Consider the following hierarchy of two- and three-dimensional shapes:

Each of these shapes can be implemented as a class; the hierarchy dictates the inheritance properties of the classes. For example, a Circle is a Shape2D.

An object belonging to the Shape class should provide the following service:

- an abstract `toString()` method, which returns a String containing the type and dimensions of the object.

An object belonging to the Shape2D class should have the following additional capabilities, above and beyond those provided by the Shape class:

- two constructors:
  - a zero-parameter ctor, which sets the x- and y-coordinates of the object’s center to each be zero,
  - a two-parameter ctor, which sets the x- and y-coordinates of the object’s center,
- `getCenterX()` and `getCenterY()` methods, returning the x- and y-coordinates of the object’s center.
- an abstract `perimeter()` method, which returns the perimeter of the object, and
- an abstract `area()` method, which returns the area of the object.

An object belonging to the Shape3D class should have the following additional capabilities, above and beyond those provided by the Shape class:

- two constructors:
  - a zero-parameter ctor, which sets the x-, y-, and z-coordinates of the object’s center to each be zero,
  - a three-parameter ctor, which sets the x-, y-, and z-coordinates of the object’s center,
- `getCenterX()`, `getCenterY()`, and `getCenterZ()` methods, returning the x-, y-, and z-coordinates of the object’s center.

---

1Title of an old song by The Yardbirds, one of the earlier bands in which Eric Clapton played.
• an abstract area() method, which returns the surface area of the object, and
• an abstract volume() method, which returns the volume of the object.

An object belonging to the Circle class should have the following additional capabilities, above and beyond those provided by the Shape2D class:

• four different constructors:
  – a zero-parameter ctor, which sets the radius, x-coordinate, and y-coordinate of the circle to all be zero,
  – a one-parameter ctor (said parameter being the radius of the circle being instantiated)—the x- and y-coordinates of the center should both be zero,
  – a two-parameter ctor (said parameters being the x- and y-coordinates of the center)—the radius of the circle should be set to zero,
  – a three-parameter ctor, the parameters being the radius, the x-coordinate of the center, and the y-coordinate of the center,
• a getRadius() method, which returns the radius of the circle.

An object belonging to the Square class should have the following additional capabilities, above and beyond those provided by the Shape2D class:

• four different constructors:
  – a zero-parameter ctor, which sets the side length, x-coordinate, and y-coordinate of the square to all be zero,
  – a one-parameter ctor (said parameter being the side length of the square being instantiated)—the x- and y-coordinates of the center should both be zero,
  – a two-parameter ctor (said parameters being the x- and y-coordinates of the center)—the side length of the square should be set to zero,
  – a three-parameter ctor, the parameters being the side length, the x-coordinate of the center, and the y-coordinate of the center,
• a getSideLength() method, which returns the side length of the square.

An object belonging to the Sphere class should have the following additional capabilities, above and beyond those provided by the Shape3D class:

• four different constructors:
  – a zero-parameter ctor, which sets the radius and the x-, y- and z-coordinates of the sphere’s origin to all be zero,
  – a one-parameter ctor (said parameter being the radius of the sphere being instantiated)—the x-, y-, and z-coordinates of the center should both be zero,
  – a three-parameter ctor (said parameters being the x-, y-, and z-coordinates of the center)—the radius of the sphere should be set to zero,
  – a four-parameter ctor, the parameters being the radius, and the x-, y- and z-coordinates of the center,
• a getRadius() method, which returns the radius of the sphere

An object belonging to the Cube class should have the following additional capabilities, above and beyond those provided by the Shape3D class:

• four different constructors:
– a zero-parameter ctor, which sets the side length and the $x$, $y$, and $z$-coordinates of the cube’s origin to all be zero,
– a one-parameter ctor (said parameter being the side length of the cube being instantiated)—the $x$, $y$, and $z$-coordinates of the center should both be zero,
– a three-parameter ctor (said parameters being the $x$, $y$, and $z$-coordinates of the center)—the side length of the cube should be set to zero,
– a four-parameter ctor, the parameters being the side length and the $x$, $y$, and $z$-coordinates of the center,

• a `getSideLength()` method, which returns the side length of the cube.

Clearly, the `Shape`, `Shape2D`, and `Shape3D` classes are to be abstract classes.

Your task is to implement this class hierarchy. Your implementation should be documented according to the JavaDoc standard, as discussed in class.

I am providing you with a driver program `Proj2.java`. This is located in the directory

```
/u/sobolev/agu/class/client-server/share/proj2
```

You should make a directory that is devoted to this project and copy this file into that directory. Of course, said directory should be the proper subdirectory of your private directory.

Use JavaDoc for documentation. Put the resulting documentation tree in your directory

```
~/public_html/client-server/projects/proj2
```

The quality of this documentation will be count as part of your grade, under the `programming style` rubric. You may find my JavaDoc documentation at

```
http://www.dsm.fordham.edu/~agw/client-server/projects/proj2
```

Note that when you run `javadoc`, the command line should be

```
```

This specifies that only the files whose names match `C*.java` and `S*.java` will be consulted when building the documentation tree. You don’t want `Proj2.java` to be included: you’re documenting the `Shape` class hierarchy, and not client code that uses same.

The output you get when you execute the driver program should look like the following:

```java
$ java Proj2
Circle with:
  radius 3.5
  center at (6.0, 9.0)
  perimeter of 21.991
  area of 38.485

Square with:
  side length 12.0
  center at (2.0, 2.0)
  perimeter of 48
  area of 144

Sphere with:
  radius 5.0
  center at (1.5, 4.5, 2.2)
  area of 314.159
  volume of 523.599
```
Cube with:
  side length 2.2
  center at (0.0, 0.0, 0.0)
  area of 29.04
  volume of 10.648

Circle with:
  radius 0.0
  center at (0.0, 0.0)
  perimeter of 0
  area of 0

Square with:
  side length 0.0
  center at (3.2, 1.6)
  perimeter of 0
  area of 0

**Deliverables:** You are to turn in a clean typescript (produced by the photo program) containing

- a cat of the Java code that you wrote, namely, the files
  - Shape.java
  - Shape2D.java
  - Square.java
  - Circle.java
  - Shape3D.java
  - Cube.java
  - Sphere.java

Do not turn in a listing of Proj2.java, since I’m providing it to you. *Please list them in the order given above!*

- “rm *.class”, followed by “javac Proj2.java”.
- the results of running the Proj2 program.

Good luck!