Exercise 4 – June 23rd, 2020
Design Patterns

For this exercise, go to https://classroom.github.com/g/WwET7E9J, and accept the assignment.

In this exercise, we will refactor our code to implement two patterns, the **Composite** pattern and the **Factory** pattern.

Part 1 – The Factory pattern.

Our current code implementation parses shapes from a `scene.x3d` file and carries out ray tracing for each shape that is determined. Although we parse each of the shapes into a `Shape`, we do not know what *type* of shapes the `*.x3d` file will specify *ahead of time*. To implement the logic behind creating/parsing shapes (manufacturing `Shape` objects, so to speak), we will need to:

- Implement a `ShapeFactory`, making sure we observe the **Factory pattern**.

Part 2 – The Composite pattern.

Each transform specifies geometric translation, scaling, and rotation for the shapes defined within the provided `scene.x3d` file. A transform can contain multiple transforms within itself, i.e., each transform can be a parent `Transform` Object that can contain one or more child `Transform` Objects. Likewise, each child `Transform` Object can be a parent `Transform` Object that can contain one or more child `Transform` Objects. In other words, the transforms can be nested, and stack inherently. For this part we will need to:

- parse all transforms into `Transform` Objects, making sure to observe the **Composite pattern**, and
- ray trace shapes with respect to their transformed positions, making sure to observe the **Composite pattern** (hint: the translations are nested! factor this in your implementation).

Deliverable

On GitHub, commit your solution with the requested features by **6th of July**, i.e., before **midnight**. In the team.md, specify the full name and email address of each team member (**n.b.**, as appearing in Moodle).

Scoring

Prerequisite: Your assignment will only be graded if it builds in Travis without error!

- Travis builds without error: **1/2 point.**
- Good coding and git practices: **1/2 point.**
- UML representation of your Factory pattern: **1 point.**
- UML representation of your Composite pattern: **1 point.**
- All tests in `AbstractPatternTests` pass: **2 points.**
- All tests in `CompositePatternTests` pass: **2 points.**
- `Image.png` output is correct: **1 point.**

Total: **8 points.**