

# Pattern Exercise

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 18

# Acknowledgement & Materials Copyright

- I'd like to start by acknowledging Dr. Ken Anderson
- Ken is a Professor and the Chair of the Department of Computer Science
- Ken taught OOAD on several occasions, and has graciously allowed me to use his copyrighted material for this instance of the class
- Although I will modify the materials to update and personalize this class, the original materials this class is based on are all copyrighted © Kenneth M. Anderson; the materials are used with his consent; and this use in no way challenges his copyright

# A Lecture Break

I feel like we all deserve a lecture break after our pattern lectures lectures, so here's a thing...

# How well do you know your patterns?

- Prepare yourself...
  - Clear away all notes and connected machinery (other than Zoom)
  - You need paper and a writing thing
- 
- Please remember – if you're not here participating in person, or you are, and it doesn't go well, there will be other avenues for extra credit and participation – please, don't panic.

# Task 1

- Draw the UML Class Diagram for the Strategy Pattern

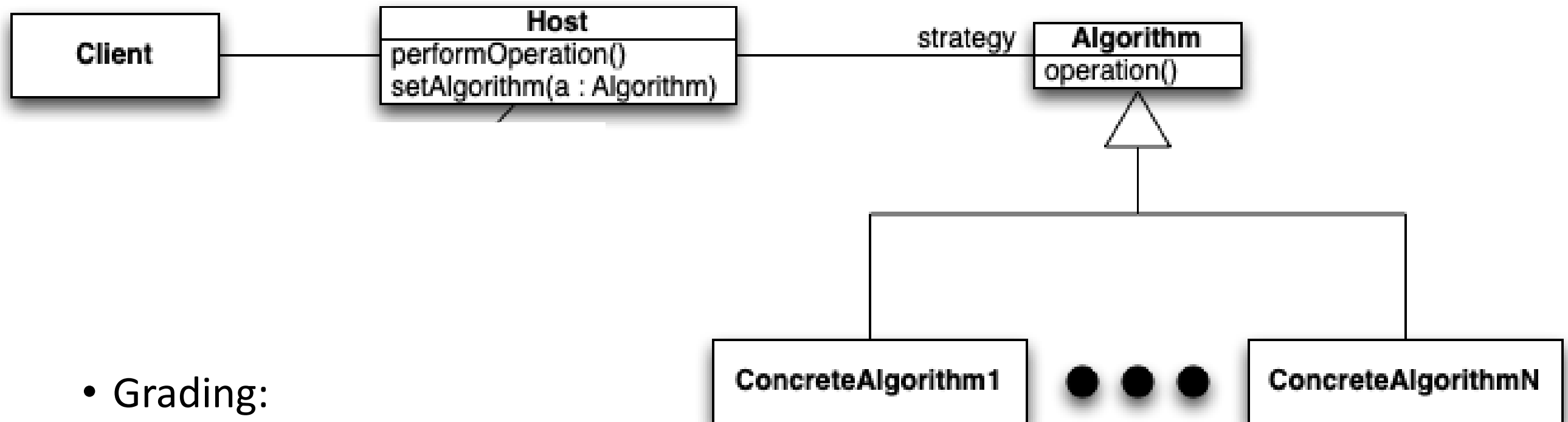
AND

- Complete this OO Principle
- Favor \_\_\_\_\_ over \_\_\_\_\_

# Grading the tasks

- You'll be grading your own work here...
- Patterns
  - The drawings can be arranged differently – look for parts and connections
  - Exact match or really, really close = Perfect
  - Not bad, missed a couple of things = Close
  - Otherwise = A Duck
- Phrases/Definitions
  - must be right (spelling doesn't count) to get the point
- I can intervene in grading if you're concerned...

