CS-2340-B/GR Midterm Exam - Diagramming Part

Vidit Dharmendra Pokharna, Jon Ruogu Wang, Ananth Ram Vivekanand, Nandini Ramakrishnan, Dhruv Raj Adha, Shivansh Sharma

TOTAL POINTS

47.5 / 60

QUESTION 1

1 Sequence Diagram **20.5 / 25**

✓ **+ 2.5 pts** *At least 1 actor - shown correctly is stick figure*

✓ **+ 5 pts** *At least 1 loop fragment/frame*

✓ **+ 5 pts** *At least 1 alternate (alt) fragment/frame*

✓ **+ 2.5 pts** *At least 1 external system*

  + 2.5 pts Activation boxes for objects

✓ **+ 2.5 pts** *Correctly labeled instances of the class*

✓ **+ 5 pts** *Messages sent to correct objects*

✓ **- 1 pts** *Minor messages error*

  - 2 pts Major messages error

✓ **- 1 pts** *minor syntax error*

  - 2 pts major syntax error

  - 2 pts Extra objects (unnecessary for Grocery Store System)

  + 0 pts Includes team members that contributed

  💬 classes know to send information even when a message is not sent to them

QUESTION 2

2 Design Class Diagram (DCD) **17 / 25**

✓ **+ 5 pts** *an aggregation relationship*

✓ **+ 5 pts** *a composition relationship*

✓ **+ 2.5 pts** *inheritance relationship*

✓ **+ 2.5 pts** *relevant methods*

✓ **+ 2.5 pts** *relevant attributes*

✓ **+ 5 pts** *multiplicity relationship*

✓ **+ 2.5 pts** *appropriately applied visibility indicators on attribute*

✓ **- 1 pts** *minor syntax error*

  - 2.5 pts major syntax error

✓ **- 1 pts** *Minor attribute/method error*

  + 0 pts Includes team members that contributed

✓ **- 6 pts** *Design doesn't uphold conceptual integrity (DCD and SD do not match)*

  💬 attributes unlabelled, multiplicities need 1..* notation, not 1...

