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 across the team. In any case, since the grade you earn for the project is for the whole team, the team should review the submission before it is turned in.

Part 1: OO and UML exercises – 20 points

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

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

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

3. (10 points) Draw a UML class diagram for the FNPS simulation described in part 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.

Part 2: FNPS simulation – 30 points

This is the first of three pet store simulation projects. You'll reuse all or part of this code in projects 3 and 4. I will provide my approach to the project 2 simulation for your consideration in class after you complete this.

Your team will create a Java program to simulate the daily operations of friendly neighborhood pet stores (FNPS). A store contains a variety of animals and supplies, known in the simulation as items. The inventory for a store will have subclass instances that describes the particular pet or supply item further:

- Item (name, purchasePrice, listPrice, salePrice, dayArrived, daySold)
    - Pet (breed, age, healthy)
        - Dog (size, color, housebroken, purebred)
        - Cat (color, housebroken, purebred)
        - Bird (size, mimicry, exotic, papers)
    - Supplies
        - Food (size, animal, type)
        - Leash (animal)
        - Toy (animal)
        - Cat Litter (size)

You should use a multi-level inheritance structure for these objects as shown. If an object is followed by words in parenthesis, these are attributes particular to that subclass. You can determine values for any attributes not specified in the directions. For instance, each item needs a name, but you can determine how you want to name item instances (if not specified).

The store's inventory items (pets and supplies) should be initialized before the simulation starts. The initialization will put three instances of each lowest subclass (e.g. Dog, Toy) in the store prior to starting. Determine a name (any way you choose) and a purchase price ($1 to $100) per item. The list price will be set to 2 x the purchase price for each item. The attribute dayArrived should be set to 0 for the initial item set. Other

attributes can be initialized as you see fit. The attributes named healthy, housebroken, purebred, mimicry, exotic, papers are all intended to be boolean. All animals are initially healthy, other attributes can be set as you like. The attributes salePrice and daySold will only be set when the Inventory instance is sold.

The store keeps all money for buying and selling Inventory in a cash register. At the start of the simulation, the amount of money in the register is $0. The store is open for business every day of the simulation.

The store has two types of employees, clerks and trainers. There will be two clerks and two trainers working for the store, but on a given simulated day, only one of a randomly selected instance of each employee type will come into work. You should name your clerks and trainers. Clerks and trainers will not work more than three days in a row, so if you randomly select an employee for a fourth consecutive day of work, you should switch to the other employee of that type.

On each day the simulation runs after the store opens, the following actions will be taken by the employees that are working that day:

ArriveAtStore: Each active employee will announce their arrival at the FNPS. All announcements should be text output to the console and should look something like: “Bob the Clerk arrives at the store on Day 4.”

ProcessDeliveries: The clerk should see if new inventory (Pets or Supplies) has arrived. (This can be caused by the PlaceAnOrder action later on.) If so, they should be announced and added to the store’s inventory. Arriving items should update the dayArrived attribute to this day.

FeedAnimals: The trainer will feed all the animals in two collections – the store’s inventory of healthy pets and its collection of sick pets, announcing the visit to each animal. While feeding store inventory animals, there is a 5% chance that an animal’s healthy attribute may change to false. If so, this animal will be removed from the store inventory and be placed in the collection of sick pets. The trainer will also feed each of the pets in the sick pet collection, and there is a 25% chance that the animal’s healthy attribute will change to true when visited. If that happens, the trainer will move the animal to the store inventory.

CheckRegister: The clerk will count and announce the amount of money found in the register. If there is insufficient money in the register, the Clerk must perform the GoToBank Action (next) before going on to the DoInventory Action.

GoToBank: This action is only performed if there is less than $200 in the register. The Clerk will go to the bank, withdraw $1000, and put the money in the store register with an announcement. The amount of money withdrawn from the bank in this manner should be tracked.

