Cylinder

1. Write a class with the name **Circle.** The class needs **one field (instance variable)** with name **radius** of type **double**.

The class needs to have one constructor with parameter **radius** of type **double** and it needs to initialize the fields.

In case the **radius** parameter is **less than 0** it needs to set the **radius** field value to **0**.

Write the following **methods** (instance methods):

- Method named **getRadius without any parameters**, it needs to **return** the value of **radius** field.
- Method named **getArea without any parameters**, it needs to **return** the calculated area **(radius * radius * PI)**. For **PI** use **Math.PI** constant.

2. Write a class with the name **Cylinder** that extends **Circle** class. The class needs **one field (instance variable)** with name **height** of type **double**.

The class needs to have one constructor with two parameters **radius and height both of** type **double.** It needs to **call parent constructor** and initialize a **height** field.

In case the **height** parameter is **less than 0** it needs to set the **height** field value to **0**.

Write the following **methods** (instance methods):

- Method named **getHeight without any parameters**, it needs to **return** the value of **height** field.
- Method named **getVolume without any parameters,** it needs to **return** the calculated volume. **To calculate volume multiply the area with height.**

TEST EXAMPLE

→ TEST CODE:

```java
Circle circle = new Circle(3.75);
System.out.println("circle.radius= " + circle.getRadius());
System.out.println("circle.area= " + circle.getArea());
Cylinder cylinder = new Cylinder(5.55, 7.25);
System.out.println("cylinder.radius= " + cylinder.getRadius());
System.out.println("cylinder.height= " + cylinder.getHeight());
System.out.println("cylinder.area= " + cylinder.getArea());
System.out.println("cylinder.volume= " + cylinder.getVolume());
```

→ **OUTPUT**

```
circle.radius= 3.75
circle.area= 44.178646691106465
cylinder.radius= 5.55
cylinder.height= 7.25
cylinder.area= 96.76890771219959
cylinder.volume= 701.574580913447
```

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **2 classes.**

NOTE: Make sure to put each class in its own file. **See tabs to the left.**

Person

Write a class with the name **Person.** The class needs **three fields (instance variables)** with the names **firstName, lastName** of type **String** and **age** of type **int**.

Write the following **methods** (instance methods):

- Method named **getFirstName without any parameters**, it needs to **return** the value of the **firstName** field.
- Method named **getLastName without any parameters**, it needs to **return** the value of the **lastName** field.
- Method named **getAge without any parameters**, it needs to **return** the value of the **age** field.
- Method named **setFirstName with one parameter of type String**, it needs to **set the value** of the **firstName** field.
- Method named **setLastName with one parameter of type String**, it needs to **set the value** of the **lastName** field.
- Method named **setAge with one parameter of type int**, it needs to **set the value** of the **age** field. If the **parameter** is **less than 0 or greater than 100,** it needs to set the **age** field value to **0**.
- Method named **isTeen without any parameters,** it needs to **return true** if the value of the **age** field is **greater than 12 and less than 20,** otherwise, **return false**.
- Method named **getFullName without any parameters,** it needs to return the full name of the person.
    - In case both **firstName** and **lastName** fields are empty, Strings **return an empty String**.
    - In case **lastName** is an empty String, **return firstName.**
    - In case **firstName** is an empty String, **return lastName.**

To check if s String is empty, use the method **isEmpty** from the **String** class. For example, **firstName.isEmpty()** returns true if the String is empty or in other words, when the String does not contain any characters.

TEST EXAMPLE

TEST CODE:

1. Person person = new Person();
2. person.setFirstName("");   // firstName is set to empty string
3. person.setLastName("");   // lastName is set to empty string
4. person.setAge(10);
5. System.out.println("fullName= " + person.getFullName());

6. System.out.println("teen= " + person.isTeen());
7. person.setFirstName("John");   // firstName is set to John
8. person.setAge(18);
9. System.out.println("fullName= " + person.getFullName());
10. System.out.println("teen= " + person.isTeen());
11. person.setLastName("Smith");   // lastName is set to Smith
12. System.out.println("fullName= " + person.getFullName());

OUTPUT

1. fullName=
2. teen= false
3. fullName= John
4. teen= true
5. fullName= John Smith

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **8 methods**.

