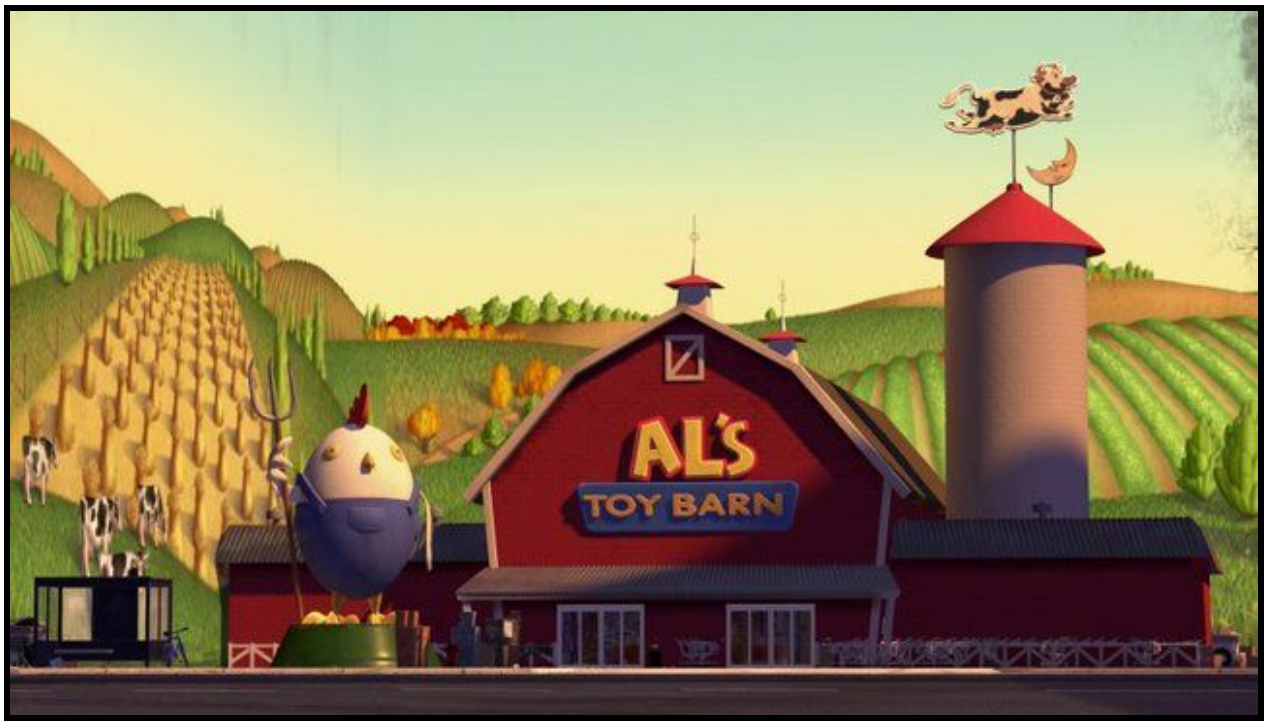
Software Dev. & Problem Solving II CSEC/SWEN-124

Interfaces Assignment 3.3



Goals of the Assignment

The goal of this assignment is to expand the class hierarchy you started in the previous assignment to include Toy Factories capable of building toys! As always, you are expected to practice good software engineering, including unit testing and practicing the Git workflow. Please read this document ***in its entirety*** before asking for help from the course staff.

Background

A ***product*** has a ***name***, a unique ***product code*** that is at least 7-digits, and a ***manufacturer's suggested retail price*** (***MSRP***).

A ***truck*** has a ***capacity*** that determines how many products can be ***loaded*** onto the truck. Trucks are loaded front to back, and ***unloaded*** back to front.

A ***factory*** manufactures ***products***. Each individual factory manufactures a specific kind of product, e.g. a Car Factory manufactures Cars.

A ***complex*** is a collection of ***factories*** that are geographically close to each other. When trucks arrive at the complex, they are ***loaded*** with a random assortment of products from the factories in the complex.

Activities

1. Create a package called "`products`" and, unless otherwise instructed, use it for the artifacts required for this assignment.

2. Use the background information provided above to perform a ***noun/verb*** analysis using the tool of your choice. The analysis should ***only*** include the nouns/verbs mentioned above. Tool options include:
    a. A text file.
    b. A table like those used in the lecture (e.g. Goats, Trolls)
    c. Hand-written on a whiteboard or paper and photographed ***legibly***.

    You must ***commit*** your analysis to your repository ***before*** beginning the next step.

3. Use your noun/verb analysis to create a draft diagram of your classes. It does not have to be complete or perfect; a decent attempt will suffice. You ***must*** have at least one `interface` and several relationships between classes. ***Hint***: Your diagram should have *at least* 4 classes/interfaces in it when you are done. Tool options include:
    a. Tables in a Google Slide (screenshot)
    b. Lucidchart or another UML Drawing Tool (screenshot or PDF)
    c. Hand-drawn on paper or a whiteboard ***legibly*** and photographed.

    You must ***commit*** your UML class diagram to your repository ***before*** moving to the next step. Use a comment like "initial UML class diagram" to make it easy for your grader to find it in your commit history. You do not need to revisit the diagram if you make changes to your design afterward, but it should show obvious effort.

4. Implement the classes and interfaces in your UML diagram. If you have to change the classes and interfaces so that they are different from your diagram, that's OK. Don't worry about updating the diagram. Here are some things to consider:
    a. As usual, you are restricted to using only Java language features discussed in class or specifically allowed by your instructor. If you have a question about whether or not you can use something, *ask your instructor*.
    b. Every factory can manufacture a product, but each one makes a different product. You are not given enough detail (yet) to implement a factory.
    c. Trucks can be loaded until they are full. They can be unloaded one product at a time until they are empty. They are loaded and unloaded in the opposite order (front to back, then back to front). Do not worry about trying to load a full truck or unload an empty truck (it is OK if an exception occurs).
    d. A complex is created with a specific set of factories. When a truck arrives to be loaded, factories are chosen randomly from the set to manufacture products one at a time until the truck is fully loaded.

5. Update the toy class in your toy assignment to implement or extend your newly created product type. Make the necessary changes so that method signatures appropriately

match. This *should* require minimal changes given that your toy class should already include a name, product code, and MSRP.

6. Create at least *two* toy factories using the requirements below:
    a. Robots require one sound randomly chosen from a set of choices determined by you. The MSRP must be between $30-100 and the price must end in .99, e.g. $34.99.
    b. Dolls have hair color and eye color randomly chosen from a set of choices determined by you. The MSRP must be between $5-30 and end in .99, e.g. $11.99.
    c. Action Figures have hair color and eye color randomly chosen from a set of choices determined by you. Whether or not the action figure has Kung-Fu Grip™ is also randomly determined. The MSRP must be between $5-15 and end in .99, e.g. $7.99.

7. Create a new Java class in a file called "`Main.java`" in the `toys` package and define a `main` method with the appropriate signature that:
    a. Creates a complex with at least one of each of your factories.
    b. Creates an empty truck.
    c. Loads the truck at the complex.
    d. Unloads the truck and prints each product to standard output.
    e. In the file JavaDoc for this class, explain how ***polymorphism*** is being used in your simulation. ***Hint***: what if you made a completely new kind of product that wasn't a toy, e.g. Microwave Ovens or Kitchen Tables? How would you need to change the implementation in your truck or complex classes to handle this new product?

Submission Instructions

You must ensure that your solution to this assignment is pushed to GitHub *before* the start of the next lecture period. See the course syllabus for the rubric that will be used to evaluate your submission.