# Classes et objets

---

**Exercice 1** *Compte bancaire et compte bancaire rémunéré*

---

On souhaite mettre en place des classes représentant des comptes bancaires. Les comptes bancaires ont un numéro de compte (qui ne varie pas au cours du temps) et un solde, et on peut y déposer ou en prélever un certain montant (le prélèvement n'est possible que si le solde est suffisant, dans le cas contraire le prélèvement échoue). Les comptes bancaires rémunérés sont des comptes bancaires pour lequel le solde est rémunéré d'un certain pourcentage, qui est le même pour tous les comptes rémunérés. Les comptes rémunérés sont plafonnés : chaque compte a un plafond de versement qui ne peut pas être dépassé. En revanche, les intérêts peuvent venir dépasser ce seuil. Les intérêts sont calculés grâce à une méthode spécifique, et viennent augmenter le solde (cette méthode sera appelée mensuellement, mais vous ne gérerez pas cet appel).

**Question 1.** Modélisez les classes pour les comptes bancaires, en y plaçant : les attributs, les constructeurs, les accesseurs et les méthodes nécessaires.

**Question 2.** Proposez une implémentation de ces classes, et une classe en permettant le test.

**Exercice 2** *Documents*

On s'intéresse à la modélisation partielle de documents textuels. Un document a un titre et un texte. Il existe deux sortes de documents particuliers : les documents avec mentions légales, et les documents cryptés. Les documents avec mentions légales disposent d'un attribut de type stockant les dites mentions légales (dans une chaîne de caractères). Les documents cryptés sont munis d'une méthode de classe crypter prenant en paramètre une chaîne de caractères à crypter et retournant la chaîne de caractères cryptée. Pour tous les documents, on souhaite disposer :

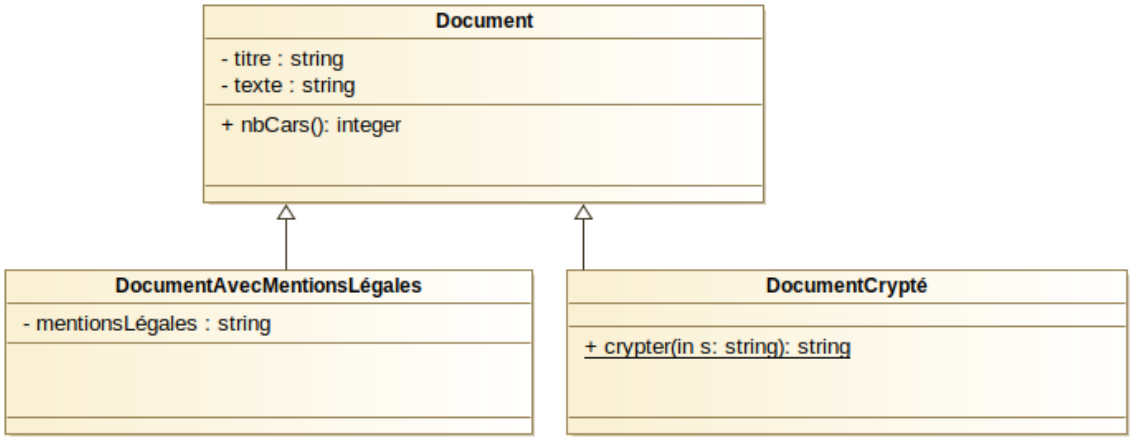


FIGURE 1 – Diagramme de classes à compléter

— d'une méthode `nbCars` qui compte le nombre de caractères du document (somme du nombre de caractères du titre et du nombre de caractères du texte),
— et une méthode `export` d'export du document qui retourne une version imprimable du document. Pour simplifier, ici, la version imprimable sera une chaîne de caractères qui est la concaténation du titre du document et du texte du document (il aurait été plus logique d'exporter dans un fichier de type pdf mais c'est plus compliqué !). Pour les documents avec mentions légales, la version imprimable se termine par les mentions légales (en plus du titre et du texte). Pour les documents cryptés, la version imprimable est cryptée (et devra donc être décryptée par un utilisateur avant impression).

On donne le diagramme de classes partiel de la figure 1.

**Question 1.** Représentez sur le diagramme de la figure 1 le fait qu'il peut exister d'autres sortes de documents spécifiques que les documents avec mentions légales et les documents cryptés, et qu'il peut exister des documents à la fois cryptés et avec mentions légales.

**Question 2.** Proposez un diagramme d'objets représentant un document avec mentions légales, en utilisant les valeurs de votre choix.

**Question 3.** Complétez le diagramme de la figure 1 pour y faire apparaître la méthode `export`, partout où elle est nécessaire.

**Question 4.** Donnez en Java une implémentation du modèle obtenu, en y ajoutant les cpnstructeurs qui vous paraissent pertinents, et en prenant en compte les éléments suivants :
— Pour la méthode `nbCars`, sachez que la classe `String` de Java dispose d'une méthode `length()` qui retourne sous forme d'entier le nombre de caractères de la chaîne de caractères.
— La méthode `crypter` ne réalisera pas un vrai cryptage mais concatènera un texte quelconque.

**Question 5.** L'export change : il est décidé de retourner l'export sous forme de markdown. Le titre sera donc précédé d'un # et suivi d'un retour à la ligne. Mettez en place cette modification, constatez que vous n'avez à la faire qu'une seule fois.

**Exercice 3** *Analyse de hiérarchies de classes*

**Question 1.** Etudiez la hiérarchie de classe donnée au code 1, et pour chaque ligne de la méthode principale, dites si elle compile et si oui quel est son comportement. Toute ligne ne compilant pas sera supposée commentée vis à vis des lignes suivantes.

Code 1 – Une première hiérarchie de classe

```
public class A {
        public void m0() {}
        public void m1() {}
}

public class B extends A {
        public void m1() {}
        public void m2() {}
        public void m3() {}
}

public class C extends A {
        public void m1() {}
        public void m3() {}
}

public class ManipHierarchie1 {
        public static void main(String[] args) {
                A a=new B();
                B b=new A();
                a=new A();
                a.m1();
                a=new B();
                a.m1();
                a.m2();
                a.m3();
        }
}
```

Code 2 – Une deuxième hiérarchie de classe

```
public class A {
        public A(int i){}
        public A m0(A a) {}
        public void m1(A a) {}
}

public class B extends A {
}

public class C extends A {
}
```

**Question 2.** La hiérarchie du code 2, ne compile pas. Expliquez pourquoi et proposez une solution.

**Question 3.** Ajoutez à la classe B une redéfinition invariante de la méthode m1 qui en masque le comportement.

**Question 4.** Ajoutez à la classe B une redéfinition covariante sur le type de retour de la méthode m0 qui en spécialise le comportement.

**Question 5.** Ajoutez à la classe C une surcharge de la méthode m0, sans modifier le nombre de ses paramètres ni son type de retour.

**Question 6.** Une fois les questions précédentes réalisées, donnez ligne par ligne le comportement du programme suivant :

Code 3 – Un programme utilisant la deuxième hiérarchie de classe complétée

```
A b=new B(1);
A c=new C(1);
b.m0(new A(1));
b.m0(new B(1));
c.m0(new A(1));
c.m0(new B(1));
```