Wall Area

Write a class with the name Wall**.** The class needs **two fields (instance variables)** with name width and height of type double.

The class needs to have **two constructors**:

- The first constructor does not have any parameters (no-args constructor).
- The second constructor has parameters width and height of type double and it needs to initialize the fields.

In case the width parameter is **less than 0** it needs to set the width field value to 0.

In case the height parameter is **less than 0** it needs to set the height field value to 0.

Write the following **methods** (instance methods):

- Method named getWidth **without any parameters**, it needs to return the value of width field.
- Method named getHeight **without any parameters**, it needs to return the value of height field.
- Method named setWidth **with one parameter of type** double, it needs to **set the value** of the width field. If the parameter is **less than 0** it needs to set the width field value to 0.
- Method named setHeight **with one parameter of type** double, it needs to **set the value** of the height field. If the parameter is **less than 0** it needs to set the height field value to 0.
- Method named getArea **without any parameters**, it needs to return the area of the wall.

TEST EXAMPLE

→ TEST CODE:

```
Wall wall = new Wall(5,4);
System.out.println("area= " + wall.getArea());

wall.setHeight(-1.5);
System.out.println("width= " + wall.getWidth());
System.out.println("height= " + wall.getHeight());
System.out.println("area= " + wall.getArea());
```

→ OUTPUT:

1. area= 20.0
2. width= 5.0
3. height= 0.0
4. area= 0.0

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **5 methods and 2 constructors**.

NOTE: Do not add a **main** method to the solution code.

Point

You have to represent a point in 2D space. Write a class with the name Point**.** The class needs **two fields (instance variables)** with name x and y of type int.

The class needs to have two **constructors**. The **first constructor** does not have any parameters (no-arg constructor). The **second constructor** has parameters x and y of type int and it needs to initialize the fields.

Write the following **methods** (instance methods):

- Method named getX **without any parameters**, it needs to **return** the value of x field.
- Method named getY **without any parameters**, it needs to **return** the value of y field.
- Method named setX **with one parameter of type int**, it needs to **set the value** of the x field.
- Method named setY **with one parameter of type int**, it needs to **set the value** of the y field.
- Method named distance **without any parameters,** it needs to return the distance between this **Point** and **Point** (0, 0) as a double.
- Method named distance **with parameter of type Point,** it needs to return the distance between this **Point** and the **parameter Point** as a double.
- Method named distance **with two parameters** x, y **both of type int,** it needs to return the distance between this **Point** and **Point** x, y as a double.

How to find the distance between two points?
To find a distance between points **A(xA,yA)** and **B(xB,yB)**, we use the formula:

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

Where **√** represents square root.

TEST EXAMPLE

→ TEST CODE:

```
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());
```

OUTPUT

```
distance(0,0)= 7.810249675906654
```

2. distance(second)= 5.0
3. distance(2,2)= 5.0
4. distance()= 0.0

NOTE: Use Math.sqrt to calculate the square root √.

NOTE: Try to avoid duplicated code.

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **7 methods**.

Carpet Cost Calculator

The Carpet Company has asked you to write an application that calculates the price of carpeting for rectangular rooms. To calculate the price, you multiply the area of the floor (width times length) by the price per square meter of carpet. For example, the area of the floor that is 12 meters long and 10 meters wide is 120 square meters. To cover the floor with a carpet that costs $8 per square meter would cost $960.

1. Write a class with the name **Floor.** The class needs **two fields (instance variables)** with name **width** and **length** of type **double**.

The class needs to have one constructor with parameters **width** and **length** of type **double** and it needs to initialize the fields.

In case the **width** parameter is **less than 0** it needs to set the **width** field value to **0**, in case the **length** parameter is **less than 0** it needs to set the **length** field value to **0**.

Write the following **methods** (instance methods):

- Method named **getArea without any parameters**, it needs to **return** the calculated area **(width * length)**.

2. Write a class with the name **Carpet.** The class needs **one field (instance variable)** with name **cost** of type **double**.

The class needs to have one constructor with parameter **cost** of type **double** and it needs to initialize the field.