# Task 1 – Strategy Answers



- Grading:
- Perfect! = 2 points
- Close! = 1 point
- Looks like a drawing of a duck = 0 points
- Favor Delegation (or Composition) over Inheritance.
- 1 Point if right, 0 if wrong or somehow rude

## Task 2

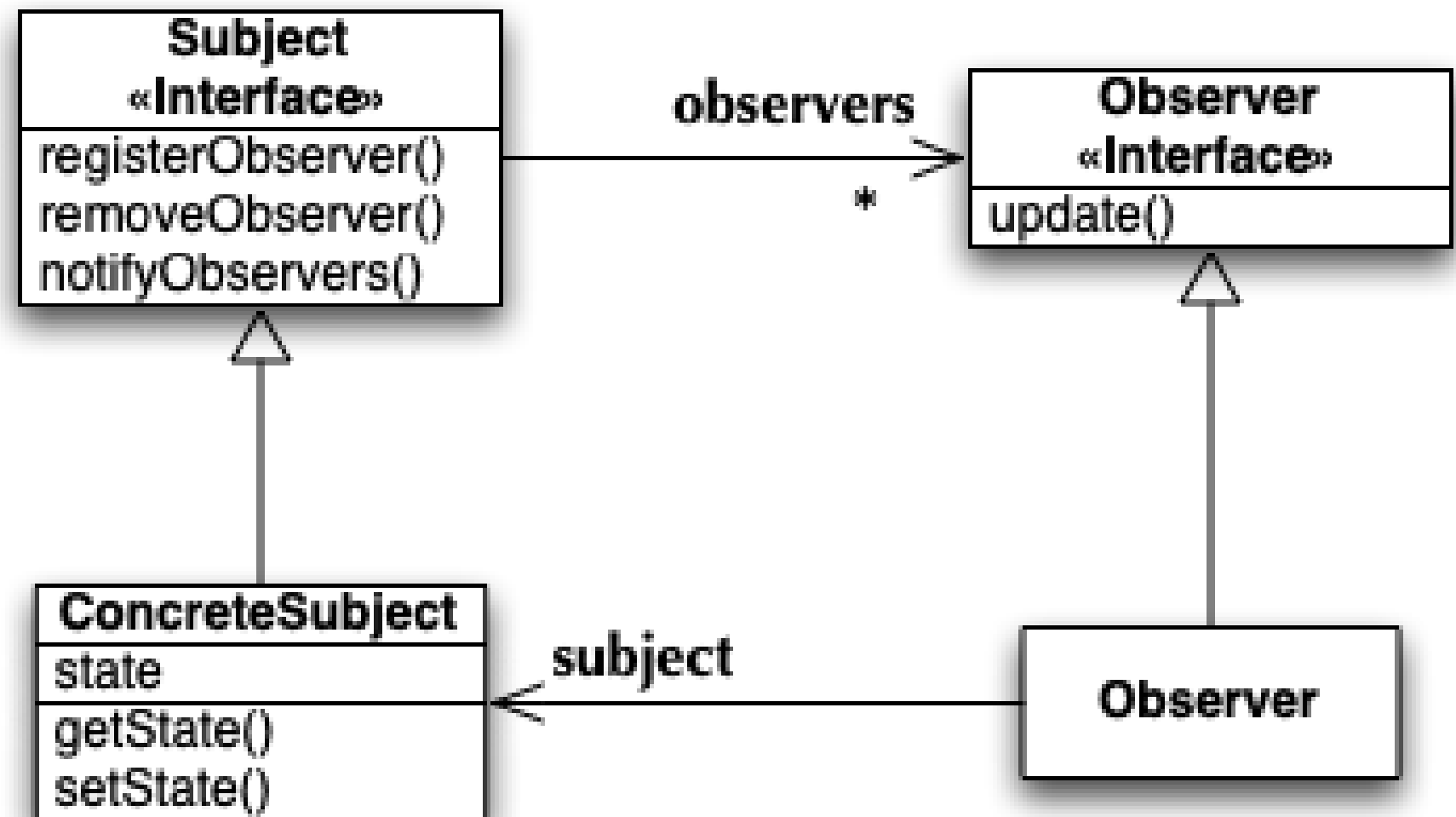
- Draw the UML Class Diagram for the Observer Pattern

AND

- In Java, \_\_\_\_\_ is an interface, but \_\_\_\_\_ is a class, which can cause problems.



