U.B.S. - I.U.T. de Vannes

Département Informatique

TD2
(Partie GL)
R2.01
Développement Orienté Objets
BUT-Info-1A

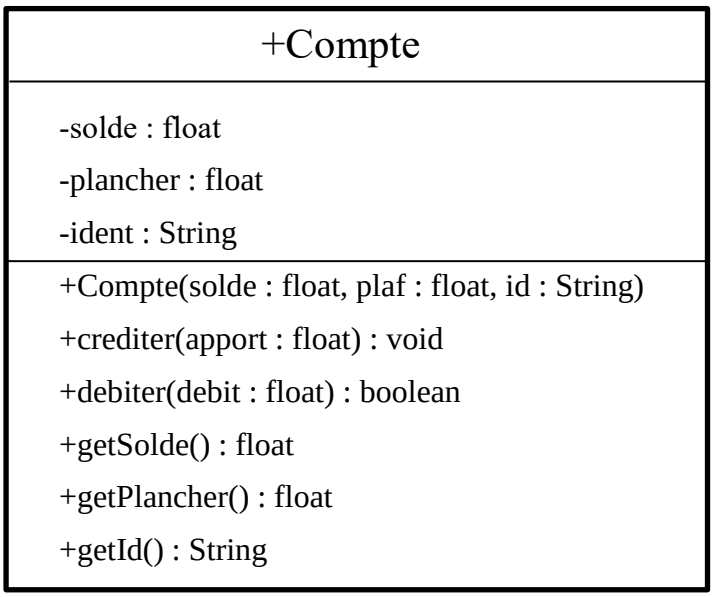
I. Borne, R. Fleurquin, L. Naert, B. Le Trionnaire

Février 2023

Exercice 1

Question 1

On considère une classe compte bancaire qui est caractérisée par un identifiant (numéro de compte dans l'attribut `ident`) de type chaîne de caractères.



Le code Java du constructeur de la classe `Compte` est donné ci-dessous :

```
public Compte ( float solde, float plaf, String id ) {
    this.solde = solde;
    this.plancher = plaf;
    this.ident = id;
}
```

1. Expliquez pourquoi il n'est pas nécessaire de réaliser une copie défensive de la chaîne pointée ni dans le paramètre du constructeur ni dans le retour de la méthode `getId()`. Rappelez les caractéristiques et avantages des objets *immuables*.
2. Identifiez dans le code ci-dessous les 3 modes de création d'objets de type `String`. En analysant le résultat de l'exécution ci-dessous que pouvez-vous déduire sur le traitement des chaînes par Java ?
3. Décrivez les effets du code objet ci-dessous s'appliquant à des objets instances de la classe `Compte`. Pour chaque instruction, dessinez clairement les variables de type *référence* et les *objets* sur lesquels ces références pointent.
4. Dans quel contexte ce code peut-il être apparaître pour pouvoir être compilé sans erreurs ?

```
Compte c1, c2, c3 ;
String id1, id2, id3, id4, id5, id6;


id1 = "num1";
System.out.println(id1);


id2 = "num2";
```