In case the **cost** parameter is **less than 0** it needs to set the **cost** field value to **0.**

Write the following **methods** (instance methods):

- Method named **getCost without any parameters**, it needs to **return** the value of **cost** field

3. Write a class with the name **Calculator.** The class needs **two fields (instance variables)** with name **floor** of type **Floor** and **carpet** of type **Carpet**.

The class needs to have one constructor with parameters **floor** of type **Floor** and **carpet** of type **Carpet** and it needs to initialize the fields.

Write the following **methods** (instance methods):

- Method named **getTotalCost without any parameters**, it needs to **return** the calculated total cost to cover the **floor** with a **carpet**.

TEST EXAMPLE

→ TEST CODE:

1. Carpet carpet = new Carpet(3.5);
2. Floor floor = new Floor(2.75, 4.0);

```
Calculator calculator = new Calculator(floor, carpet);
System.out.println("total= " + calculator.getTotalCost());
carpet = new Carpet(1.5);
floor = new Floor(5.4, 4.5);
calculator = new Calculator(floor, carpet);
System.out.println("total= " + calculator.getTotalCost());
```

→ OUTPUT

```
total= 38.5
total= 36.45
```

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **3 classes.**

NOTE: Make sure to put each class in its own file. **See tabs to the left.**

NOTE: Do not add a **main** method to the solution code.

Complex Operations

A complex number is a number that can be expressed in the form a + bi, where a and b are real numbers, and i is a solution of the equation x2 = −1. Because no real number satisfies this equation, i is called an imaginary number. For the complex number a + bi, a is called the real part, and b is called the imaginary part. **To add or subtract two complex numbers, just add or subtract the corresponding real and imaginary parts. For instance, the sum of 5 + 3*i* and 4 + 2*i* is 9 + 5*i.* For another, the sum of 3 + *i* and –1 + 2*i* is 2 + 3*i.***

Write a class with the name **ComplexNumber.** The class needs **two fields (instance variables)** with name **real** and **imaginary** of type **double**. It represents the Complex Number.

The class needs to have one constructor. The constructor has parameters **real** and **imaginary** of type double and it needs to initialize the fields.

Write the following **methods** (instance methods):

- Method named **getReal without any parameters**, it needs to **return** the value of **real** field.
- Method named **getImaginary without any parameters**, it needs to **return** the value of **imaginary** field.
- Method named **add with two parameters real and imaginary of type double**, it needs to **add** parameters to fields. In other words, it needs to do a complex number add operation as described above.
- Method named **add** with one parameter of type **ComplexNumber**. It needs to add the **ComplexNumber** parameter to the corresponding instance variables.
- Method named **subtract with two parameters real and imaginary of type double**, it needs to **subtract** parameters from fields, in other words, it needs to do a complex number subtract operation as described above.
- Method named **subtract with one parameter of type ComplexNumber.** It needs to subtract the other parameter from this complex number.

TEST EXAMPLE

→ TEST CODE:

1. ComplexNumber one = new ComplexNumber(1.0, 1.0);
2. ComplexNumber number = new ComplexNumber(2.5, -1.5);
3. one.add(1,1);
4. System.out.println("one.real= " + one.getReal());
5. System.out.println("one.imaginary= " + one.getImaginary());
6. one.subtract(number);
7. System.out.println("one.real= " + one.getReal());

```
System.out.println("one.imaginary= " + one.getImaginary());
number.subtract(one);
System.out.println("number.real= " + number.getReal());
System.out.println("number.imaginary= " + number.getImaginary());
```

→ **OUTPUT**

```
one.real= 2.0
one.imaginary= 2.0
one.real= -0.5
one.imaginary= 3.5
number.real= 3.0
number.imaginary= -5.0
```

NOTE: Try to avoid duplicated code.

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **6 methods**.

NOTE: Do not add a **main** method to the solution code.

Sum Calculator

Write a class with the name **SimpleCalculator.** The class needs **two fields (instance variables)** with names **firstNumber** and **secondNumber** both of type **double**.

