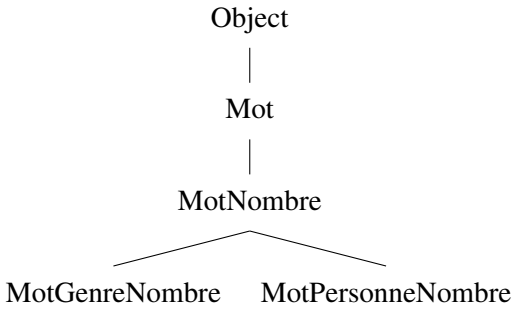
Corrigés 7

Exercices sur le typage et la surcharge

1 Exercice de typage

On reprend les classes du cours : `Mot, MotNombre, MotGenreNombre` et `MotPersonneNombre` avec l'arborescence suivante.



Voici un morceau de code avec des inconnues (XXX, YYY, ZZZ et TTT) à remplacer.

```
public static void main(String[] args){
    XXX mot1 = new YYY;
    ZZZ mot2 = new TTT;
    System.out.println(mot1.getForme());
    System.out.println(mot2.getForme());
    System.out.println(mot2.estSingulier());
}
```

Donnez toutes les combinaisons de classes possibles pour XXX, YYY, ZZZ et TTT (vous ne vous préoccuperez pas des paramètres des constructeurs pour YYY et TTT).

Ce qui va apporter des contraintes sur les types des variables mot1 et mot2, ce sont les invocations de méthode. Sur mot1, seule la méthode getForme est invoquée. Sur mot2, les deux méthodes getForme et estSingulier sont invoquées. Pour mot1, les classes Mot et ses descendantes sont des types possibles car ces classes ont toutes la méthode getForme. Pour mot2, la classe MotNombre et ses descendantes ont les deux méthodes invoquées.

Ensuite, pour chacun des types de variables, on peut affecter à la variable une instance de cette classe ou une instance d'une descendante de cette classe.

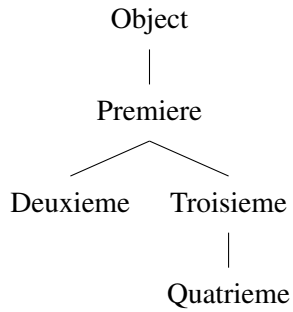
XXX	YYY
Mot	Mot MotNombre MotGenreNombre MotPersonneNombre
MotNombre	MotNombre MotGenreNombre MotPersonneNombre
MotGenreNombre	MotGenreNombre
MotPersonneNombre	MotPersonneNombre

ZZZ	TTT
MotNombre	MotNombre MotGenreNombre MotPersonneNombre
MotGenreNombre	MotGenreNombre
MotPersonneNombre	MotPersonneNombre

2 Exercice sur les constructeurs

Question 1

1. dessinez le graphe d'héritage de ce programme, retraçant les relations d'héritage entre classes sans oublier `Object`, la super-classe de `Premiere`.



2. prédisez ce que ce programme affiche.
3. compilez et exécutez ce programme pour vérifier vos prédictions.

```
constructeur de Premiere
============================
constructeur de Premiere
constructeur de Seconde
============================
constructeur de Premiere
constructeur de Troisieme
============================
constructeur de Premiere
constructeur de Troisieme
constructeur de Quatrieme
```

4. y a-t-il un constructeur qui s'exécute sans afficher de message à la création de certains objets ?

   Oui, le constructeur sans paramètre de la classe `Object` s'exécute avant tous les autres sans afficher de message. Comme cette classe est prédéfinie, on ne peut pas modifier le constructeur pour qu'un message s'affiche.

Question 2

1. prédisez ce que ce programme affiche.
2. compilez et exécutez ce programme pour vérifier vos prédictions.

```
constructeur de Premiere
constructeur de Cinquieme
============================
constructeur de Premiere
constructeur de Cinquieme
constructeur de Sixieme
```

Quand il n'y a pas d'appel explicite à un constructeur de la super-classe, le constructeur sans paramètre de la super-classe est appelé.

Question 3

Le programme suivant provoque une erreur à la compilation : la classe Huitieme est incorrecte.

```
class Septieme extends Premiere{
    Septieme(int i){
        Terminal.ecrireStringln("constructeur␣de␣Septieme");
    }
}
class Huitieme extends Septieme{
    Huitieme(){
        Terminal.ecrireStringln("constructeur␣de␣Huitieme");
    }
}
```

Essayez de trouver l'erreur. Si vous n'y arrivez pas, essayez de compiler ce programme : le message d'erreur peut vous mettre sur la voie.

```
> javac Exo19_1_3.java
Exo19_1_3.java:7: cannot find symbol
symbol  : constructor Septieme()
location: class Septieme
    Huitieme(){
             ^
1 error
```

Le constructeur de `Huitieme` ne fait pas d'appel explicite à un constructeur de sa super-classe `Septieme`. Dans ce cas, il y a un appel implicite au constructeur sans paramètre de `Septieme`, or ce constructeur n'existe pas.

Question 4

1. prédisez ce que ce programme affiche.
2. compilez et exécutez ce programme pour vérifier vos prédictions.

```
constructeur de Premiere
second constructeur de Neuvieme
premier constructeur de Dixieme
============================
constructeur de Premiere
premier constructeur de Neuvieme
second constructeur de Dixieme
```

3 Exercice d'héritage : volaille

Un éleveur de volaille reçoit d'un fournisseur de jeunes canards et de jeunes poulets qu'il élève jusqu'à ce qu'ils aient la taille nécessaire à leur commercialisation.