QUESTION 3

3 Use Case Diagram **10 / 10**

✓ **+ 5 pts** *at least 5 use cases*

✓ **+ 2.5 pts** *2+ actors*

✓ **+ 1.25 pts** *correctly uses at least 1 `<<extends>>` relationship*

✓ **+ 1.25 pts** *correctly uses at least 1 `<<includes>>` relationship*

  - 1 pts minor syntax/conceptual issue

  - 2 pts major syntax/conceptual issue

  + 5 pts Includes team members that contributed

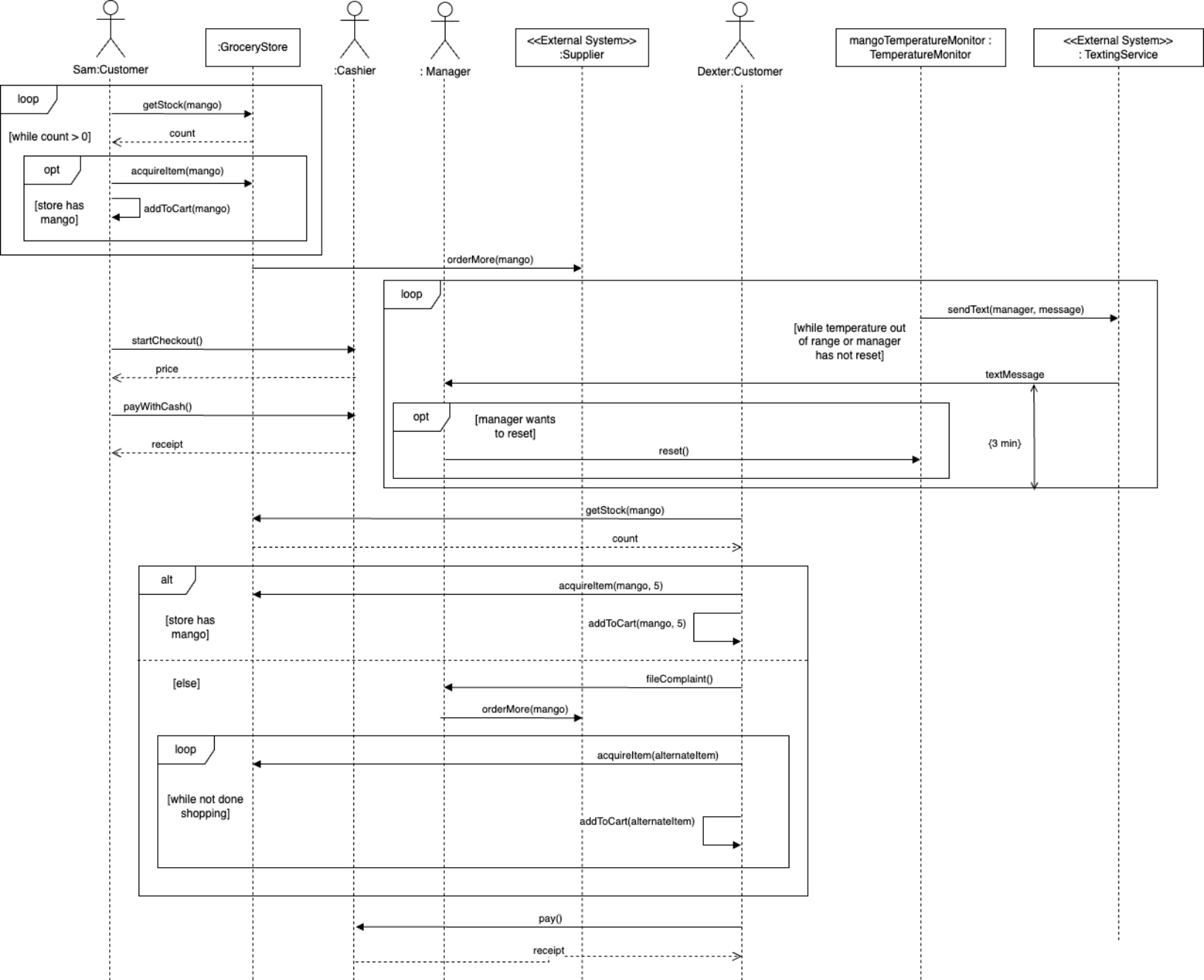## *4* Teamwork **0 / 0**

   ✓ **+ 0 pts** *Complete*

     - 6 pts Doesn't include who did what question
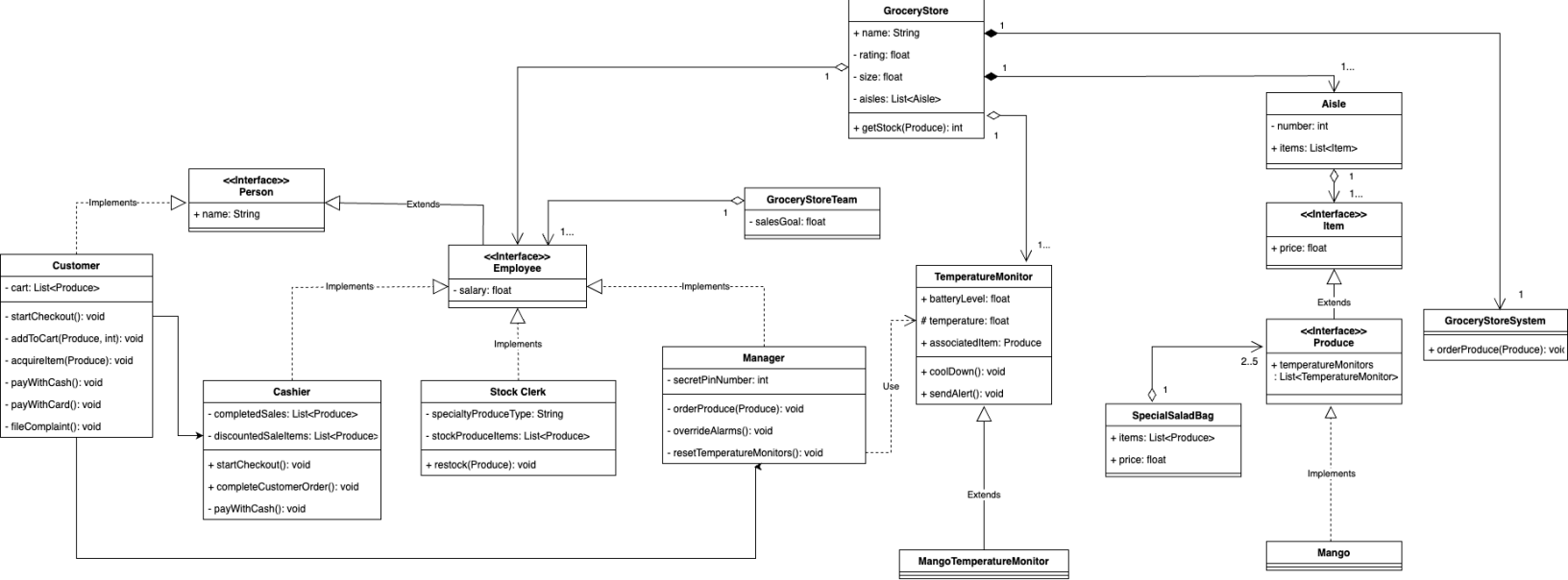
     - 6 pts Doesn't include team task delegation strategy

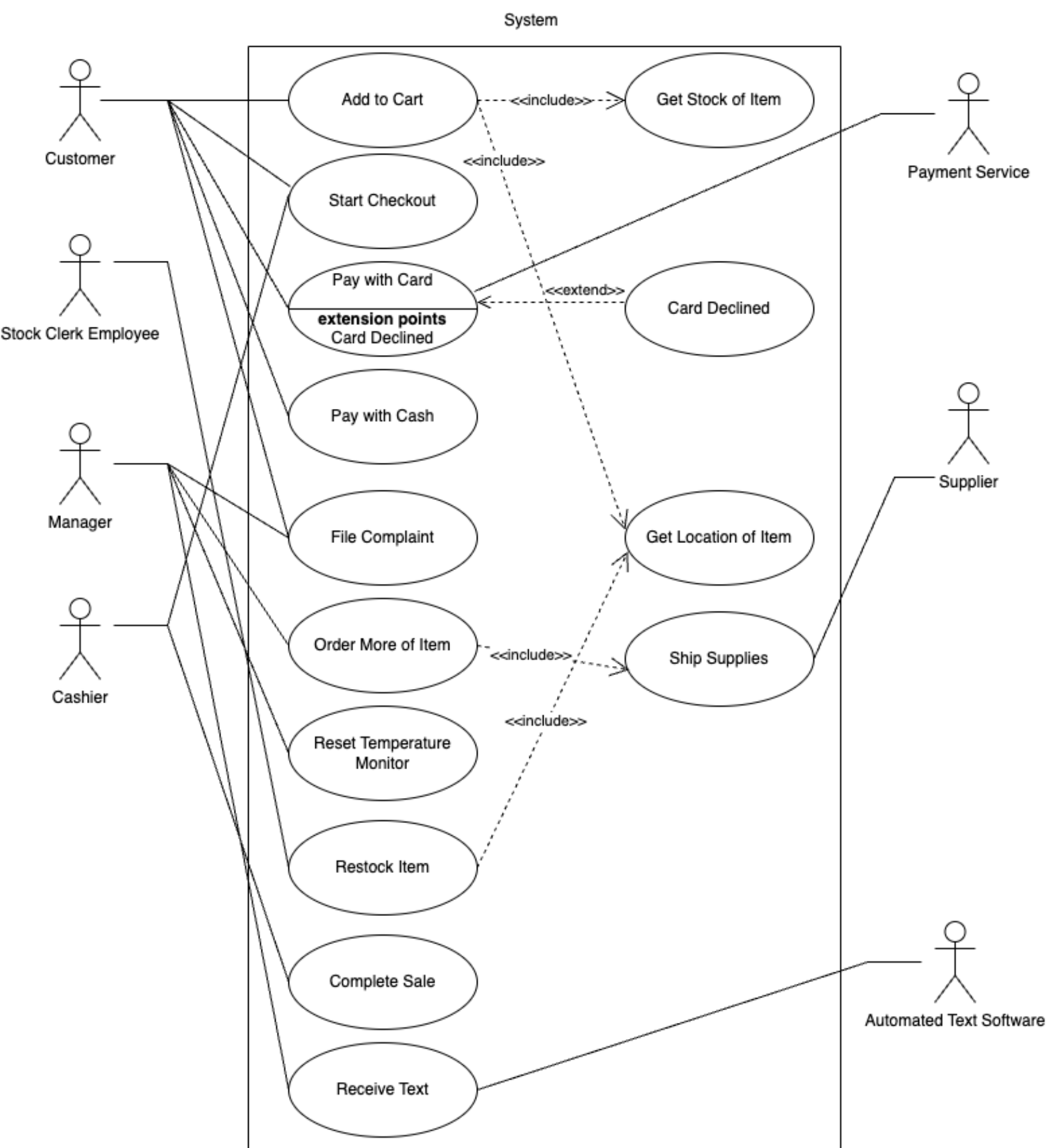   **- 1 pts** All team members did not submit the assignment as a group

