Exercise 4 – Dec 5th, 2019
Design Patterns (Part 1)

For this exercise, go to https://classroom.github.com/g/upKNqrUP, choose your Moodle email address and accept the assignment.

In this exercise, we want to integrate the `Transform` and `Appearance` objects and their main functions into our code.

Part 1 – Appearance

The `Appearance` object represents how the surface of a shape is rendered. For now, we will only use the `diffuseColor` property of the child `Material` node.

- Find a way to pass this property to the containing `Shape` object. Consider using the Proxy pattern for your solution.
- Note that any item in the object sequence (`Shape` → `Appearance` → `Material` → `diffuseColor`) may be missing or empty; in this case, use a visually distinct default color.
- Extend your raytracing code to use the `diffuseColor` property to give individual colors to each rendered shape (if defined in the X3D file).

If your code is correct, the main method will render differently colored shapes.

Part 2 – Transform

The `Transform` object specifies how to move shapes in space via its `translation` property from their default position at the origin. Important: this applies to *all* child objects, and especially to nested `Transform` objects (the translations stack up).

- Find a way to keep track of all current `Transform` objects that apply to a `Shape`, especially nested ones. Consider using the Composite pattern for your solution; this will involve extending your class hierarchy.
- Extend your raytracing code to use that current transformation while rendering, i.e. every intersection test needs to take this current state into account.

If your code is correct, the main method will render multiple separate shapes per image.

Deliverable

On GitHub, commit your solution with the requested features by Wednesday, Dec. 18th, 23:55. In the README.md, specify the full name and email address used in Moodle of each team member (max. 2).

Scoring

Prerequisite: You only get points if all source files compile when built via `gradle build`.

- Separate commits for each feature/method (best practice for Git): **1 point.**
- The color value pass-through has been implemented correctly: **1 point.**
- The renderer creates correctly colored images: **1 points.**
- Object structure is managed using Composite pattern: **2 points.**
- Object translations are handled correctly: **1 point.**
- Nested Transform objects are handled correctly: **2 points.**

Total: **8 points.**