CS2030 (2310) Exercise #2: 3D Shapes

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

3D Shapes

Topic Coverage

- Composition (has-a) and inheritance (is-a) relationships
- Interfaces
- Abstract classes
- Design principles

Problem Description

In this lab, we are going to explore different design options for modeling 3D objects. For instance, a cuboid is a box-shaped object having six flat sides with all right-angled corners. From the three sides of the cuboid, one can compute its volume. Moreover, a solid cuboid object is made up of a specific material, and by using the density of the material, the mass can be found.

Task

By using object-oriented modeling and applying the abstraction principle, design classes/interfaces to facilitate the creation of a 3D shape and a solid 3D shape, starting with a cuboid and solid cuboid. We then extend this concept to deal with other possible 3D shapes such as a sphere.

Level 1

Create an immutable class `Cuboid` with a constructor that takes in three parameters in order: `height`, `width` and `length` of type `double`. Include the method `volume()` that returns the volume. The string representation of a `Cuboid` is in the form:

```
cuboid [height x width x length]
```

with `height`, `width` and `length` formatted to two decimal places.

```
jshell> new Cuboid(2.0, 2.0, 2.0)
$.. ==> cuboid [2.00 x 2.00 x 2.00]

jshell> new Cuboid(2.0, 2.0, 2.0).volume()
$.. ==> 8.0
```

Level 2

Now create an immutable class `SolidCuboid` with a constructor that takes in four parameters in order: `height`, `width`, `length` and `density` of type `double`. Include the method `mass()` that returns the mass. The string representation of a

`SolidCuboid` is in the form:

```
solid-cuboid [height x width x length] with a mass of mass
```

```
jshell> new SolidCuboid(2.0, 2.0, 2.0, 0.5)
$.. ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> new SolidCuboid(2.0, 2.0, 2.0, 0.5).volume()
$.. ==> 8.0

jshell> new SolidCuboid(2.0, 2.0, 2.0, 0.5).mass()
$.. ==> 4.0

jshell> Cuboid cuboid = new SolidCuboid(2.0, 2.0, 2.0, 0.5)
cuboid ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> cuboid.volume()
$.. ==> 8.0

jshell> cuboid.mass()
|  Error:
|  cannot find symbol
|    symbol:   method mass()
|  cuboid.mass()
|  ^---------^
```

Level 3

In the same way, create a `Sphere` and `SolidSphere` with constructors that take in a `radius`, as well as a `density` in the case of a `SolidSphere`. The following sample run depicts the implementation details.

```
jshell> new Sphere(2.0)
$.. ==> sphere [2.00]

jshell> new Sphere(2.0).volume()
$.. ==> 33.510321638291124

jshell> new SolidSphere(2.0, 0.5)
$.. ==> solid-sphere [2.00] with a mass of 16.76

jshell> new SolidSphere(2.0, 0.5).volume()
$.. ==> 33.510321638291124

jshell> new SolidSphere(2.0, 0.5).mass()
$.. ==> 16.755160819145562

jshell> Sphere sphere = new SolidSphere(2.0, 0.5)
sphere ==> solid-sphere [2.00] with a mass of 16.76

jshell> sphere.volume()
$21 ==> 33.510321638291124

jshell> sphere.mass()
|  Error:
|  cannot find symbol
|    symbol:   method mass()
|  sphere.mass()
|  ^---------^
```

Level 4

Include the `Shape3D` interface and establish the following *is-a* relationships:

- `Cuboid` is a `Shape3D`
- `Sphere` is a `Shape3D`

```
jshell> Cuboid c = new Cuboid(2.0, 2.0, 2.0)
c ==> cuboid [2.00 x 2.00 x 2.00]

jshell> SolidCuboid sc = new SolidCuboid(2.0, 2.0, 2.0, 0.5)
sc ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00
```

```
jshell> Sphere s = new Sphere(2.0)
s ==> sphere [2.00]

jshell> SolidSphere ss = new SolidSphere(2.0, 0.5)
ss ==> solid-sphere [2.00] with a mass of 16.76

jshell> Shape3D shape = c
shape ==> cuboid [2.00 x 2.00 x 2.00]

jshell> shape = sc
shape ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> shape = s
shape ==> sphere [2.00]

jshell> shape = ss
shape ==> solid-sphere [2.00] with a mass of 16.76

jshell> shape.volume()
$.. ==> 33.510321638291124

jshell> c = sc
c ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> s = ss
s ==> solid-sphere [2.00] with a mass of 16.76

jshell> new Shape3D()
|  Error:
|  Shape is abstract; cannot be instantiated
|  new Shape3D()
|  ^---------^
```

Level 5

Notice that it is not easy to model the four classes, `Cuboid`, `SolidCuboid`, `Sphere`, and `SolidSphere`. In particular, the `mass` method has the same implementation repeated in both the `SolidCuboid` and `SolidSphere` classes. This violates the *abstraction principle*.

We have all been taught that a solid has a definite volume and shape. We introduce a `SolidImpl` class with the following relationships:

- `SolidImpl` has a `Shape3D`
- `SolidImpl` has a `density`

```
jshell> SolidImpl solid = new SolidImpl(new Sphere(1.0), 0.5)
solid ==> SolidImpl

jshell> solid.volume()
$.. ==> 4.1887902047863905

jshell> solid.mass()
$.. ==> 2.0943951023931953

jshell> Shape3D shape = solid
shape ==> SolidImpl
```

Level 6

The class `SolidImpl` contains the implementation for computing the mass. It is a good candidate to be inherited from the `SolidCuboid` and `SolidSphere` classes. However, this poses a problem as these two classes are already inheriting from `Cuboid` and `Sphere` respectively!

To circumvent the issue, we can use *delegation* to replace the `SolidImpl` class, while still appearing to allow multiple inheritance.

- Create an interface, let's call it `Solid`, with the method specifications for `mass` and `volume`. The `SolidImpl` class can now be an implementation of `Solid`.
- Let the `SolidCuboid` class inherit from `Cuboid` and implement the `Solid` interface.
- Within `SolidCuboid`, delegate the computation of mass to a `SolidImpl` property in the class.

```
jshell> SolidCuboid solidcuboid = new SolidCuboid(2.0, 2.0, 2.0, 0.5)
solidcuboid ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00
```

```
jshell> solidcuboid.volume()
$.. ==> 8.0

jshell> solidcuboid.mass()
$.. ==> 4.0

jshell> Cuboid cuboid = solidcuboid
cuboid ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> solidcuboid.volume()
$.. ==> 8.0

jshell> solidcuboid.mass()
$.. ==> 4.0

jshell> Shape3D shape = solidcuboid
shape ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> shape.volume()
$.. ==> 8.0

jshell> shape.mass()
|  Error:
|  cannot find symbol
|    symbol:   method mass()
|  shape.mass()
|  ^--------^

jshell> Solid solid = solidcuboid
solid ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> solid.volume()
$.. ==> 8.0

jshell> solid.mass()
$.. ==> 4.0

jshell> shape = cuboid
shape ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00

jshell> shape = solid
shape ==> solid-cuboid [2.00 x 2.00 x 2.00] with a mass of 4.00
```

Do the same for the `SolidSphere`.

Submission (Course)

Select course: CS2030 (2023/2024 Sem 1) - Programming Methodology II

Your Files:

BROWSE

SUBMIT (only .java, .c, .cpp, .h, .jsh, and .py extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.