Jour 2 - POO - Héritage

HERITAGE ET POLYMORPHSME, une manière de voir les arbres généalogiques

Job 0

Créer un programme java et définir un objet personne. Il doit disposer :

- **D'attributs publics** : **nom**, **prénom**
- **D'attributs protégés** : **date de naissance, lieu de naissance**
- **D'attributs privés** : **adresse**, **téléphone**

Quelles sont les différences de ces différents types d'attributs ?

Pour quels types a-t-on besoin d'accesseurs : getter et setter ?

Job 1

```
class Toto{
  int toto = 0;
  Toto() {
    toto = toto + 1;
  }
  public static void main(String[] tutu) {
    Toto t1 = new Toto();
    Toto t2 = new Toto();
    System.out.println("Toto : " + toto);
  }
}
```

Ce code fonctionne-t-il et, sinon, quelle(s) erreur(s) l'interpréteur va-t-il indiquer ?

Job 2

Le programme Erreur ci-dessous définit les classes E1, E2, E3, E4, E5 et Erreur.

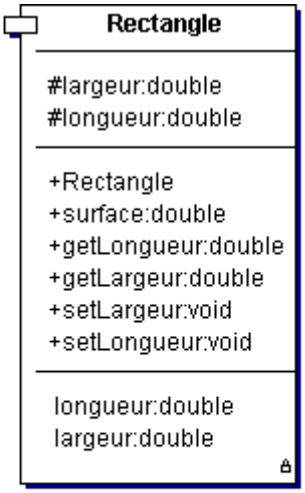
Pour chacune des 5 instructions d'affichage dans la méthode main, indiquez

si l'instruction est correcte ou fausse. Compilez et corrigez le programme pour vérifier vos réponses.

Erreur.java

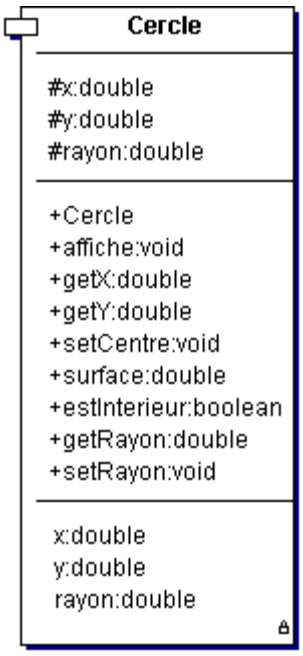
```java
class Erreur {

    public static void main(String args[]) {
        E1 x = new E1();
        E2 y = new E2();
        E3 z = new E3();
        E4 v = new E4();
        E5 w = new E5();
        System.out.println(x.a);   // Correct ou faux ?
        System.out.println(y.c);   // Correct ou faux ?
        System.out.println(z.b);   // Correct ou faux ?
        System.out.println(v.c);   // Correct ou faux ?
        System.out.println(w.a);   // Correct ou faux ?
    }

}

class E1 {
    int a = 1;
}

class E2 extends E1 {
    int b = 2;
}

class E3 extends E2 {
    int c = 3;
}

class E4 extends E1 {
    int d = 4;
}

class E5 extends E4 {
    int e = 5;
}
```

Job 3.0



- Écrire un programme **FiguresGeometriques.java** définissant l'objet **Rectangle** ayant une largeur et une longueur (format double), les getters et les setters correspondants ainsi que la méthode surface.

Job 3.1



Écrire dans le programme **FiguresGeometriques.java** la définition de l'objet **Cercle** ayant un centre avec deux coordonnées X et Y (double) ainsi que le rayon (double), les getters et les setters correspondants ainsi que la méthode surface. On peut définir également une méthode qui prend deux valeurs X et Y pour vérifier si le point est à l'intérieur du cercle ou non.

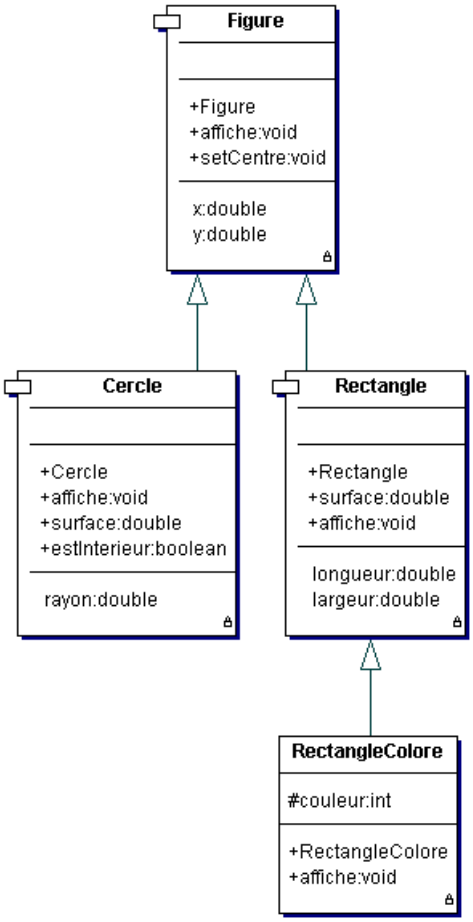
Job 3.2

Ajouter une classe **RectangleColore** qui hérite de **Rectangle**. Cette classe doit simplement avoir un attribut de plus, couleur, de type **int**.
N'oubliez pas de créer un constructeur.

Job 3.3

Ajouter une classe **RectangleColore** qui hérite de **Rectangle**. Cette classe doit simplement avoir un attribut de plus, couleur, de type **int**.
N'oubliez pas de créer un constructeur.

