)**