Write the following methods (instance methods):

- Method named **getFirstNumber without any parameters**, it needs to **return** the value of **firstNumber** field.
- Method named **getSecondNumber without any parameters**, it needs to **return** the value of **secondNumber** field.
- Method named **setFirstNumber with one parameter of type double**, it needs to **set the value** of the **firstNumber** field.
- Method named **setSecondNumber with one parameter of type double**, it needs to **set the value** of the **secondNumber**field.
- Method named **getAdditionResult without any parameters,** it needs to **return** the result of **adding** the field values of **firstNumber** and **secondNumber**.
- Method named **getSubtractionResult without any parameters**, it needs to **return** the result of **subtracting** the field values of **secondNumber** from the **firstNumber**.
- Method named **getMultiplicationResult without any parameters**, it needs to **return** the result of **multiplying** the field values of **firstNumber** and **secondNumber.**
- Method named **getDivisionResult without any parameters** it needs to **return** the result of **dividing** the field values of **firstNumber** by the **secondNumber.** In case the value of **secondNumber** is **0 then return 0**.

TEST EXAMPLE

TEST CODE:

```
SimpleCalculator calculator = new SimpleCalculator();
calculator.setFirstNumber(5.0);
calculator.setSecondNumber(4);
System.out.println("add= " + calculator.getAdditionResult());
System.out.println("subtract= " + calculator.getSubtractionResult());
calculator.setFirstNumber(5.25);
calculator.setSecondNumber(0);
System.out.println("multiply= " + calculator.getMultiplicationResult());
System.out.println("divide= " + calculator.getDivisionResult());
```

OUTPUT

```
add= 9.0
```

2. subtract= 1.0
3. multiply= 0.0
4. divide= 0.0

TIPS:

- **add= 9.0** is printed because 5.0 + 4 is 9.0
- **subtract= 1.0** is printed because 5.0 - 4 is 1.0
- **multiply= 0.0** is printed because 5.25 * 0 is 0.0
- **divide= 0.0** is printed because **secondNumber is set to 0**

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **8 methods**.

Pool Area

The Swimming Company has asked you to write an application that calculates the volume of cuboid shaped pools.

1. Write a class with the name **Rectangle.** The class needs **two fields (instance variable)** with name **width and length** both of type **double**.

The class needs to have one constructor with parameters **width and length** both of type **double** and it needs to initialize the fields.

In case the **width** parameter is **less than 0** it needs to set the **width** field value to **0**.

In case the **length** parameter is **less than 0** it needs to set the **length** field value to **0**.

Write the following **methods** (instance methods):

- Method named **getWidth without any parameters**, it needs to **return** the value of **width** field.
- Method named **getLength without any parameters**, it needs to **return** the value of **length** field.
- Method named **getArea without any parameters**, it needs to **return** the calculated area **(width * length)**.

2. Write a class with the name **Cuboid** that extends **Rectangle** class. The class needs **one field (instance variable)** with name **height** of type **double**.

The class needs to have one constructor with three parameters **width, length, and height all of** type **double.** It needs to **call parent constructor** and initialize a **height** field.

In case the **height** parameter is **less than 0** it needs to set the **height** field value to **0**.

Write the following **methods** (instance methods):

- Method named **getHeight without any parameters**, it needs to **return** the value of **height** field.
- Method named **getVolume without any parameters,** it needs to **return** the calculated volume. **To calculate volume multiply the area with height.**

TEST EXAMPLE

→ TEST CODE:

1. Rectangle rectangle = new Rectangle(5, 10);

```
System.out.println("rectangle.width= " + rectangle.getWidth());
System.out.println("rectangle.length= " + rectangle.getLength());
System.out.println("rectangle.area= " + rectangle.getArea());
Cuboid cuboid = new Cuboid(5,10,5);
System.out.println("cuboid.width= " + cuboid.getWidth());
System.out.println("cuboid.length= " + cuboid.getLength());
System.out.println("cuboid.area= " + cuboid.getArea());
System.out.println("cuboid.height= " + cuboid.getHeight());
System.out.println("cuboid.volume= " + cuboid.getVolume());
```

→ **OUTPUT**

```
rectangle.width= 5.0
rectangle.length= 10.0
rectangle.area= 50.0
cuboid.width= 5.0
cuboid.length= 10.0
cuboid.area= 50.0
cuboid.height= 5.0
cuboid.volume= 250.0
```

NOTE: All methods should be defined as **public NOT public static**.

NOTE: In total, you have to write **2 classes.**

NOTE: Make sure to put each class in its own file. **See tabs to the left.**