Kresimir Tokic  
CMSC335 Project 1  
User Guide, Test Plan, UML diagram, Lessons Learned

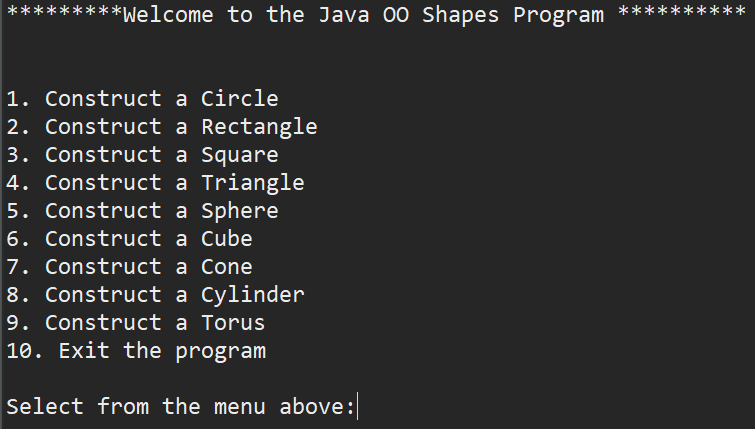
**User Guide**

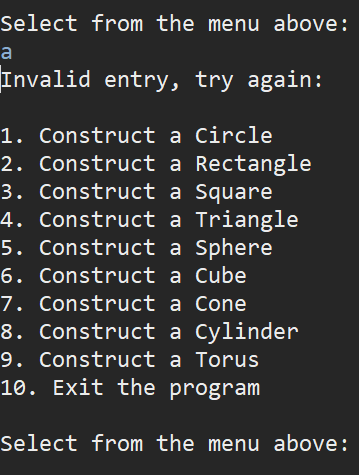
Unzip all \*.java files into the same folder. Execute the Main.java file. When prompted enter a value 1-10. Continue following prompts as desired.

**Test Plan**

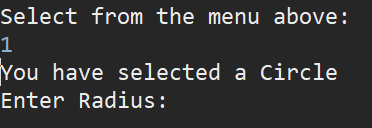
The following sequence illustrates proof that all of the methods in the Program class function correctly and evidence that the area and volume methods of two subclasses work correctly as well.

* Execute Main.java
* Enter “a” instead of a number 1-10.
* Expect to see error message, new menu and prompt.

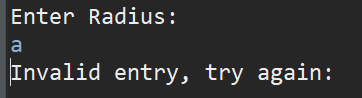




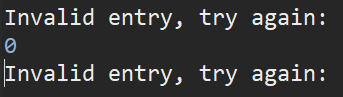
* Enter “1”
* Expect to see prompts to enter relevant dimension



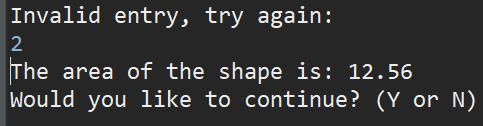
* Enter “a”
* Expect to see error message and prompt to try again



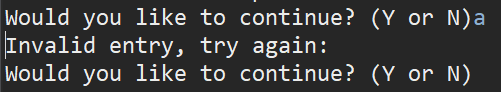
* Enter “0”
* Expect to see error message and prompt to try again



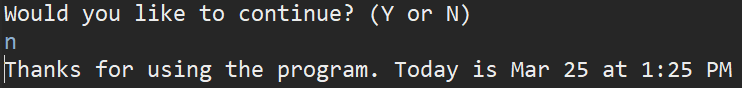
* Enter “2”
* Expect to see relevant area or volume and prompt to continue



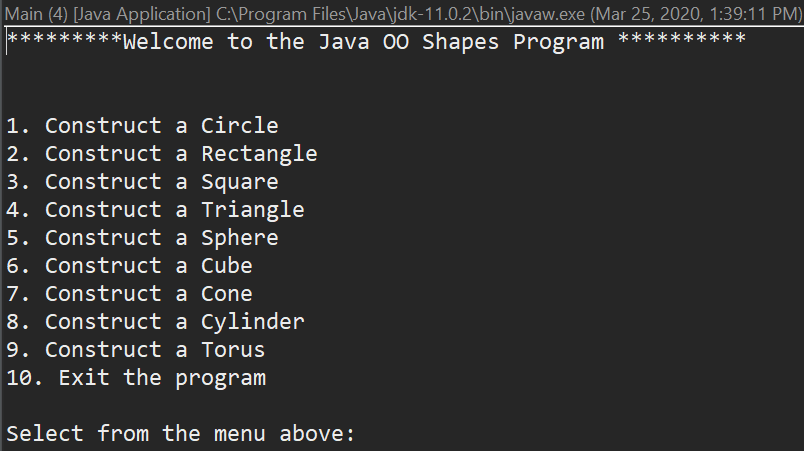
* Enter “a”
* Expect to see error message and prompt again