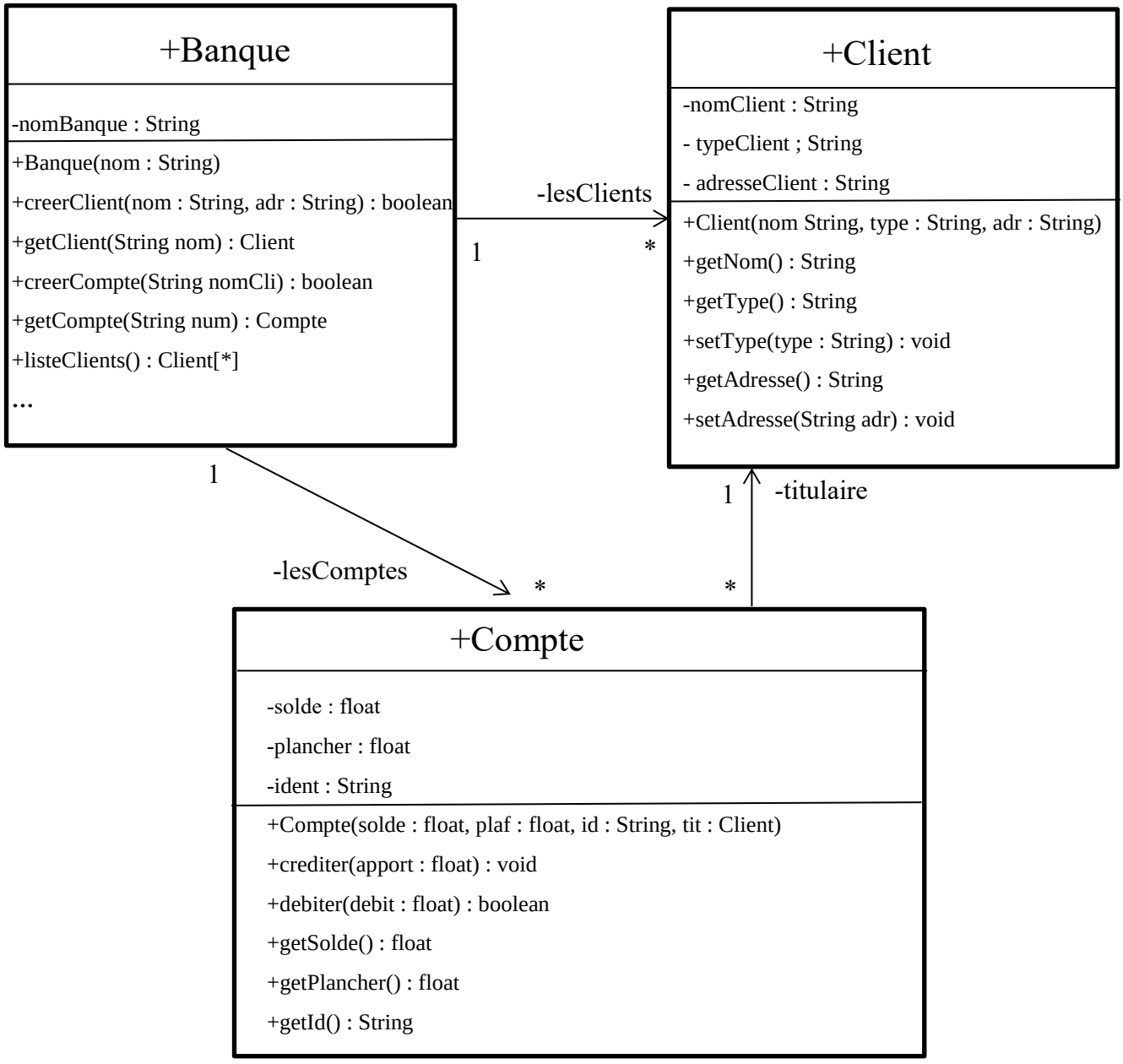
```
System.out.println(id2);

id3 = new String ( "num1" );
System.out.println(id3);

id4 = "num1";
System.out.println(id4);

id5 = "num00";
System.out.println(id5);

id6= id5;
id5=id5+"";

if (id1==id2){System.out.println("id1 et id2 Meme objet");}
else{System.out.println("id1 et id2 deux objets differents");}
if (id1.equals(id2)){System.out.println("id1 et id2 Meme chaine");}
else{System.out.println("id1 et id2 deux chaines differentes");}

if (id1==id3){System.out.println("id1 et id3 Meme objet");}
else{System.out.println("id1 et id3 deux objets differents");}
if (id1.equals(id3)){System.out.println("id1 et id3 Meme chaine");}
else{System.out.println("id1 et id3 deux chaines differentes");}

if (id1==id4){System.out.println("id1 et id4 Meme objet");}
else{System.out.println("id1 et id4 deux objets differents");}
if (id1.equals(id4)){System.out.println("id1 et id4 Meme chaine");}
else{System.out.println("id1 et id4 deux chaines differentes");}

c1 = new Compte ( 400, -100, "num1" );
if (id1==c1.ident){System.out.println("id1 et c1.ident Meme objet");}
else{System.out.println("id1 et c1.ident deux objets differents");}
if (id1.equals(c1.ident)){System.out.println("id1 et c1.ident Meme chaine");}
else{System.out.println("id1 et c1.ident deux chaines differentes");}

c2 = new Compte ( 500, -20, id5);
if (id6==c2.ident){System.out.println("id6 et c2.ident Meme objet");}
else{System.out.println("id6 et c2.ident deux objets differents");}
if (id6.equals(c2.ident)){System.out.println("id6 et c2.ident Meme chaine");}
else{System.out.println("id6 et c2.ident deux chaines differentes");}

c1 = new Compte ( 300, -30, "num00" );
if (id6==c1.ident){System.out.println("id6 et c1.ident Meme objet");}
else{System.out.println("id6 et c1.ident deux objets differents");}

id6=id6+"0";
c3 = new Compte ( 200, -10, "num000" );
```

```
        if (id6==c3.ident){System.out.println("id6 et c3.ident Meme objet");}
        else{System.out.println("id6 et c3.ident deux objets differents");}
        if (id6.equals(c3.ident)){System.out.println("id6 et c3.ident Meme chaine");}
        else{System.out.println("id6 et c3.ident deux chaines differentes");}

        id1 = null;
        id4=null;
        id5=null;
        c3.ident = c2.ident;
        c1.ident = new String ( "num000" );
        id2 = null;

-------------------------------------------------Le résultat de l'exécution
num1
num2
num1
num1
num00
id1 et id2 deux objets differents
id1 et id2 deux chaines differentes
id1 et id3 deux objets differents
id1 et id3 Meme chaine
id1 et id4 Meme objet
id1 et id4 Meme chaine
id1 et c1.ident Meme objet
id1 et c1.ident Meme chaine
id6 et c2.ident deux objets differents
id6 et c2.ident Meme chaine
id6 et c1.ident Meme objet
id6 et c3.ident deux objets differents
id6 et c3.ident Meme chaine
```