DoInventory: The clerk will add up the value of all the items in the store inventory (based on purchase price, not including sick animals) and announce that total value. If any of the item subclasses has a count of 0 (for instance, there are 0 Dog items in inventory), the Clerk must perform the PlaceAnOrder action (next) for each missing item type before going on to the OpenTheStore Action. All activity should be announced.

PlaceAnOrder: For any subclass item that is at 0 inventory, the clerk will order 3 of those items, each with a random purchase price and other randomly determined characteristics (just as when adding items on Day 0). The purchase price of each of these items should be paid for by removing the funds from the Cash Register. These items should arrive at the store in the next 1 to 3 days, to be found by the clerk in the ArriveAtStore step, at which point the clerk will put them into the store inventory. All activity should be announced.

OpenTheStore: The store will now respond to arriving customers. Customers will come in to buy an inventory item, a Pet or Supplies. There will be 3 to 10 buying customers each day. All actions taken by customers should be announced. Customers will examine each item in inventory (healthy Pets and Supplies) until they randomly (10% chance) decide to try to buy an item. If there is an item they want to buy, there is a 50% chance they will pay the clerk the listPrice for the item. If they do not buy the item, the trainer will discuss the item with the customer and offer them a 10% discount to the listPrice. There will then be a 75% chance they will buy the item they selected at that price. If they do not buy the item after that, they will continue looking at remaining inventory items until they finally buy something or leave the store. When an item is sold it should be moved from the store inventory into a sold items collection, and the daySold and salePrice should be updated. The money paid by the customer should go into the Cash Register.

Announcements should make clear what happens during the process, for example:

- "Dawn the Trainer sold a Dog to Customer 3 for $13 after a 10% discount."
- "Customer 6 wanted to buy Food but no items were in inventory, so they left."

CleanTheStore: Before closing, the trainer will clean the animal cages (for healthy and sick animals). There is a 5% chance that the trainer may accidentally have an animal escape. Randomly, either the clerk or the trainer will catch the Pet and return it to its cage. The clerk will vacuum the store as part of cleaning. All events should be announced.

LeaveTheStore: The clerk will lock up the store and both employees will go home for the day (which should be announced).

Simulate the running of the FNPS store for 30 days. At the end of the 30 days, print out a summary of the state of the simulated store, including:

- the items left in inventory and their total value (purchasePrice)
- the items sold, including the daySold and the salePrice, with a total of the salePrice
- the pets remaining in the sick pets collection
- the final count of money in the Cash Register
- how much money was added to the register from the GoToBank Action

There may be possible error conditions due to insufficient funds or lack of inventory that you may need to define policies for and then check for their occurrence. You may find requirements are for this simulation are not complete in all cases. In such cases, either document your interpretation or find class staff for clarifications.

The code should be in Java 8 or higher and should be object-oriented in design and implementation. **In commenting the code, point out at least one example of: Inheritance, Polymorphism, Cohesion, Identity, Encapsulation, and Abstraction.**

Capture all output from a single simulation run in your repository in a text file called Output.txt (by writing directly to it or by cutting/pasting from a console run).

Grading Rubric:

Homework/Project 2 is worth 50 points total and is due Wednesday 6/15 at 8 PM.

The submission for Part 1 will be a single PDF per team labeled "PartOne.pdf". The PDF must contain the names of all team members and be easily found in the repository.

Questions 1-2 will be graded on the quality (not quantity) of the answer. Questions one and two should include examples and citations. Solid answers will get full points, poor quality or missing citations 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 FNPS simulation program in Java. This includes identifying major contributing or communicating classes (ex. Items, Employees, 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 -10 points.

The Java code submission will be submitted via the URL to the containing GitHub repository. The repository should contain well-structured OO Java code for the simulation, a captured Output.txt text file with program results, 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 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 a run captured in the text file mentioned per exercise should be present. A penalty of -1 to -2 will be applied per exercise for incomplete or missing output.

5 points for the README file: A README markdown 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.

Please ensure all class staff are added as project collaborators (brmcu, gayathripadmanabhan1, wmaxdonovan-cu) to allow access to your private GitHub repository. Do not use public repositories.

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.