This is a UML sequence diagram.

Lifelines (participants):
- Sam:Customer
- :GroceryStore
- :Cashier
- : Manager
- <<External System>> :Supplier
- Dexter:Customer
- mangoTemperatureMonitor : TemperatureMonitor
- <<External System>> : TextingService

loop [while count > 0]
- Sam:Customer → :GroceryStore : getStock(mango)
- :GroceryStore --> Sam:Customer : count
- **opt** [store has mango]
  - Sam:Customer → :GroceryStore : acquireItem(mango)
  - :GroceryStore → :GroceryStore : addToCart(mango)

- :GroceryStore → :Supplier : orderMore(mango)

loop [while temperature out of range or manager has not reset]
- mangoTemperatureMonitor → TextingService : sendText(manager, message)
- TextingService → : Manager : textMessage {3 min}
- **opt** [manager wants to reset]
  - : Manager → mangoTemperatureMonitor : reset()

- Sam:Customer → :Cashier : startCheckout()
- :Cashier --> Sam:Customer : price
- Sam:Customer → :Cashier : payWithCash()
- :Cashier --> Sam:Customer : receipt

- Dexter:Customer → :GroceryStore : getStock(mango)
- :GroceryStore --> Dexter:Customer : count

alt [store has mango]
- Dexter:Customer → :GroceryStore : acquireItem(mango, 5)
- :GroceryStore → :GroceryStore : addToCart(mango, 5)

[else]
- Dexter:Customer → :Cashier : fileComplaint()
- :Cashier → : Manager : orderMore(mango)
- **loop** [while not done shopping]
  - Dexter:Customer → :GroceryStore : acquireItem(alternateItem)
  - :GroceryStore → :GroceryStore : addToCart(alternateItem)

- Dexter:Customer → :Cashier : pay()
- :Cashier --> Dexter:Customer : receipt

1 Sequence Diagram **20.5 / 25**

✓ **+ 2.5 pts** *At least 1 actor - shown correctly is stick figure*

✓ **+ 5 pts** *At least 1 loop fragment/frame*

✓ **+ 5 pts** *At least 1 alternate (alt) fragment/frame*

✓ **+ 2.5 pts** *At least 1 external system*

+ 2.5 pts Activation boxes for objects

✓ **+ 2.5 pts** *Correctly labeled instances of the class*

✓ **+ 5 pts** *Messages sent to correct objects*

✓ **- 1 pts** *Minor messages error*

- 2 pts Major messages error

✓ **- 1 pts** *minor syntax error*

- 2 pts major syntax error

- 2 pts Extra objects (unnecessary for Grocery Store System)

