UML Lab

Appalachian State University

Mx. Willow Sapphire

CS 3667

This project will help you get familiar with the different types of UML diagrams and give you some practice in reading and creating them.

For the questions that ask you to create UML diagrams you may use any diagram software of your choice. I would highly recommend LucidChart. It is free, although you are limited to the number of diagrams you can have at once. When you have finished a diagram and exported it as a png, you can delete it from LucidChart to free up space. If you prefer a different software, you are welcome to use it. However, all images should be png files.

For every assignment, you will create a new branch. add, commit, and push your work to this new branch. Remember, the first time that you push the branch to GitHub you will need to use the -u flag to tell GitHub that it is a new branch. When you have finished, make a pull request on GitHub to merge the feature branch into main. You can merge this pull request yourself. If you work directly on main or do not use a Pull Request to merge the branches, you will lose some points. There are six assignments so I should see seven total branches in your repository and six pull requests (or more if you make changes after merging the first time). You can verify that you have done this correctly by checking your branches and pull requests on GitHub.

Table of Contents

Use Case Diagrams

Reading Use Case Diagrams

Make a branch named self-checkout. Create a file named answers.txt in the UseCaseDiagram directory. Take a look at the self-checkout.png UML diagram and use it to answer the following questions. Put your answers in the answers.txt file. Make sure to number the answers so it is easy to tell which answer goes to which question. Most questions can be answered with a short list or one or two sentences.

1. What are the primary actors?
2. What are the secondary actors?
3. What type of word describes the use cases? (noun, adjective, verb, adverb, etc)
4. What type of word describes the actors?
5. Why is/are the primary actor(s) primary and not secondary?
6. Why is/are the secondary actor(s) secondary and not primary?
7. Why does Verify ID extend Scan Items?
8. Why does Scan Items include Check Cart for Items?

Creating Use Case Diagrams

Make a branch named vending. Use a UML diagram software to create a use case diagram for a vending machine system like those on the bottom floor of Anne Belk. You should have at least five use cases and two actors. There should be at least one each of extends, includes, and generalization. It is okay if you don't fully understand the inner workings of vending machines. Your diagram does not necessarily have to be accurate, but it should be reasonable. When you have finished, export your file as a png named checkout.png and add it to the UseCaseDiagrams directory.

Class Diagrams

Reading Class Diagrams

Make a branch name stove. In the ClassDiagram directory, there is a diagram named Stove.png. This diagram models, as you would expect, a stove. In the src directory, write skeleton classes for the entire program as described in Stove.png. You do not need to implement the methods, as their implementation is not described, but you should add return statements as necessary to make the program compile. Every object in the diagram should be in its own Java file. You can run `javac -d bin src/*.java` to compile the program. Run this command from the root folder. You can check the bin folder to ensure the class files were generated there. Make sure that your program compiles before making your pull request and merging the feature branch into main.

Creating Class Diagrams

Let's imagine a scenario where we have been hired to create a program that models the card game Uno. If you are not familiar with Uno, you can read about the game here: https://www.unorules.com.

We are preparing to write our program and, as an early step, we want to create a high level class diagram. There should be at least three classes, some inheritance, aggregation, and two enumerated types. Include at least a few fields and methods for each class, but you do not need to include every single attribute and method that we may require. When you are finished, export your diagram as a png and name it uno.png. Add it to the ClassDiagram directory. Make sure to do this on a separate branch.

Your Choice of Diagrams

Creating Two Diagrams

Create two diagrams modeling one system of your choice. At least one of the diagrams should be a sequence diagram, activity diagram, or a state diagram. The other can be whatever type of diagram you want that we have gone over in class. Your diagrams do not have to be complex, but they should be reasonable.

Some examples of systems to model: a traffic light system, an elevator system, a fire alarm system. Each of these diagrams should be added on their own branches.