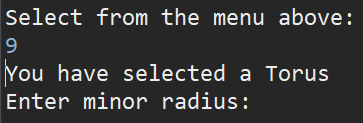


* Enter “n”
* Expect to see exit message with time stamp

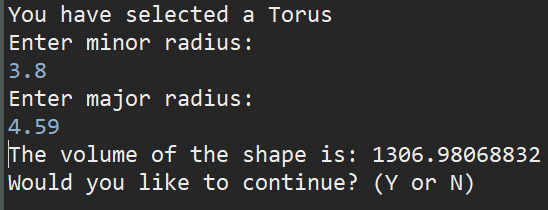


* Execute Main.java
* Enter “9”
* Expect to see prompts to enter relevant dimension

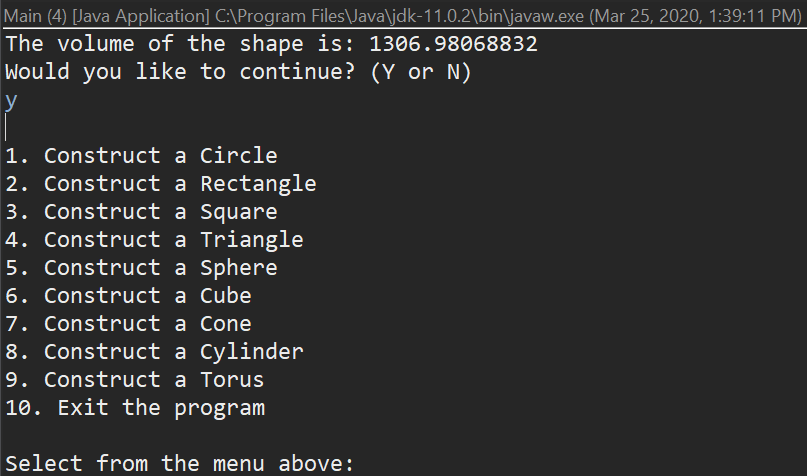




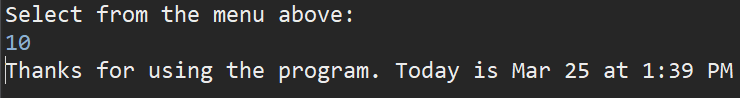
* Enter “3.8”
* Expect to see next prompt for next dim
* Enter “4.59”
* Expect to see relevant area or volume and prompt to continue



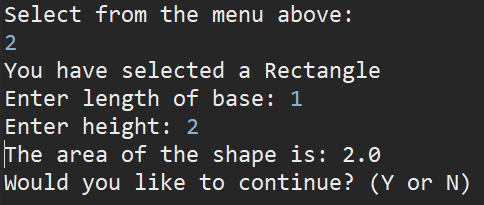
* Enter “y”
* Expect to see top menu



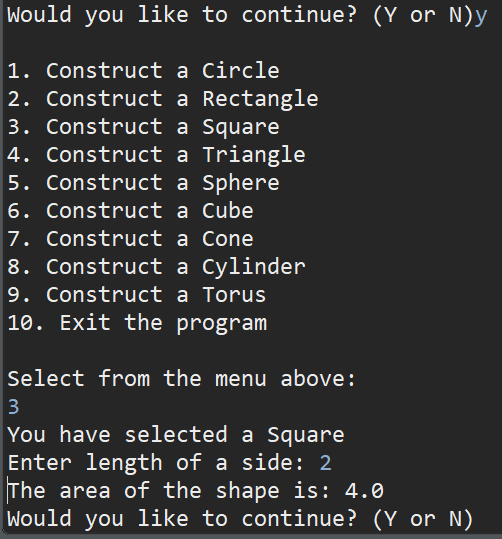
* Enter “10”
* Expect to see exit message with time stamp



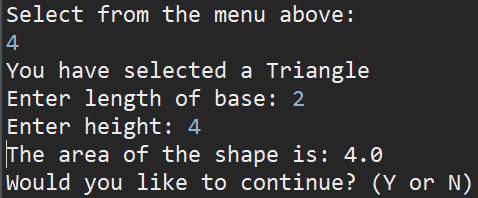
* The following screen shots illustrate successful execution of all the sub classes starting with selection “2” the rectangle then moving through the selection “8” the cylinder.



