Solution TD3

Solution Exercice 1 :

Employe.java

```
public class Employe {

   private String nom;
   private int age;
   private float salaire;

   /*Constructeur pardéfaut*/
   public Employe() {   }
   /*Constructeur avec 3 paramètres*/
   public Employe(String nom, int age, float salaire) {
        this.nom = nom;
        this.age = age;
        this.salaire = salaire;
  }
   /*constructeur par copie*/
  public Employe(Employe oldEmploye) {
    nom = oldEmploye.nom;
    age = oldEmploye.age;
    salaire = oldEmploye.salaire;
  }
   /*mutateurs (modificateurs) */
  public void setNom(String nom) {
            this.nom = nom;
    }

    public void setAge(int age) {
            this.age = age;
    }

    public void setSalaire(float salaire) {
            this.salaire = salaire;
    }
     /*accesseurs  (recupérateurs) */
    public String getNom() {
            return (this.nom);
    }

    public int getAge() {
            return (this.age);
    }

    public float getSalaire() {
            return (this.salaire);
   }
    /*toString*/
   public String toString() {
        return nom+"\t"+age+"\t"+salaire;
  }


  public void afficher() {
      System.out.println(toString());
    }

  public void augmentation(float a) {
    salaire=salaire+a;

  }
```

Technicien.java

```
public class Technicien extends Employe {

   private byte grade;

    public Technicien(String nom, int age, float salaire, byte grade)
   {
      super(nom,age,salaire);
      this.grade=grade;
   }

   public int prime() {

     switch(grade)
      {
        case 1: return 100;
        case 2: return 200;
        case 3: return 300;
      }
      return 0;

   }

   public float calculSalaire()
   {return getSalaire()+prime();}

   public String toString() {

      return super.toString() + "\t" +grade;
   }

}

/*****************************************/
class TestEmpTech
{
   public static void main(String args[])
   {
     Employe e1=new Employe("Albert",28,4500);
     Employe e2=new Technicien("Bernard",50,8000,(byte)5);

     //Technicien e3 =new Employe();  //erreur

     Technicien e3 =new Technicien("Jacques",25,5000,(byte)4);

     System.out.println("Avant augmentation:");
     e1.afficher();
     e2.afficher();
     e3.afficher();

     e1.augmentation(600);
     e2.augmentation(500);
     e3.augmentation(650);

     System.out.println("Apres augmentation:");
     e1.afficher();
     e2.afficher();
     e3.afficher();


   }
}
```

Solution Exercice 2 :

2.1 La classe Vehicule

Définissez une classe **`Vehicule`** qui a pour attributs des informations valables pour tout type de véhicule : sa marque , sa date d'achat, son prix d'achat et son prix courant.

Solution :

```
class Vehicule {

   private String marque;
   private int dateAchat;
   private double prixAchat;
   private double prixCourant;
```

Ces attributs sont **`private`** pour assurer une bonne encapsulation.

Définissez un constructeur prenant en paramètres la marque, la date d'achat et le prix d'achat.

Une solution possible: :

```
    public Vehicule(String marque, int date, double prix) {
        this.marque = marque;
        dateAchat = date;
        prixAchat = prix;
        prixCourant = prix;
    }
```

Définissez une méthode publique **`affiche()`** qui affiche la valeur des attributs.

```
    public void affiche() {
        System.out.print("marque : " + marque +
                ", date d'achat : " + dateAchat +
                ", prix actuel : " + prixCourant);
        System.out.println();
    }
```

2.2 La classe Voiture et la classe Avion

Définissez deux classes **`Voiture`** et **`Avion`**, héritant de la classe **`Vehicule`** et ayant les attributs supplémentaires suivants :

Commençons par la classe **`Voiture`**. Elle doit hériter de la classe **`Vehicule`** :

```
class Voiture extends Vehicule {
```

On ajoute ensuite les champs spécifiques à la classe **`Voiture`** :

```
    private double cylindree;
    private int nbPortes;
    private double puissance;
    private double kilometrage;
```

Pour la classe **`Avion`**, on procède de même :

```
class Avion extends Vehicule {

    private String moteur;
    private int heuresVol;
```

Définissez maintenant un constructeur pour **`Voiture`**, ainsi qu'une méthode affichant la valeur des attributs.

```
    public Voiture(String marque, int date, double prix,
            double cylindree, int portes, double cv, double km) {
        super(marque, date, prix);
        this.cylindree = cylindree;
        nbPortes = portes;
        puissance = cv;
        kilometrage = km;
    }

    public void affiche() {
        System.out.println("---- Voiture ----");
        super.affiche();
        System.out.println(cylindree + " litres, "
                + nbPortes + " portes, "
                + puissance + " CV, "
                + kilometrage + " km.");
    }
```

Ces deux méthodes doivent, bien entendu, être publiques puisqu'elles sont précisément faites pour être utilisées hors de la classe.

Notez que pour le constructeur de **`Voiture`**, on fait appel au constructeur de **`Vehicule`** . On fait également appel à la méthode d'affichage de la super-classe dans la méthode **`affiche`** de **`Voiture`**.

Les méthodes de la classe **`Avion`** s'implémentent de même :

```
    public Avion(String marque, int date, double prix, String moteur, int
heures) {
        super(marque, date, prix);
        this.moteur = moteur;
        heuresVol = heures;
    }

    public void affiche() {
        System.out.println("---- Avion à " + moteur + "----");
        super.affiche();
        System.out.println(heuresVol + " heures de vol.");
    }
```

Encore des méthodes

Ajoutez une méthode **`void calculePrix()`** dans la classe **`Vehicule`**
Le prix doit rester positif (donc s'il est négatif, on le met à 0).

```
public void calculePrix(int anneActuelle) {
    double decote = (anneActuelle - dateAchat) * 0.01;
    prixCourant = Math.max(0.0, (1.0 - decote) * prixAchat);
}
```

Redéfinissez cette méthode dans les deux sous-classes **`Voiture`** et **`Avion`**.

