TP POO

Sum Calculator

Écrivez une classe portant le nom **SimpleCalculator.** La classe a besoin **deux champs (variables d'instance)** avec les noms **firstNumber** et **secondNumber**, tous deux de type double.

Ecrivez les méthodes suivantes (méthodes d'instance) :

- Méthode nommée **getFirstNumber** sans aucun paramètre, elle doit retourner la valeur du champ **firstNumber**.
- Méthode nommée **getSecondNumber** sans aucun paramètre, elle doit renvoyer la valeur du champ **secondNumber**.
- Méthode nommée **setFirstNumber** avec un paramètre de type double, elle doit fixer la valeur du champ **firstNumber**.
- Méthode nommée **setSecondNumber** avec un paramètre de type double, elle doit fixer la valeur du champ **secondNumber**.
- Méthode nommée **getAdditionResult** sans aucun paramètre, elle doit retourner le résultat de l'addition des valeurs des champs **firstNumber** et **secondNumber**.
- Méthode nommée **getSubtractionResult** sans aucun paramètre, elle doit retourner le résultat de la soustraction des valeurs du champ **secondNumber** du **premierNumber**.
- Méthode nommée **getMultiplicationResult** sans aucun paramètre, elle doit retourner le résultat de la multiplication des valeurs du champ **firstNumber** et **secondNumber**.
- La méthode **getDivisionResult**, sans aucun paramètre, doit renvoyer le résultat de la division des valeurs du champ **firstNumber** par le **secondNumber**. Si la valeur de **secondNumber** est 0, elle renvoie 0.

EXEMPLE DE TEST

CODE DE TEST:

```
1. SimpleCalculator calculator = new SimpleCalculator();
2. calculator.setFirstNumber(5.0);
3. calculator.setSecondNumber(4);
4. System.out.println("add= " + calculator.getAdditionResult());
5. System.out.println("subtract= " + calculator.getSubtractionResult());
6. calculator.setFirstNumber(5.25);
7. calculator.setSecondNumber(0);
8. System.out.println("multiply= " + calculator.getMultiplicationResult());
9. System.out.println("divide= " + calculator.getDivisionResult());
```

SORTIE

```
1. add= 9.0
2. subtract= 1.0
3. multiply= 0.0
4. divide= 0.0
```

CONSEILS :

- `add= 9.0` est affiché parce que 5.0 + 4 donne 9.0
- `subtract= 1.0` est affiché parce que 5.0 - 4 donne 1.0
- `multiply= 0.0` est affiché parce que 5.25 * 0 donne 0.0
- `divide= 0.0` est affiché parce que **secondNumber est fixé à 0**

REMARQUE :
- toutes les méthodes doivent être définies comme **public et NON comme public static**.
- Au total, vous devez écrire 8 méthodes.

Person

Écrivez une classe portant le nom **Person**. La classe a besoin de **trois champs (variables d'instance)** avec les noms **firstName**, **lastName** de type **String** et **age** de type **int**.

Écrivez les méthodes suivantes (méthodes d'instance) :

- Méthode nommée **getFirstName** sans aucun paramètre, elle doit retourner la valeur du champ **firstName**.
- Méthode nommée **getLastName** sans aucun paramètre, elle doit retourner la valeur du champ **lastName**.
- Méthode nommée **getAge** sans aucun paramètre, elle doit retourner la valeur du champ **age**.
- Méthode nommée **setFirstName** avec un paramètre de type String, elle doit définir la valeur du champ **firstName**.
- Méthode nommée **setLastName** avec un paramètre de type String, elle doit définir la valeur du champ **lastName**.
- Méthode nommée **setAge** avec un paramètre de type int, elle doit définir la valeur du champ **age**. Si le paramètre est inférieur à 0 ou supérieur à 100, elle doit fixer la valeur du champ **age** à 0.
- Méthode nommée **isTeen** sans aucun paramètre, elle doit renvoyer **true** si la valeur du champ **age** est supérieure à 12 et inférieure à 20, sinon, elle doit renvoyer false.
- Méthode nommée **getFullName** sans aucun paramètre, elle doit renvoyer le nom complet de la personne.
    - Si les champs **firstName** et **lastName** sont des chaînes vides, renvoie une chaîne vide.
    - Si **lastName** est une chaîne vide, il renvoie **firstName**.
    - Si **firstName** es une chaîne vide, il renvoie **lastName.**

Pour vérifier si une chaîne est vide, utilisez la méthode **isEmpty** de la classe **String**. Par exemple, **firstName**.**isEmpty()** renvoie la valeur **true** si la chaîne est vide ou, en d'autres termes, si elle ne contient aucun caractère.