Une volaille est caractérisée par son poids et un numéro d'identification reporté sur une bague qu'elle porte a sa petite patte. Les volailles arrivent à l'élevage à l'âge de trois semaines. Elles sont baguées et enregistrées dans le système informatique.

Il y a deux sortes de volailles : des canards et des poulets. Le prix du canard et celui du poulet sont deux prix différents, exprimés en euros par kilo. En revanche, le prix est le même pour tous les individus de la même espèce. Ce prix varie chaque jour. Le poids auquel on abat les bêtes est différents pour les canards et les poulets, mais c'est le même pour tous les poulets (respectivement, tous les canards).

Ecrivez une classe des volailles avec deux sous-classes des poulets et des canards. Il faut pouvoir enregistrer les prix du jour, les poids d'abattage, le poids d'une volaille donnée.

Écrivez une classe permettant de représenter l'ensemble des animaux de l'élevage au moyen d'un tableau. Des méthodes doivent permettrent de trier les animaux à abattre et d'évaluer le prix obtenu pour ces animaux. Il faut également pouvoir enregistrer les jeunes animaux qui arrivent.

```
class Volaille{
    protected double poids;
    protected int identite;
    public Volaille(double p, int i){
        poids = p;
        identite = i;
    }
    public void changerPoids(double np){
        poids = np;
    }
    public double prix(){
        return 0.0;
    }
    public boolean assezGrosse(){
        return false;
    }
    public static void changerPrix(double x){
    }
```

```
    public static void changerPoidsAbattage(double x){
    }
    public String toString(){
        return identite + "␣poids:␣";
    }
}
class Poulet extends Volaille{
    static double prixAuKilo = 1.0;
    static double poidsAbattage = 1.2;
    public Poulet(double p, int i){
        super(p,i);
    }
    public static void changerPrix(double x){
        prixAuKilo = x;
    }
    public static void changerPoidsAbattage(double x){
        poidsAbattage = x;
    }
    public double prix(){
        return poids *prixAuKilo;
    }
    public boolean assezGrosse(){
        return poids >= poidsAbattage;
    }
    public String toString(){
        return "Poulet␣" + identite + "␣poids:␣" + poids;
    }
}
class Canard extends Volaille{
    protected static double prixAuKilo = 1.2;
    protected static double poidsAbattage = 1.5;
    public Canard(double p, int i){
        super(p,i);
    }
    public static void changePrix(double x){
        prixAuKilo = x;
    }
    public static void changePoidsAbattage(double x){
        poidsAbattage = x;
    }
    public double prix(){
        return poids *prixAuKilo;
    }
    public boolean assezGrosse(){
        return poids >= poidsAbattage;
    }
    public String toString(){
        return "Canard␣" + identite + "␣poids:␣" + poids;
    }
}
class Elevage{
    protected Volaille[] tab = new Volaille[100];
    protected int nbBetes = 0;
```

```
    public void ajouter(Volaille v){
        tab[nbBetes] = v;
        nbBetes++;
    }
    public Volaille rechercher(int id){
        for (int i=0; i<nbBetes; i++){
            if (tab[i].identite == id){
                return tab[i];
            }
        }
        return null;
    }
    public void changerPoids(int id, double np){
        rechercher(id).changerPoids(np);
    }
    public double evaluerBetesAAbattre(){
        double res = 0;
        for (int i=0; i<nbBetes; i++){
            if (tab[i].assezGrosse()){
                res = res+tab[i].prix();
            }
        }
        return res;
    }
    public Volaille[] envoyerALAbattoir(){
        Volaille[] res = new Volaille[100];
        int nb = 0;
        int i = 0;
        while (i < nbBetes){
            if (tab[i].assezGrosse()){
                res[nb] = tab[i];
                nb++;
                tab[i]=tab[nbBetes-1];
                nbBetes--;
            }else{
                i++;
            }
        }
        return res;
    }
    public String toString(){
        String res = "";
        for (int i=0; i<nbBetes; i++){
            res = res + tab[i] +"\n";
        }
        return res;
    }
}
public class MainVolaille{
    public static void main(String[] args){
        Elevage laFerme = new Elevage();
        for (int i=0; i<15; i++){
            laFerme.ajouter(new Poulet(0.250,150+i));
```

```
        }
        for (int i=0; i<15; i++){
            laFerme.ajouter(new Canard(0.250,380+i));
        }
        for (int i=0; i<10; i++){
            laFerme.ajouter(new Poulet(0.250,700+i));
        }
        laFerme.ajouter(new Canard(0.750,825));
        for (int i=0; i<8; i++){
            laFerme.changerPoids(155+i,1.3);
            laFerme.changerPoids(382+i,1.55);
        }
        System.out.print(laFerme.toString());
        System.out.println("Valeur␣des␣animaux␣a␣abattre:␣");
        System.out.println(laFerme.evaluerBetesAAbattre());
        laFerme.envoyerALAbattoir();
        System.out.print(laFerme.toString());
        System.out.println("Valeur␣des␣animaux␣a␣abattre:␣");
        System.out.println(laFerme.evaluerBetesAAbattre());
    }
}
```