This assignment is intended to be done by your team of two students. You may collaborate on answers to all questions or divide the work for the team. In any case, the team should review the submission as a team before it is turned in.

Project 2.1: OO and UML exercises – 15 points

Provide answers to each of the following in a PDF document:

1. (5 points) What are three ways modern Java interfaces differ from the traditional OO interface design concept that describes only abstract method signatures to be implemented? Provide a Java code example of each.

2. (5 points) Describe the differences and relationship between abstraction and encapsulation. Provide a text, pseudo code, or Java code example that illustrates the difference.

3. (5 points) Draw a UML class diagram for the FNCD simulation described in Project 2.2. The class diagram should contain any classes, abstract classes, or interfaces you plan to implement to make the system work. Classes should include any key methods or attributes (not including constructors). Delegation or inheritance links should be clear. Multiplicity and accessibility tags are optional. Design of the UML diagram should be a team activity if possible and should help with eventual code development.

Project 2.2: The FNCD simulation – 35 points

The Java code you develop here for Project 2 will also be used for Projects 3 and 4. In most semesters we use Project 2, 3, and 4 to practice OO programming by simulating days of activities at different places, zoos, restaurants, dungeons, etc. This time, we are going to simulate a Friendly Neighborhood Car Dealership (FNCD).

The FNCD sells lightly used cars of different kinds to people in the neighborhood, employing a variety of staff members. Your simulation will run selected operations at a model of the FNCD using the specifications below for a 30-day simulated period. The FNCD will start with an operating budget of $500000 for buying cars and paying staff.

The FNCD has a group of Staff that work there. The Staff is made up of three distinct types: Salespeople, Mechanics, and Interns. (The Staff should be represented with an IS-A inheritance hierarchy.) You will create instances of the different types of Staff members, and you'll need to come up with an algorithm to give them unique names. They will also have a unique daily salary rate by type (you should select a salary level that works with the simulation for each Staff type). Individual Staff members will need to be able to track salary and bonus earned as well as total days worked.

The FNCD sells Vehicles. Vehicles come in three types, Performance Cars, Cars, and Pickups. (The Vehicles should be represented in an IS-A inheritance hierarchy as well.) You will create instances of Vehicles that will arrive at the FNCD to be worked on by the staff and sold to buyers that visit. Instances of individual Vehicle types will also need a naming algorithm to give them unique names (or numbers).

Different types of Vehicles will have a different bonus for successful sales, repair, or wash (you will decide the values). Individual Vehicles will have a cost (what the FNCD pays for the Vehicle) and a sales price (starting at 2x the final determined cost). The cost of a Performance Car will be between $20000 and $40000. The cost of a regular car will be $10000 to $20000. The cost of a Pickup will be $10000 to

$40000. Vehicles have a Condition of: Like New, Used, or Broken (randomly determined when initialized). Used Vehicle initial cost is reduced by 20%. If Broken, the initial cost is reduced by 50%. Vehicles will also arrive in a state of cleanliness: sparkling (5%), clean (35%), and dirty (60%).

Each day that the FNCD is open, several Activities will occur. Note that the FNCD is not open on Sundays. It is possible, during Activities, the FNCD might run out of money in the operating budget. If this happens, add $250000 to the FNCD operating budget, announce the event, and keep track of money added in this fashion.

Activities are as follows:

Opening – When the FNCD opens, several operations happen.

First, the current Staff of three Salespeople, three Mechanics, and three Interns show up for work. If, because of other events, there are less than three Interns, hire (e.g. add) new Interns to the FNCD to bring the count back to three. (For day one, clearly all staff will have to be added.)

Second, there should be four of each type of Vehicle in inventory. If there are not for any reason, additional vehicles will be sent to the FNCD. These newly instantiated Vehicles must be paid for, reducing the operating budget by the cost of the Vehicle. The Vehicles will then be part of the FNCD inventory.

Washing – Every working day, the Interns will wash Vehicles.