EXEMPLE DE TEST

CODE DE TEST :

```
Person person = new Person();
person.setFirstName("");   // firstName est fixé à une chaîne vide
person.setLastName("");    // lastName est fixé à une chaîne vide
person.setAge(10);
System.out.println("fullName= " + person.getFullName());
System.out.println("teen= " + person.isTeen());
person.setFirstName("John");    // firstName est fixé à John
person.setAge(18);
System.out.println("fullName= " + person.getFullName());
System.out.println("teen= " + person.isTeen());
person.setLastName("Smith");    // lastName est fixé à Smith
System.out.println("fullName= " + person.getFullName());
```

SORTIE

```
fullName=
teen= false
fullName= John
teen= true
fullName= John Smith
```

REMARQUE:
- toutes les méthodes doivent être définies comme **public et NON comme public static**.
- Au total, vous devez écrire 8 méthodes.

Wall Area

Écrivez une classe portant le nom **Wall**. La classe a besoin de **deux champs (variables d'instance)** nommés **width** et **height** de type **double**.
La classe doit avoir deux constructeurs. Le premier constructeur n'a pas de paramètres (constructeur sans argument). Le second constructeur a des paramètres **width** et **height** de type double et doit initialiser ces champs. Si **width** est **inférieure à 0**, la valeur du champ **width** doit être fixée à **0**, et si le paramètre **height** est **inférieur à 0**, la valeur du champ **height** doit être fixée à **0**.
Écrivez les méthodes suivantes (méthodes d'instance) :

- Méthode nommée **getWidth** sans aucun paramètre, elle doit renvoyer la valeur du champ **width**.
- Méthode nommée **getHeight** sans aucun paramètre, elle doit renvoyer la valeur du champ **height**.
- Méthode appelée **setWidth** avec un paramètre de type **double**, elle doit définir la valeur du champ largeur. Si le paramètre **est inférieur à 0**, la valeur du champ **width** doit être fixée à **0**.
- Méthode nommée **setHeight** avec un paramètre de type double, elle doit définir la valeur du champ **height**. Si le paramètre est **inférieur à 0**, la valeur du champ **height** doit être fixée à **0**.
- Méthode nommée **getArea** sans aucun paramètre, elle doit retourner la surface du mur.

EXEMPLE DE TEST
→ CODE DE TEST :

```
1  Wall wall = new Wall(5,4);
2  System.out.println("area= " + wall.getArea());
3
4  wall.setHeight(-1.5);
5  System.out.println("width= " + wall.getWidth());
6  System.out.println("height= " + wall.getHeight());
7  System.out.println("area= " + wall.getArea());
```

→ SORTIE :

```
area= 20.0
width= 5.0
height= 0.0
area= 0.0
```

REMARQUE:
- toutes les méthodes doivent être définies comme **public et NON comme public static**.
- Au total, vous devez écrire 5 méthodes et deux constructeurs.

Point

Vous devez représenter un point dans un espace 2D. Écrivez une classe portant le nom **Point**. La classe a besoin de **deux champs (variables d'instance)** avec les noms **x** et **y** de type **int**.
La classe doit avoir deux constructeurs. Le premier constructeur n'a pas de paramètres (constructeur sans argument). Le second constructeur a des paramètres **x** et **y** de type **int** et doit initialiser les champs.
Écrivez les méthodes suivantes (méthodes d'instance) :

- Méthode nommée **getX** sans aucun paramètre, elle doit retourner la valeur du champ **x**.
- Méthode nommée **getY** sans aucun paramètre, elle doit retourner la valeur du champ **y**.
- Méthode nommée **setX** avec un paramètre de type **int**, elle doit fixer la valeur du champ **x**.
- Méthode nommée **setY** avec un paramètre de type **int**, elle doit fixer la valeur du champ **y**.
- Méthode nommée **distance** sans aucun paramètre, elle doit retourner la distance entre ce **Point** et le **Point** 0,0 sous la forme d'un **double**.
- Méthode nommée **distance** avec **deux paramètres x, y** tous deux de type **int**, elle doit retourner la distance entre ce **Point** et le **Point x,y** sous la forme d'un **double**.

