This assignment is intended to be done by your team of two to three 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.

**Part 1: OO and UML exercises – 20 points**

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

1. (5 points) What does "design by contract" mean in an OO program? What is the difference between implicit and explicit contracts? Provide a text, pseudo code, or code example of each contract type.

2. (5 points) What are three ways Java interfaces differ from the pure OO interface intent that describes only function signatures and return types to be implemented? Provide a Java code example of each.

3. (10 points) Draw a class diagram for the FLGS simulation described in part 2. The class diagram should contain any classes, abstract classes, or interfaces you plan to implement. 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 this UML diagram should be a team activity if possible.

**Part 2: FLGS simulation – 30 points**

Your team will create a Java program to simulate the daily operations of a friendly local game store (FLGS). The store contains the following hierarchy of game(-related) inventory items:

- Games
  - Family
    - Monopoly, Clue, Life
  - Kids
    - Mousetrap, Candyland, Connect Four
  - Card
    - Magic, Pokémon, Netrunner
  - Board
    - Catan, Risk, Gloomhaven

(You should use a two- or three-level inheritance structure for these objects, even though right now, there is not much difference between the types and subtypes. All project 2 code will be reused in the next two projects.)

At the low level of the hierarchy (Clue, Risk, etc.) each type of game has a different price between $5 and $100; a box height, width, and length (in inches); the number in inventory, the number sold; and the current shelf position (from 1 to the number of different games). The initial number of each game in inventory is 3, the initial number sold is 0. You can assign game sizes and prices as you wish – randomly or by selecting values. The Cash Register for the store begins empty ($0).

The game store also has two Employees, both of which are Employee subtype Cashier, and are named Burt and Ernie. Each day, there is a 50% chance one of the two will be working, determined randomly.

Each day, the Cashier that is working will perform the following tasks:

- Arrive (at the Store) – for this task, a Cashier will announce they have arrived on a given day
  - Announce means print a message, such as "Burt the Cashier has arrived at the store on Day 7"

    - There may be new games delivered from the previous day (see the Order step); if there are, the inventory for those games should be increased by the number that arrived and the arrival of new games should be announced
- Count (the Money) – for this task, the Cashier will look at how much money is in the Cash Register and announce how much money is there; if the Cash Register money drops below $100, add $1000 to the Cash Register and announce that the money was added
- Vacuum (the Store) – for this task, the Cashier will announce they are vacuuming the store; there is a chance that a random game will be damaged by the Cashier (5% for Ernie, 10% for Burt) – if that happens the Cashier will announce what game was damaged, take the damaged game out of inventory and put it in a Damaged Game container
- Stack (the Games) – for this task, the Cashier will stack the games on the shelves – Ernie likes to stack from shortest to tallest total game pile height, Burt likes to stack the widest games first down to the narrowest; the Cashier will announce each stack as it is placed – for example:
    - "Ernie stacks 2 Magic games in shelf position 4 (pile height = 14")" or
    - "Burt stacks 3 Gloomhaven games in shelf position 7 (game width = 36")"
- Open (the Store) – for this task, the Cashier will announce the store is open; 0 to 4 random Customers will arrive, and will begin looking at games in the shelf positions
    - Customers will have a 20% chance of buying a game from shelf position 1; 18% from shelf position 2, 16% from position 3, etc.; note shelf positions may become empty
    - If a customer decides to buy a game, add the price of the game to the Cash Register, reduce the inventory of that game by 1, and increase the number sold by 1; each customer will buy 0 to 2 games in a visit
    - Announce any games purchased or any customers that did not buy a game
        - Ex: "Burt sold a Mousetrap game to customer 1 for $10."
- Order (new games) – for this task, the Cashier will announce and then order an overnight delivery of 3 of any games that have an inventory of 0; pay for these new games by removing ½ the cost of each game ordered from the Cash Register; these games will arrive at the store the next morning
- Close (the store) – for this task, announce the cashier is leaving and the store is closed

Simulate the running of the store for 30 days. At the end of the 30 days, for each game type, list the number in inventory, the number sold, and the total sales for that game type. Also list the contents of the Damaged Game container, the final count of money in the Cash Register, as well as how many times money was added to the register due to low funds in the Count the Money step.

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).

Also include in your repository an updated version of the FLGS UML diagram from part 1 that matches your actual implementation in part 2. Note what changed between part 1 and part 2 (if anything) in a comment paragraph.

**Grading Rubric:**

**Homework/Project 2 is worth 50 points total.**

**Part 1 is worth 20 points and is due on Wednesday 9/15 at 8 PM.** The submission will be a single PDF per team. The PDF must contain the names of all team members.

Questions 1 and 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 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 FLGS simulation program in Java. This includes identifying major contributing or communicating classes (ex. Games, Employees) 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 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.

**Part 2 is worth 30 points and is due Wednesday 9/22 at 8 PM.** The submission will be a URL to a 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 any assumptions or interpretations of the problem. Only one URL submission is required per team.

10 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 program logic in main).

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

10 points for the updated UML file from part 1 as described. Incomplete or missing elements in the UML diagram will be penalized -2 to -4 points.

Please ensure all class staff are added as project collaborators 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.