SE116 – LAB#3 2023-2024 SPRING

Aim: Understanding constructors, overloaded methods, "this" reference, "has-a" relationship, static class members, "final" instance variables.

1. Implement a class called `Pyramid` for representing a "right rectangular pyramid" shape. It has the following `private` data members: `String color, double length, double width,` and `double height`. Implement appropriate `public` set/get methods for these data members. Regarding the setter methods for length, width and height members you should validate the corresponding argument values so that the numeric data members should not be assigned a negative value. If there is an attempt for a numeric variable to be assigned a negative value, you should use 1.0 instead.

Implement a method `getVolume` that calculates and returns the volume of this shape. Implement also a `printInfo` method to print out all the data members of this shape together with its volume.

The formula for the volume of a pyramid is the following:

$$volume = \frac{(length * width * height)}{3}$$

Implement a no-argument constructor (non-parameterized constructor) where the data members should be initialized with the following default values: color is "white"; length, width and height are 1.0 in default.

Implement also a parameterized constructor that takes the required arguments to initialize all the data members. In the body of the parameterized constructor, validate the corresponding argument values so that the numeric data members should not be assigned a negative value.

Define a class `Test` that deals with the method `main`. In `main`, define an array of 3 `Pyramid` references (Use a `final` variable with the constant value 3 as the size of this array). Remember that the initial contents of the array are all `null`. Therefore, you need to construct 3 new `Pyramid` instances and their references should be assigned to the contents of the array (You can create one `Pyramid` instance using default constructor and create the other two `Pyramid` instances using parameterized constructor). To see the information of the objects, you can call the method `printInfo` via each reference available in the array.

2. Modify your application implemented in 1st question.

First of all, implement a class called `Rectangle`. A rectangle will represent the bottom face (i.e., base) of a pyramid. The class `Rectangle` will contain the following `private` data members: `double length` and `double width`. The class should also have the proper set/get methods, a non-parameterized constructor, and a parameterized constructor. Regarding the setters and constructors, the numeric data members of a `Rectangle` instance should not be assigned a negative value.

Implement also a method `getArea` that calculates and returns the area of this shape.
[You do know how to compute the area of a rectangle, don't you? :)]

Revise your class `Pyramid` to use the "has-a" relationship between class `Pyramid` and class `Rectangle`. Consider that a right rectangular pyramid "has-a" rectangle as its base. In this manner,

the class `Pyramid` has the following `private` data members: `String color, Rectangle base,` and `double height`. Implement appropriate `public` set/get methods for these data members. Furthermore, the class `Pyramid` has a `private static` data member called `numberOfPyramids` to count the total pyramid objects to be created in the memory throughout the program lifetime. The initial value of the class variable `numberOfPyramids` will be 0 and it will be incremented within the constructor methods to count the number of pyramid objects properly. Additionally, define a `public static` getter method that returns the value of `numberOfPyramids`.

Implement a method g`etVolume` that calculates and returns the volume of the pyramid. Implement also a `printInfo` method to print out the color, the base area, and the volume of the pyramid. Implement a non-parameterized constructor where the data members should be initialized with the following default values: color is “white”; base will refer to a new `Rectangle` instance with its default member values; height is 1.0 in default. Implement also a parameterized constructor that takes three argument values to initialize the color, the base and the height respectively. In the body of the parameterized constructor, validate the corresponding argument values so that the numeric data members should not be assigned a negative value. By the way, do not forget to increment the class member `numberOfPyramids` within both of the constructors.

Define a class `Test` that contains the method `main` where you need to demonstrate all the details of your application.

Extra: How would you calculate the areas of 4 triangle surfaces of a right rectangular pyramid?