Question 2

On étend l'application précédente avec deux nouvelles classes (diagramme de classes UML ci-dessous) : une classe `Banque` qui gère des `clients` et des comptes. Un compte n'a qu'un et un seul titulaire mais un client peut avoir plusieurs comptes. Les clients sont identifiés de manière unique par leur nom et les comptes par leur identifiant. Dans ce domaine métier, une banque est seule habilitée à gérer ses clients et ses comptes.

1. Interprétez la sémantique de ce diagramme de classes ; en particulier des associations avec navigation.



2. Pourquoi est-il souhaitable de ne pas implanter le stockage des comptes et des clients dans des tableaux ? Donnez l'entête des 3 classes en supposant que l'on use pour ces deux stockages du type `ArrayList`.
3. Donnez en conséquence le code du constructeur de la classe `Banque`.
4. Un jeune développeur stagiaire de l'IUT de Vannes s'est vu confier le codage de la classe `Banque`. Il a produit le code qui suit pour la méthode `creerClient()`. Essayez de le comprendre. En déduire la pré-condition et la post-condition de cette méthode « en l'état » de ce code. Un développeur sénior qui inspecte ce code affirme qu'il pose un grave problème (un indice, on a travaillé sur la factorisation des chaines de caractères en mémoire 😉). Lequel ? Illustrez ce problème à l'aide d'un `main()` de test qui montre pourquoi un test unitaire peut difficilement repérer l'erreur grossière commise. Corrigez ce code pour supprimer ce problème et testez le.

```
public boolean creerClient(String nom, String adr) {
	boolean ret =false;
	boolean trouve=false;
	Client c;
	if ((nom!=null) && (adr!=null)){
		Iterator<Client> itr=this.lesClients.iterator();//getting the Iterator
		while ((itr.hasNext()) && (!trouve) ) {//check if iterator has the elements
			c=itr.next();
			if (c.getNom()==nom){
				trouve=true;
			}
		}
		if (!trouve){
			this.lesClients.add(new Client(nom, "stantard", adr));
			ret=true;
		}
		else{
			System.out.println("le client existe deja : pas de creation");
		}
	}
	else{
		System.out.println("Vous devez passer un nom et une adresse valide");
	}
    return ret;
  }
```