Les entêtes sont les mêmes pour **`Voiture`** et **`Avion`** que pour **`Vehicule`**, par contre les définitions diffèrent.

Voiture:

```
public void calculePrix(int anneActuelle) {
    double decote = (anneActuelle - getDateAchat()) * .02;
    // On force le type en int de manière  arrondir le résultat
    // On verra dans quelques semaines une manière plus élégante de faire
    // ce genre de choses...
    decote += 0.05 * (int)(kilometrage / 10000);
    if ((getMarque() == "Fiat") || (getMarque() == "Renault")) {
      decote += 0.1;
    } else if ((getMarque() == "Ferrari") || (getMarque() == "Porsche")) {
      decote -= 0.2;
    }
    setPrixCourant(Math.max(0.0, (1.0 - decote) * getPrixAchat()));
}
```

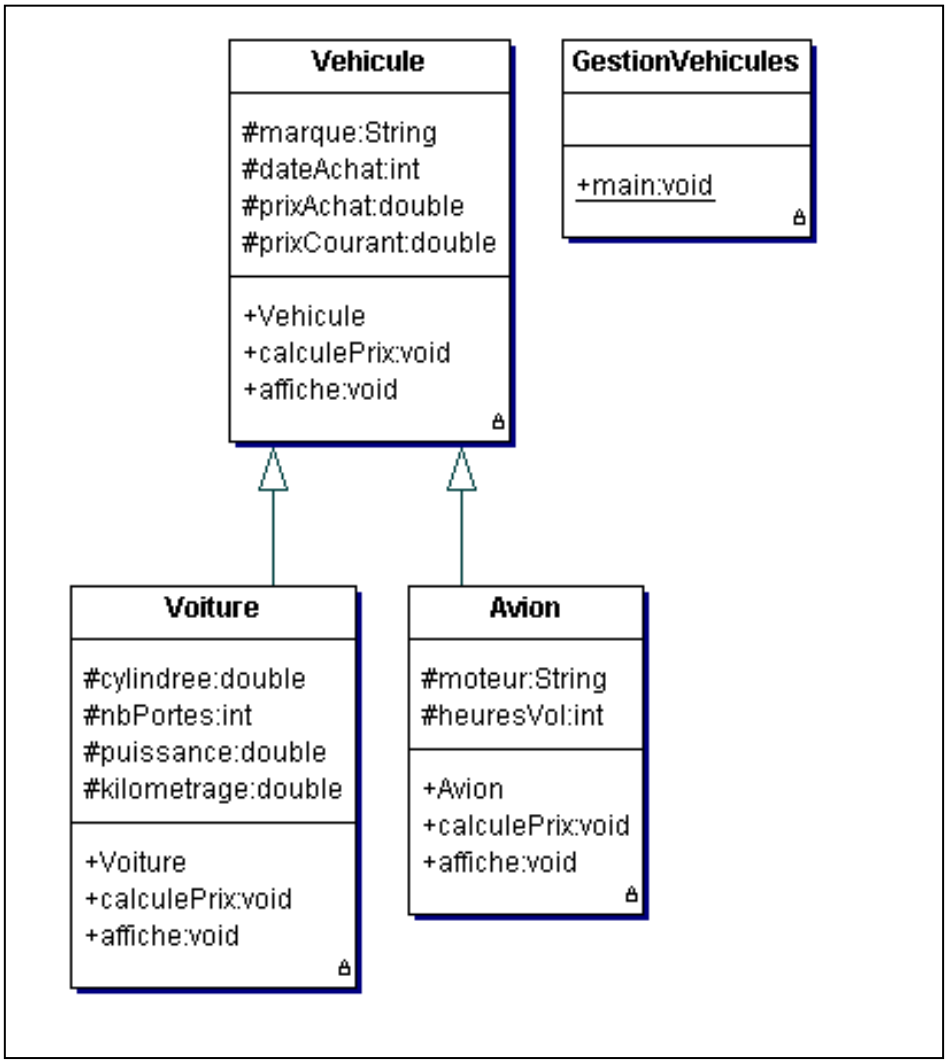
On se rend compte ici que l'on a besoin d'accèder à des attributs privés de la super-classe, ou de les modifier (**`dateAchat`**, **`marque`**, **`prixAchat`** et **`prixCourant`**). Il faut donc enrichir la classe **`Vehicule`**des "getters/setters" nécessaires :

```
public int getDateAchat(){
  return dateAchat;
}
public String getMarque(){
  return marque;
}
public double getPrixAchat(){
  return prixAchat;
}
public void setPrixCourant(double prix){
  prixCourant = prix;
}
```

Il aurait été possible de déclarer les attributs nécessaires comme **`protected`** dans la classe **`Vehicule`** pour s'éviter la peine de définir les getters/setters. Ceci peut cependant nuire à une bonne encapsulation : un autre programmeur peut hériter de votre classe **`Vehicule`**. Il aurait alors accès aux détails d'implémentation et vous ne pourriez

plus modifier librement cette implémentation sans potentiellement causer du tort à ses programmes ! Voici enfin la méthode **`calculePrix`** pour les avions :

```
public void calculePrix(int anneActuelle) {
    double decote;
    if (moteur == "HELICES") {
      decote = 0.1 * heuresVol / 100.0;
    } else {
      decote = 0.1 * heuresVol / 1000.0;
    }
    setPrixCourant(Math.max(0.0, (1.0 - decote) * getPrixAchat()));
  }
```



Solution Exercice 3 :