TD n°1 : Programmation Orientée Objet, UML

Avant de commencer...

Pour définir un tableau (d'entier par exemple) en Java :

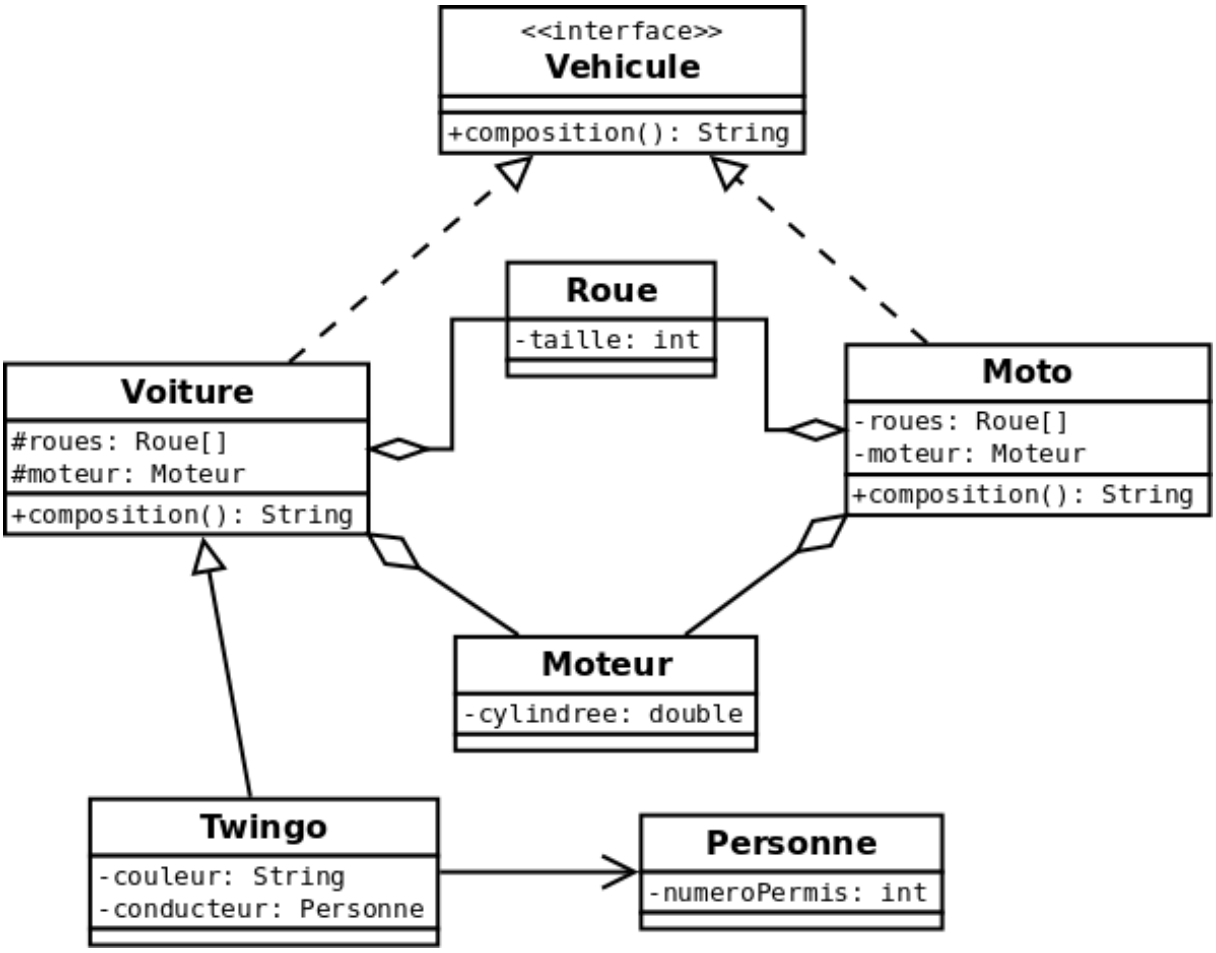
```java
// Definir un tableau d'entier
int tab[]; // Tableau d'une longueur non definie
int[] tab; // Peut s'ecrire aussi comme ca.

// Initialiser un tableau avec des valeurs
int[] tab2 = {1, 2, 3};

// Instancier un tableau (deja declare) de 5 elements
tab = new int[5];
```

Exercice 1 (De l'UML à Java)

Écrire les classes Java correspondant au diagramme UML ci-dessous.



Exercice 2 (Polymorphisme)

Considérons le code Java suivant :

```
import java.io.*;

class A {
  public double operation(int x, double y) {
    return 2 * (x + y);
  }

  public double operation(double x, double y) {
    return x + y;
  }
}

class B extends A {
  public double operation(int x, double y) {
    return x / y;
  }

  public double operation(int x, int y) {
    return x - y;
  }
}

class C extends B {
  public double operation(int x, int y) {
    return x * y;
  }
}

class Main {
  public static void main(String [] args) {
    A a = new C();
    B b = new C();

    /* 1 */ System.out.println( a.operation(2.0, 4.0) );
    /* 2 */ System.out.println( a.operation(2  , 4  ) );
    /* 3 */ System.out.println( a.operation(2  , 4.0) );

    /* 4 */ System.out.println( b.operation(2.0, 4.0) );
    /* 5 */ System.out.println( b.operation(2  , 4  ) );
    /* 6 */ System.out.println( b.operation(2  , 4.0) );
  }
}
```

Écrivez les 6 résultats affichés en justifiant votre réponse. Les détails de votre réponse doivent décrire ce qui se passe à la compilation et lors de l'exécution.

Exercice 3 (De Java à l'UML)

Faire le diagramme de classe UML décrivant les classes Java ci-dessous.

```
interface Animal
{
  public void cri();
  public boolean estCarnivore();
}

class Chien implements Animal
{
  public String nom;
  protected int age;
  private String tatouage_;

  public void cri() { System.out.println("Aboie"); }

  public boolean estCarnivore()
  {
    return true;
  }
}

class Lapin implements Animal
{
  private double taille_;

  public void cri() { System.out.println("Clapit"); }

  public boolean estCarnivore()
  {
    return false;
  }

  void combienDeCarottes(int carottes)
  {
    System.out.println("Mange " + carottes + "carotte(s)");
  }
}

class Levrier extends Chien
{
  protected Nourriture nourritureFavorite;

  public void cri() { System.out.println("Jappe"); }
}

class Nourriture
{
  public String nom;
  int valeurCalorique;
}
```

Exercice 4 (Passage des arguments en Java)

En Java, tout se passe par valeur. Le passage par référence, possible par exemple en C++, n'existe pas ici.

```
public void method(Person p)
{
  if( p.getName().equals("Thomas") )
    System.out.println("test␣#1␣equals");
  else
    System.out.println("test␣#1␣different");

  p = new Person("Leo");

  if( p.getName().equals("Leo") )
    System.out.println("test␣#2␣equals");
  else
    System.out.println("test␣#2␣different");
}

Person person = new Person("Thomas");
method( person );
if( person.getName().equals("Thomas") )
  System.out.println("test␣#3␣equals");
else
  System.out.println("test␣#3␣different");
```

1. Écrire le texte affiché par le code ci-dessus. Justifier votre réponse à l'aide d'un schéma montrant l'état de la mémoire, les "pointeurs", les cases-mémoire et leur valeur, etc.
2. Imaginons que nous soyons dans un univers parallèle où le passage de l'argument dans le code Java ci-dessus se fait par référence. Quel serait l'affichage ? Justifier à l'aide d'un schéma.