5. Le même jeune développeur propose le code qui suit pour la méthode pour la méthode `getClient()`. Ce code fonctionne-t-il correctement ? Corrigez le problème. Il pose là encore d'après le développeur sénior un autre grave problème de sécurité. Lequel ? Illustrez ce problème avec un code de test. Corrigez ce problème en mettant en avant l'intérêt de disposer d'un « copy constructor » dans la classe `Client`. Pourquoi ne peut-on user en l'état d'un simple clonage et donc de l'instruction `ret=(Client) c.clone()`.

```
public Client getClient(String nom) {
	Client ret=null;
	Client c;
	boolean trouve=false;

	if (nom !=null){
		Iterator<Client> itr=this.lesClients.iterator();//getting the Iterator
		while ((itr.hasNext()) && (!trouve)) {//check if iterator has the elements
			c=itr.next();
```

```
                    if (c.getNom()==nom){
                           ret=c;
                           trouve=true;
                    }
                }
                if (!trouve){
                       System.out.println("Desole ce n'est pas un client de notre banque");
                }
             }
        else{
               System.out.println("Desole il faut passer un nom pour mener une recherche");
        }
    return ret;
  }
```

Voici le code de la méthode `listeClients()` écrit par notre jeune développeur. Il semble très fier de lui mais le développeur sénior lui affirme encore que son code est dangereux et qu'il n'a pas dû bien écouter les enseignements de M. Fleurquin. Quel est son erreur ? Donnez un code pour tester que la copie effectuée de l'`ArrayList` est de type « shallow copy » et non « deep copy ». Que faudrait-il faire pour remédier à ce problème ?

```
public Client[] listeClients(){
        Client[] ret;
        ret= new Client[this.lesClients.size()];
        ret = this.lesClients.toArray(ret);
        return ret;
   }
```

Exercice 2

Soit le programme ci-dessous, donnez l'état du *Root Set of Reference* de l'application et des objets en cours d'exécution après chaque instruction. En déduire les moments à partir desquels peuvent être détruits les objets par le ramasse-miettes.

```
class A {
    double d;
    A ( double v ) { this.d = v; }
    double get() { return this.d; }
}

class B {
    A unA;
    int val;
    public static void main ( String[] args ) {
           boolean comp;
           B b1 = new B();
           B b2;
           b1.setVal(10);
           b2 = b1;
           b2.setVal(20);
           b1 = new B();
           b1.setVal(30);
           comp = b1.compare(b2);
           b2 = null;
           b1.setVal(20);
    }

    B( ) { this.unA = new A ( 10.0 ); }

    boolean compare ( B unB ) {
           int a, b;
           double c, d;
           A inutile = new A(5);

           a = this.val;
           b = unB.val;
           c = this.unA.get();
           inutile = unB.unA ;
           unB = null ;
           d = inutile.get() ;
           return ( (a==b) && (c==d) );
    }

    void setVal ( int v ) { this.val = v; }
}
```

Exercice 3

Question 1

Soit la classe « Duree » ci-dessous. La variable « leTemps » mémorise un intervalle de temps qui s'exprime en millisecondes.

Duree
-leTemps : int
+Duree (millisec : int) +Duree (heures : int, minutes : int, secondes : int) +Duree (autreD : Duree) +ajoute (autreD : Duree) +getLeTemps() : int +compareA (autreD : Duree) : int

A. Cette classe contient une donnée et des méthodes. Placer les champs de la classe *Duree* dans une des 4 catégories de champs qui constituent une classe en Java : attributs, constructeurs, accesseurs (éventuellement getter), modificateurs (éventuellement setter). Expliquer, en une ligne et en français, le rôle de chacun de ces champs spécifiquement pour la classe *Duree*.

B. Donner la représentation UML « objets » d'une instance de la classe *Duree*.

C. Donner le code Java de chacune des méthodes ci-dessus. Faire obligatoirement utilisation, dans chaque méthode, du mot-clé « this » (pour information, la méthode « compareA(…) » renvoie -1, 0 ou 1 si l'objet courant est respectivement + petit, égale ou + grand que la durée passée en paramètre).