+ 0 pts Includes team members that contributed

💬 classes know to send information even when a message is not sent to them

GroceryStore
+ name: String
- rating: float
- size: float
- aisles: List<Aisle>

+ getStock(Produce): int

<<Interface>> Person
+ name: String

GroceryStoreTeam
- salesGoal: float

<<Interface>> Employee
- salary: float

Aisle
- number: int
+ items: List<Item>

<<Interface>> Item
+ price: float

GroceryStoreSystem
+ orderProduce(Produce): void

Customer
- cart: List<Produce>

- startCheckout(): void
- addToCart(Produce, int): void
- acquireItem(Produce): void
- payWithCash(): void
- payWithCard(): void
- fileComplaint(): void

Cashier
- completedSales: List<Produce>
- discountedSaleItems: List<Produce>

+ startCheckout(): void
+ completeCustomerOrder(): void
- payWithCash(): void

Stock Clerk
- specialtyProduceType: String
- stockProduceItems: List<Produce>

+ restock(Produce): void

Manager
- secretPinNumber: int

- orderProduce(Produce): void
- overrideAlarms(): void
- resetTemperatureMonitors(): void

TemperatureMonitor
+ batteryLevel: float
# temperature: float
+ associatedItem: Produce

+ coolDown(): void
+ sendAlert(): void

<<Interface>> Produce
+ temperatureMonitors
: List<TemperatureMonitor>

SpecialSaladBag
+ items: List<Produce>
+ price: float

MangoTemperatureMonitor

Mango

Implements

Extends

Use

Extends

Implements

Implements

Implements

1...

2..5

1

2 Design Class Diagram (DCD) **17 / 25**

✓ **+ 5 pts** *an aggregation relationship*

✓ **+ 5 pts** *a composition relationship*

✓ **+ 2.5 pts** *inheritance relationship*

✓ **+ 2.5 pts** *relevant methods*

✓ **+ 2.5 pts** *relevant attributes*

✓ **+ 5 pts** *multiplicity relationship*

✓ **+ 2.5 pts** *appropriately applied visibility indicators on attribute*

✓ **- 1 pts** *minor syntax error*

　- 2.5 pts major syntax error

✓ **- 1 pts** *Minor attribute/method error*

　+ 0 pts Includes team members that contributed

✓ **- 6 pts** *Design doesn't uphold conceptual integrity (DCD and SD do not match)*

💬　attributes unlabelled, multiplicities need 1..* notation, not 1...

3 Use Case Diagram **10 / 10**

✓ **+ 5 pts** *at least 5 use cases*

✓ **+ 2.5 pts** *2+ actors*

✓ **+ 1.25 pts** *correctly uses at least 1 `<<extends>>` relationship*

✓ **+ 1.25 pts** *correctly uses at least 1 `<<includes>>` relationship*

- 1 pts minor syntax/conceptual issue

- 2 pts major syntax/conceptual issue

+ 5 pts Includes team members that contributed

gradescope

Which team member worked on which diagram? What was your team task delegation strategy (i.e., how did you work on this exam as a team)? Please note anyone that did not contribute.

Our strategy was to delegate each diagram to two members:
- Diagram 1: Nandini and Ananth
- Diagram 2: Vidit and Dhruv
- Diagram 3: Jon and Shivansh

After completing each diagram to the extent each group could attempt, we all deliberated as a team on each diagram. Each group presented their diagram, followed by a feedback loop from the other team members to the group that worked on the diagram. This helped complete the diagram with utmost deliberation and a voice from each member. Hearing from everyone, we completed our diagrams to represent our understanding of the scenarios at hand.

✓ **+ 0 pts** *Complete*

    - 6 pts Doesn't include who did what question

    - 6 pts Doesn't include team task delegation strategy

    - 1 pts All team members did not submit the assignment as a group