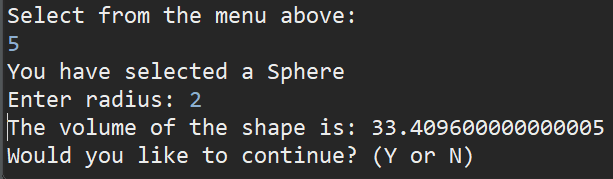
* Square



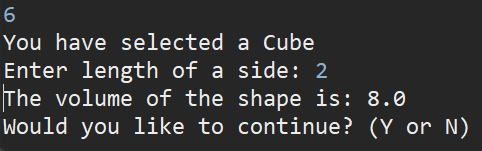
* Triangle



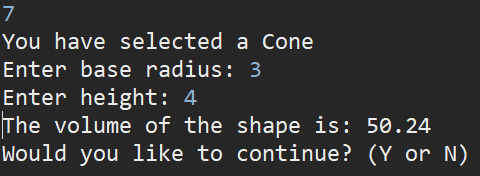
* Sphere



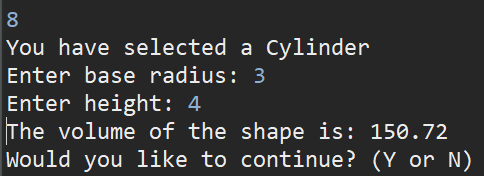
* Cube



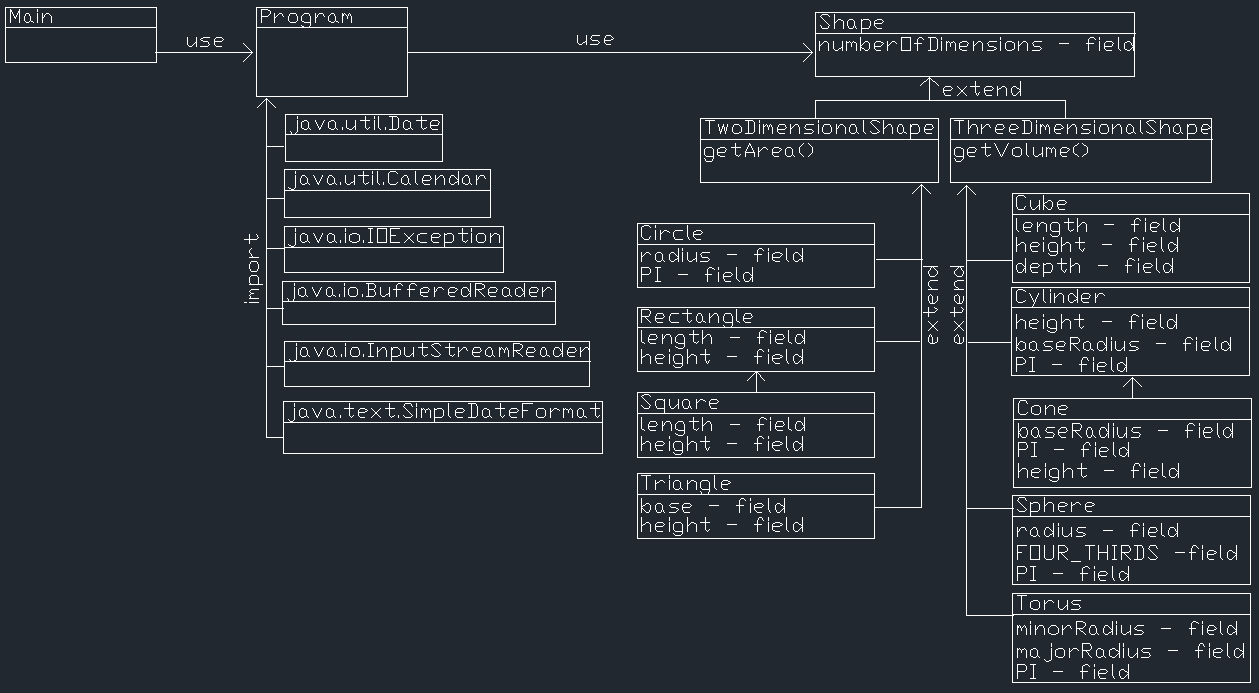
* Cone



* Cylinder



UML



**Lessons Learned**

Initially I assumed I would need to use interfaces but after rereading the project rubric I realized I needed to make classes of Shape its subclasses Two-Dimensional Shape and Three-Dimensional Shape. Creating all the subclasses wasn’t too difficult but it takes time to create lots of them. I realized while making the subclasses that there were a few that felt almost identical. I had to try using “instanceof” to determine if I could get away with extending Rectangle to Square. Leveraging “instanceof” allowed me to minimize some of my repeated code in my Program. It’s still not immediately obvious to me how to break down the program code into most efficient and effective reusable elements. It’s not always clear to me how far you should go in creating reusable code. For example, in my Program class I created lots of static final strings to simplify printing those strings. I also tried to create a bunch of methods that can be called from different areas of the program. I assume this creates more reusable code but at what point is it overkill and no longer efficient? I am under the impression that this becomes almost second nature with experience.

Technically my program does not loop, instead the methods will be called multiple times. I assume this is creates a smaller Big-O number, but this program could’ve been completed with a single loop and that’s not too significant in my opinion.

Additionally, three dimensional shapes have surface areas, but this is not required of the program. I think have a cube extend a square and override the number the dimensions and area method while adding the volume method. I don’t think you’d want a volume method in the Shape superclass or Two-Dimensional subclass because those shapes do not have volumes. If you had a volume method in the Shape class could you override it to return a null value or print a statement explaining it does not apply to two-dimensional shapes? For example, creating a sub and super relationship between circle and sphere.