D. Soit le code Java, ci-dessous, écrit dans un lanceur qui ne fait pas partie de la classe *Duree*. Quelles sont les instructions qui seront rejetées par le compilateur ? Pourquoi ?

```
Duree d1 = new Duree ( 35000 ) ;
Duree d2 = new Duree ( 0, 0, 56 ) ;
Duree d3 = new Duree ( d2 ) ;
d1.leTemps = 47000 ;
int d4 = d1.getLeTemps() ;
d1.ajoute ( d4 ) ;
d1.ajoute ( d3 ) ;
System.out.println ( "La durée d1 est : " + d1 ) ;
System.out.println ( "La durée d1 est : " + d1.getLeTemps() ) ;
```

Question 2

Soit le diagramme de classes de conception UML en annexe1.

Ce diagramme de classes modélise les données que l'on peut trouver sur un support de type CD de musique. En effet, un CD est composé de plages musicales et chaque plage musicale est caractérisée par une durée, le titre du morceau et l'interprète du morceau.

A) Pour la classe Plage et la classe CD, donner la catégorie à laquelle appartient chaque champ de la classe : attribut, constructeur, accesseur (éventuellement getter), modificateur (éventuellement setter).

B) A partir du diagramme de classes, expliquer :

- le sens des flèches,
- les cardinalités,
- les rôles,
- la différence entre les flèches continues et les flèches pointillées.

C) Donner la représentation UML « objets », d'une instance de la classe « Plage ». A partir de l'attribut « laDuree » de la classe « Plage », quelles sont les champs de la classe « Duree » auxquels il est possible d'accéder ?

D) Donner le code Java de déclaration de l'attribut « lesPlages » de la classe « CD ». Sachant qu'un objet de type « CD » pourra stocker 17 plages musicales, donner le code Java du constructeur de la classe « CD » (on suppose que l'allocation du tableau se fait dans le constructeur et que l'ajout des plages dans ce tableau se fait en appelant ensuite la méthode « graverCD() »).

E) Donner le code Java des méthodes « getUnePlage (…) » et « getDureeTotale() » de la classe « CD ». A partir de l'objet « Plage » renvoyé par la méthode « getUnePlage (…) », quelles sont les champs de la classe « Plage » auxquels il est possible d'accéder ?

Question 3

Soit, ci-dessous, le code Java de la méthode « graverCD() » de la classe « CD ». Cette méthode est appelée par le constructeur. Ce code compile sans erreurs. Répondre aux questions écrites en commentaire dans cette méthode.

```
private void graverCD () {
      Duree  uneD ;
      // 1. Quel est le contenu de la variable uneD ?
      int cpt ;
      // 2. Quel est le contenu de la variable cpt ?
      uneD = new Duree ( 0, 3, 30 ) ;
      // 3. Quel est le contenu de l'attribut « leTemps » de l'objet « uneD » ?
      Plage  p1 = new Plage ( uneD, "Ca plane", "Pour moi" ) ;
      // 4. L'instruction p1.leTitre = new String ("La danse des canards") ; est-elle possible ?
      Plage  p2 = new Plage ( uneD, "Ca plane", "Pour moi" ) ;
      // 5. Quel est le résultat de la comparaison : p1 == p2 ? Pourquoi ?

      for ( cpt = 0 ; cpt < this.lesPlages.length ; cpt++ ) {
            this.lesPlages[cpt] = p1 ;
      }
      // 6. Quel est le contenu exact du tableau après exécution de la boucle « for » ?

      Plage uneP ;
      uneP = this.lesPlages[16] ;

      if ( uneP == p1 ) { System.out.println ( "Egalité des références" ) ; }
      else { System.out.println ( "Inégalité des références" ) ; }
      // 7. Qu'est-il affiché ? Pourquoi ?

      int t1 = ( ( this.lesPlages[2] ).getLaDuree() ).getLeTemps() ;
      // 8. Que contient la variable t1 ?

      int t2 = ( p2.getLaDuree() ). getLeTemps() ;
      // 9. Que contient la variable t2 ?

      // 10. Quel est le résultat de la comparaison : t1 > t2 ?
}
```

Annexe 1 : diagramme de classes de conception « CD de musique »