Job 3.4



Définir une classe **Figure** contenant deux attributs X et Y (double) qui représentent le centre de la figure, une méthode affiche() qui affiche simplement les coordonnées du centre et un constructeur prenant les coordonnées du centre en paramètre.
Modifier votre programme pour que les classes **Cercle** et **Rectangle** héritent de la classe **Figure**.

Job 4.0

Dans un fichier **GestionVehicules.java**, définir 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** ;
- → **son prix courant**.

Définir un constructeur prenant en paramètre les trois attributs correspondant à la marque, la date d'achat et le prix d'achat. Le prix courant sera calculé plus tard.
Définir une méthode publique void **affiche()** qui affiche l'état de l'instance, c'est-à-dire la valeur de ses attributs.

Job 4.1

- → Définir deux classes **Voiture** et **Avion**, héritant de la classe **Vehicule** et ayant les attributs supplémentaires suivants :
- → pour la classe Voiture :
- → sa cylindrée ;
- → son nombre de portes ;
- → sa puissance ;
- → son kilométrage.
- → pour la classe Avion :
- → son type de moteur ("HELICES" ou autre, nous utiliserons "REACTION" pour les exemples) ;
- → son nombre d'heures de vol.

Définir, pour chacune de ces classes, un constructeur permettant l'initialisation explicite de l'ensemble des attributs, ainsi qu'une méthode affichant la valeur des attributs. Constructeurs et méthodes d'affichage devront utiliser les méthodes appropriées de la classe parente !

Job 4.2

Ajouter une méthode void **calculePrix**(int anneActuelle) dans la classe **Vehicule** qui, à ce niveau, fixe le prix courant au prix d'achat moins 1% par année (entre la date d'achat et la date actuelle).

Re-définir cette méthode dans les deux sous-classes **Voiture** et **Avion** de sorte à calculer le prix courant en fonction de certains critères, et mettre à jour l'attribut correspondant au prix courant :

- ➔ Pour une voiture, le prix courant est égal au prix d'achat, moins :
  - ◆ 2% pour chaque année depuis l'achat jusqu'à la date actuelle
  - ◆ 5% pour chaque tranche de 10000km parcourus (on arrondit à la tranche la plus proche)
  - ◆ 10% s'il s'agit d'un véhicule de marque "Renault" ou "Fiat" (ou d'autres marques de votre choix)
  - ◆ et plus 20% s'il s'agit d'un véhicule de marque "Ferrari" ou "Porsche" (idem).
- ➔ Pour un avion, le prix courant est égal au prix d'achat, moins :
  - ◆ 10 % pour chaque tranche de 100 heures de vol s'il s'agit d'un avion à hélices.
  - ◆ 10 % pour chaque tranche de 1000 heures de vol pour les autres types de moteurs.

Le prix doit rester positif (donc s'il est négatif, on le met à 0).

Job 5.0

Le programme **ABCDEF** ci-dessous implémente la hiérarchie de 6 classes :

```
class A {
    public A() { }
}
```

```
class B extends A {
    public B() {
        super();
    }
}
```

```
class C extends B {
    public C() {
        super();
    }
}
```

```
class D extends A {
    protected int d = 1;

    public D(int x) {
        super();
        d = x;
    }

    public D() {
    }
}
```

```
class E extends D {
    public E() {
        super();
    }
}
```

```
class F extends D {
    public F() {
        super();
    }
}
```

```java
class ABCDEF {
    public static void main(String[] args) {
        // Indiquez si les affectations suivantes sont correctes:
        A a = new A();
        B b = new B();
        C c = new C();
        D d = new D();
        E e = new E();
        F f = new F();

        a = b;
        b = a;
        a = (A) b;
        a = null;
        null = a;
        a = d;
        b = d;
        a = e;
        d = e;

        // Remplissage d'un tableau:
        A[] as = new A[10];
        as[0] = new A();
        as[1] = new B();
        as[2] = new D(2);
        as[3] = new E();
        as[4] = new C();
        as[5] = new D(4);
        as[6] = new B();

        // A vous d'ajouter le code de ces deux méthodes:
        rechercher(as);
        additionner(as);
    }

    private static void rechercher(A[] as) {
        // A remplir
    }

    private static void additionner(A[] as) {
        // A remplir
    }
}
```

Affectations entre classes

Dans la méthode main du programme **ABCDEF**, pour chaque affectation à des variables représentant des objets des classes A, B, C, D, E et F, indiquez si elle est correcte. Si ce n'est pas le cas, expliquez pourquoi.

Job 5.1

Dans la deuxième partie de la méthode main, le tableau **as** (de type A[]) est rempli d'objets des classes A, B, C, D, E et F. Écrivez le code de la méthode **rechercher** pour qu'elle affiche le nombre d'objets de type B qui se trouvent dans le tableau.

Exemple d'exécution :

```
Il y a 4 instances de la classe B
```

Job 5.2

Écrivez la méthode **additionner** pour qu'elle affiche la somme des variables d'instance d des objets qui se trouvent dans le tableau *as*.

Exemple d'exécution :

```
Somme des variables d : 9
```

Compétences visées

- → Programmation Orientée Objet (POO)
- → JAVA

Rendu

Le projet est à rendre sur **https://github.com/prenom-nom/runtrackJava**.

Base de connaissances

- → Apprendre le JAVA
- → La syntaxe de base
- → Aide-mémoire JAVA