1 The Problem

This deals with a program takes three types of Polyhedra from an input file and constructs the appropriate objects.

you will be working on the Polyhedron Hierarchy–specifically the Cylinder class.

1.1 Input

The program reads data from one file, polyhedra1.txt. File extensions on Linux may be arbitrary–i.e., this file could have been named with .dat as the extension.

Example 1: polyhedra_1.txt

```
sphere 1
cylinder 2 1
sphere 4
cylinder 2 3
sphere 3
```

Each polyhedron line is formatted as a keyword–i.e., the name of the polyhedron–and all appropriate attributes. A sphere is defined by a radius:

```
sphere 1
```

A cylinder with height 2 and radius 3 would take the form:

```
cylinder 2 3
```

You may assume a valid input file. All input is well-formed.

1.2 Output

The output consists of two reports written to standard output, one after the other.

A report listing all polyhedra that were read from polyhedra1.txt.

A report listing the result of scaling all polyhedra.

If the program is run with polyhedra1.txt as the input file, the following output should be generated:

Example 2: Reference Output

```
Original Polyhedra
------------------------------------------------------------
[Sphere] (2.0, 2.0, 2.0)->Radius: 1.0 Diameter: 2.0
[Cylinder] (2.0, 2.0, 2.0)->Radius: 1.0 Height: 2.0
[Sphere] (8.0, 8.0, 8.0)->Radius: 4.0 Diameter: 8.0
[Cylinder] (6.0, 6.0, 2.0)->Radius: 3.0 Height: 2.0
[Sphere] (6.0, 6.0, 6.0)->Radius: 3.0 Diameter: 6.0


Scaled Polyhedra (Clones)
------------------------------------------------------------
[Sphere] (4.0, 4.0, 4.0)->Radius: 2.0 Diameter: 4.0
[Cylinder] (4.0, 4.0, 4.0)->Radius: 2.0 Height: 4.0
[Sphere] (16.0, 16.0, 16.0)->Radius: 8.0 Diameter: 16.0
[Cylinder] (12.0, 12.0, 4.0)->Radius: 6.0 Height: 4.0
```

```
[Sphere] (12.0, 12.0, 12.0)->Radius: 6.0 Diameter: 12.0
```

The easiest way to generate the expected output is to run the sample executable solution I have provided. These two files are named as command-line parameters when the program is executed.

1.3 Compiling and Running the Java Program

If the sample data above is kept in polyhedra1.txt, to run this program, type:

```
 java -jar build/libs/CreatePolyhedra.jar src/main/resources/polyhedra1.txt 2
```

or

```
./gradlew run
```

Where the latter command executes the run target in the Gradle buildfile (which runs the first command for you). This allows us to use fewer keystrokes (which are expensive).

Run the compiled solution with both the provided input file and your own test input files.

Once you have completed your solution, compare the output generated by your solution to the output generated by my solution. The two sets must be identical.

1.4 Compiling and Running the Tests

You can compile and and run the test driver (TestCylinder.class) with

```
./gradlew test
```

If you implemented everything in Composite correctly you will see:

```
polyhedra.TestCylinder > testClone PASSED
polyhedra.TestCylinder > testDefaultConstructor PASSED
polyhedra.TestCylinder > testNonDefaultConstructor PASSED
polyhedra.TestCylinder > testRead PASSED
polyhedra.TestCylinder > testScale PASSED
polyhedra.TestCylinder > testSetHeight PASSED
polyhedra.TestCylinder > testSetRadius PASSED
polyhedra.TestCylinder > testToStringAfterScale PASSED
polyhedra.TestCylinder > testToStringConstructor PASSED
polyhedra.TestCylinder > testToStringDefaultConstructor PASSED
```

If you see FAILED you must revisit revisit the corresponding function(s). There is a mistake in your code.

1.5 Your Tasks

Complete the Cylinder class. You need to fully implement:

Note that I have provided you a clone function and a skeleton for a copy constructor. You can either complete the copy constructor, or remove the copy constructor (updating clone as appropriate).

setRadius
setHeight
toString

read
clone & the Cylinder Copy Constructor
scale
Complete clone (copy constructor portion), scale, and toString, and read functions for Composite class.

2 Mechanics

2.1 Packages

Do not change/add any packages or modules
Do not modify any package lines,

```
package polyhedra;
```

If you change the package, the tests will fail.

Tests 000 through 005 evaluate each of your Composite methods (member functions):

Test 000 evaluates Cylinder.setRadius
Test 001 evaluates Cylinder.setHeight
Test 002 evaluates Cylinder.toString.
Test 003 evaluates Cylinder.read.
Test 004 evaluates Cylinder.scale.
Test 005 evaluates Cylinder.clone.
Of course, you should already know the results of each test. Maybe you should use ./gradlew test. Unit Testing is fun…