- Méthode nommée **distance** avec comme paramètre un autre **p** de type **Point**, elle doit renvoyer la distance entre ce **Point** et un autre **Point** sous la forme d'un double.

Comment trouver la distance entre deux points ?
Pour trouver la distance entre les points **A(xA,yA)** and **B(xB,yB)**, on utilise la formule :

d(A,B)=√ (xB − xA) * (xB - xA) + (yB − yA) * (yB - yA)

Où √ représente la racine carrée.

EXEMPLE DE TEST
→ CODE DE TEST :

```
Point first = new Point(6, 5);
Point second = new Point(3, 1);
System.out.println("distance(0,0)= " + first.distance());
System.out.println("distance(second)= " + first.distance(second));
System.out.println("distance(2,2)= " + first.distance(2, 2));
Point point = new Point();
System.out.println("distance()= " + point.distance());
```

SORTIE

```
distance(0,0)= 7.810249675906654
distance(second)= 5.0
distance(2,2)= 5.0
distance()= 0.0
```

REMARQUE :
- Utilisez **Math.sqrt** pour calculer la racine carrée
- Evitez de dupliquer le code.
- toutes les méthodes doivent être définies comme **public et NON comme public static**.
- Au total, vous devez écrire 7 méthodes.

Carpet Cost Calculator

L'entreprise de tapis vous a demandé d'écrire une application qui calcule le prix de la moquette pour des pièces rectangulaires. Pour calculer le prix, vous multipliez la surface du sol (largeur multipliée par la longueur) par le prix au mètre carré de la moquette. Par exemple, la surface d'un sol de 12 mètres de long et 10 mètres de large est de 120 mètres carrés. Recouvrir le sol d'une moquette coûtant 8 $ le mètre carré coûterait 960 $.
1. Créez une classe portant le nom **Floor**. La classe a besoin de **deux champs (variables d'instance)** nommés **width** et **length** de type **double**.
La classe doit avoir un constructeur avec les paramètres **width** et **length** de type **double** et doit initialiser les champs.
Si le paramètre **width** est **inférieur à 0**, la valeur du champ **width** doit être fixée à **0**, et si le paramètre **length** est **inférieur à 0**, la valeur du champ **length** doit être fixée à **0**.
Écrivez les **méthodes** suivantes (méthodes d'instance) :

- Méthode nommée **getArea** sans aucun paramètre, elle doit renvoyer la surface calculée (**width * length)**.

2. Écrivez une classe portant le nom **Carpet**. La classe a besoin d'un **champ (variable d'instance)** avec le nom **cost** de type **double**.
La classe doit avoir un constructeur avec un paramètre **cost** de type **double** et doit initialiser ce champ.
Si le paramètre **cost** est **inférieur à 0**, la valeur du champ **cost** doit être fixée à **0**.
Écrivez les **méthodes** suivantes (méthodes d'instance) :

- Méthode nommée **getCost sans aucun paramètre**, elle doit renvoyer la valeur du champ **cost.**

3. Écrivez une classe portant le nom **Calculator**. La classe a besoin de **deux champs (variables d'instance)** nommés **floor** (plancher) de type **Floor** et **carpet** de type **Carpet**.
La classe doit avoir un constructeur avec les paramètres **floor** de type **Floor** et **carpet** de type **Carpet** et doit initialiser ces champs.

Écrivez les **méthodes** suivantes (méthodes d'instance) :

- Méthode nommée **getTotalCost sans aucun paramètre**, elle doit renvoyer le coût total calculé pour couvrir le **floor** (plancher) avec un **carpet** (tapis).

EXEMPLE DE TEST
→ CODE DE TEST :

```
1. Carpet carpet = new Carpet(3.5);
2. Floor floor = new Floor(2.75, 4.0);
3. Calculator calculator = new Calculator(floor, carpet);
4. System.out.println("total= " + calculator.getTotalCost());
5. carpet = new Carpet(1.5);
6. floor = new Floor(5.4, 4.5);
7. calculator = new Calculator(floor, carpet);
8. System.out.println("total= " + calculator.getTotalCost());
```

→ SORTIE

```
1. total= 38.5
2. total= 36.45
```

REMARQUE :
- toutes les méthodes doivent être définies comme **public et NON comme public static**.
- Au total, vous devez écrire 3 classes.
- Rassurez-vous de mettre chaque classe dans son propre fichier.