Each Intern can wash two Vehicles per day. They will start with randomly selected Dirty Vehicles and then move on to randomly selected clean Vehicles. Vehicles that are Sparkling will not be rewashed. A Dirty Vehicle that is washed has an 80% chance of becoming Clean and a 10% chance of becoming Sparkling. A Clean Vehicle has a 5% chance of becoming Dirty and a 30% chance of becoming Sparkling. If an intern makes a Vehicle Sparkling, they earn a bonus by type of vehicle.

Repairing – Every working day, Mechanics will fix Vehicles.

Each Mechanic can fix two Vehicles per day. Broken Vehicles that are fixed become Used. Used Vehicles that are fixed become Like New. Each Mechanic has an 80% chance of fixing any Vehicle worked on. Whether fixed or not, any Vehicle worked on will go down one class of cleanliness (if not already Dirty). A Vehicle that becomes Used has its sales price increased 50%. A Vehicle that becomes Like New has its sales price increased 25%. Mechanics receive a bonus from each successful repair by Vehicle type.

Selling – Every working day, 0 to 5 Buyers will arrive to buy a Vehicle (2-8 Buyers on Friday/Saturday) from a Salesperson

Buyers buy from a Salesperson (randomly selected). Buyers are initialized randomly with one of three types of Buying, each with a base chance of buying a Vehicle: Just Looking (10%), Wants One (40%), Needs One (70%). The Buyers will have a type of Vehicle they want (Performance Car, Car, Pickup) randomly determined.

A Salesperson will always try to sell the most expensive Vehicle of the type the Buyer wants that is in the inventory. (Note that Broken Vehicles cannot be sold to Buyers.) The chance of a sale increases by 10% if the Vehicle is Like New and another 10% if the Vehicle is Sparkling. If no Vehicles of the Buyer's

Vehicle type choice are available, the Salesperson will try to sell the most expensive Vehicle left in Inventory at a -20% chance for the sale. If the buyer buys the Vehicle, they will give the FNCD the sales price (which goes into the operating budget) and the Salesperson will get a bonus by Vehicle Type; the Vehicle should be removed from the FNCD working inventory, but should be stored in a collection of Sold Vehicles for later review.

Ending – Every day, pay the Staff and report on progress.

Give all Staff members their daily salary pay for today's work (out of the operating budget).

It's possible at the end of every day one Mechanic, one Salesperson, and one Intern may quit the FNCD. There is a 10% chance for each Staff type that one member may quit. If an Intern quits, simply remove them from Staff. If a Mechanic or a Salesperson quits, remove them from the store Staff and add a new Mechanic or Salesperson to the Staff (as needed) using an existing Intern to provide the name for this new Staff member. Remove the Intern used here from the Interns working at the FNCD. (The Intern has become the new Mechanic or Salesperson.) When removing Staff from the FNCD, store the leaving instance in a collection of Departed Staff.

Produce a readable, tabular report of:

Staff members – with total days worked, total normal pay, total bonus pay, working or quit the FNCD

Inventory – List of all Vehicles with Name, Cost, Sale Price, Condition, Cleanliness, Sold or In Stock

Total $ in operating budget, total sales $ for day.

Output for and Thoughts on the FNCD Simulation

The code should be in Java 8 or higher and should be object-oriented in design and implementation.

Captured output: Run the simulation for 30 simulated days. All noteworthy events or state changes in the simulation should generate output, for example, some typical output messages:

> *** FNCD Day 4 ***
>
> Opening… (current budget $238160)
>
> Hired Intern Priya
>
> Purchased Used, Dirty Performance Car for $12000 Cost
>
> Washing…
>
> Intern Priya washed Performance Car Racer-X and made it Sparkling (earned $100 bonus)
>
> Mechanic Yan repaired Broken Car Celica 1 and made it Used (earned $200 bonus)
>
> Salesperson Piotr sold Sparkling Used Pickup Ranger 3 to Buyer for $23000 (earned $500 bonus)
>
> Intern Priya has quit the FNCD

Capture the output from the entire run in a text file called SimResults.txt.

Identify OO Elements: In commenting the code, include comments for at least one example of: **Inheritance, Polymorphism, Cohesion, Identity, Encapsulation, and Abstraction.**

Include a UML diagram update: Also include in your repository an updated version of the FNCD UML diagram from 2.1 that shows your actual class implementations in project 2.2. Note what changed between part 2.1 and part 2.2 (if anything) in a comment paragraph.

Hints on construction: I suggest you start with a very simple version of the simulation that has small examples of all elements and actions (this is known as a "vertical slice"), then add variations and more instances once you have the basics working. ArrayLists make great holders for collections of objects instances.

Errors, boundary conditions, incomplete specifications: There may be possible error conditions that you may need to define policies for and then check for their occurrence. These include running out of Vehicles to sell or money in the FNCD budget. You may also find requirements are not complete in all cases. Document any assumptions you make in the project's README file. See class staff for any guidance needed.

Grading Rubric

Project 2 is worth 50 points total.

Part 2.1 is worth 15 points and is due on Wednesday 2/8 at 8 PM. The submission will be a single PDF per team. The PDF must contain the names of all team members.

Questions 1-2 will be graded on the quality (not quantity) of the answer. Questions 1-3 should include examples and supporting citations. Solid answers will get full points, poor-quality or missing elements for answers will cost -1 or -2 points each, missing answers completely will be -5 points.

Question 3 should provide a UML class diagram that could be followed to produce the FNCD simulation program in Java. This includes identifying major contributing or communicating classes (ex. Staff, Vehicles, etc.) and any methods or attributes found in their design. As stated, multiplicity and accessibility tags are optional. Use any method reviewed in class to create the diagram that provides a readable result, including diagrams from graphics tools or hand drawn images. A considered, complete UML diagram will earn full points, poorly defined or clearly missing elements will cost -1 to -2 points, missing the diagram is -5 points.

Part 2.2 is worth 35 points and is due Wednesday 2/15 at 8 PM. The submission will be a URL to a GitHub repository. The repository should contain well-structured OO Java code for the FNCD simulation, the captured text file with program results requested, and a README file that has the names of the team members, the Java version, and any other comments on the work – including clearly documenting any assumptions or interpretations of the problem. *Only one URL submission is required per team.*

15 points for comments and readable OO style code: Code should be commented appropriately, including citations (URLs) of any code taken from external sources. We will also be looking for clearly indicated comments for the six OO terms to be illustrated in the code. A penalty of -1 to -2 will be applied for instances of poor or missing comments or excessive procedural style code (for instance, executing significant procedural logic in main).

10 points for correctly structured output: The output from the single and multiple runs captured in the text files mentioned for the exercise should be present. A penalty of -1 to -2 will be applied for each incomplete or missing output file.

5 points for the README file: A README file with names of the team members, the Java version, and any other comments about your implementation or assumptions should be present in the GitHub repo. Incomplete/missing READMEs will be penalized -2 to -5 points.

5 points for the updated UML file from project 2.1 as described, including how the structure changed between projects 2.1 and 2.2. Incomplete or missing elements in the UML diagram will be penalized -1 to -2 points.

Please ensure all class staff are added as project collaborators to allow access to your private GitHub repository. Do not use public repositories. Do use a .gitignore file (as discussed in lecture) to prevent your repo from clogging with unneeded files.

Overall Project Guidelines

Assignments will be accepted late for four days. There is no late penalty within 4 hours of the due date/time. In the next 48 hours, the penalty for a late submission is 5%. In the next 48 hours, the late penalty increases to 15% of the grade. After this point, assignments will not be accepted.

Use e-mail or Piazza to reach the class staff regarding homework/project questions, or if you have issues in completing the assignment for any reason.