## Task 2 – Observer Answers



- Grading:
- Perfect! = 2 points
- Close! = 1 point
- Duck issues = 0 points
- In Java originally, Observer is an interface, but Observable is a class, which didn't work so well...
- 1 Point if right, 0 if otherwise

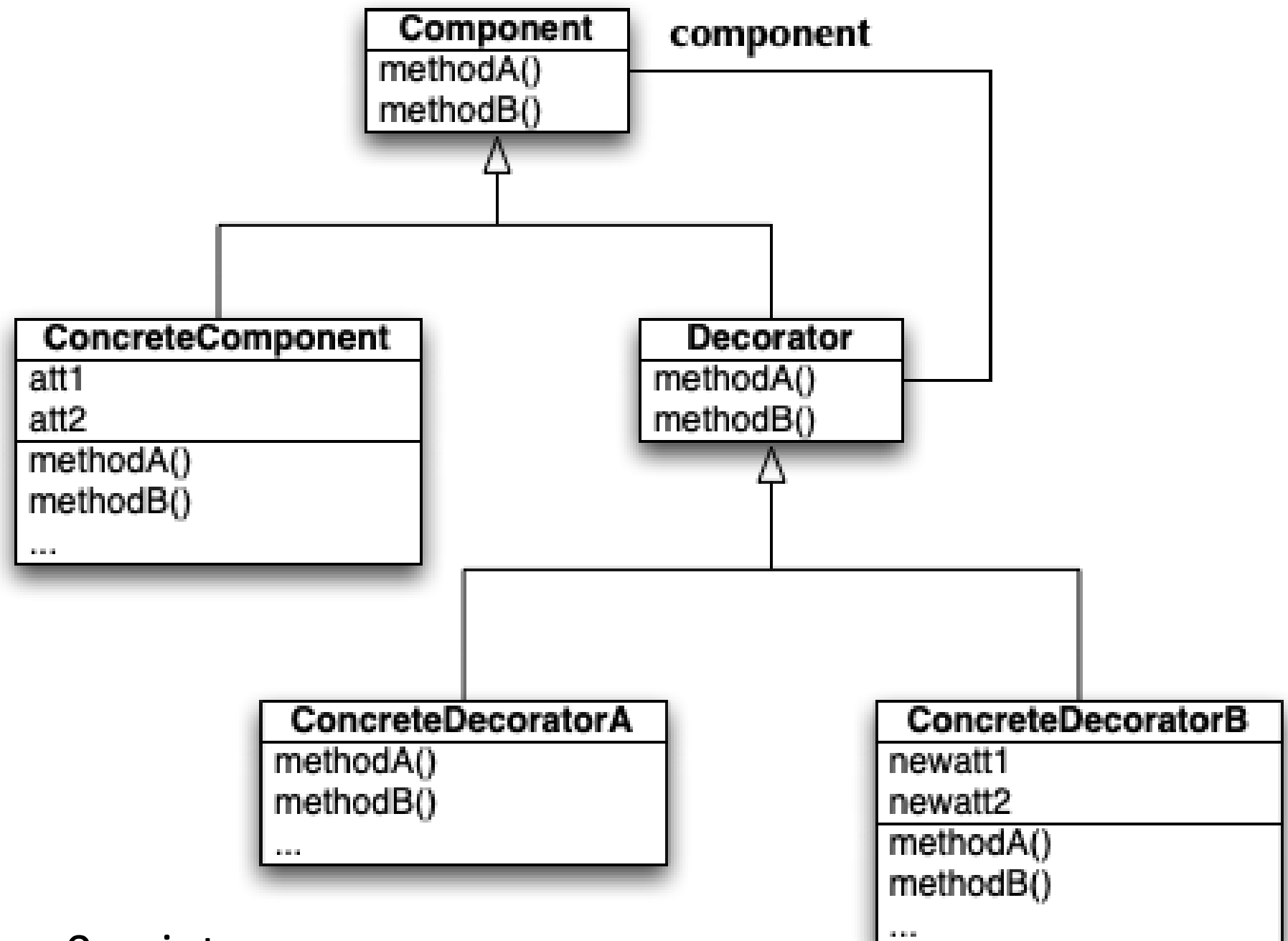
# Task 3

- Draw the UML Class Diagram for the Decorator Pattern

AND

- Open Closed Principle: Classes should be open for \_\_\_\_\_ but closed to \_\_\_\_\_

## Task 3 – Decorator Answers



- Grading:
  - Perfect! = 2 points
  - Close! = 1 point
  - Definitely duck-like = 0 points
- 
- Open Closed Principle: Classes should be open for extension but closed to modification
  - 1 Point if right, 0 if not

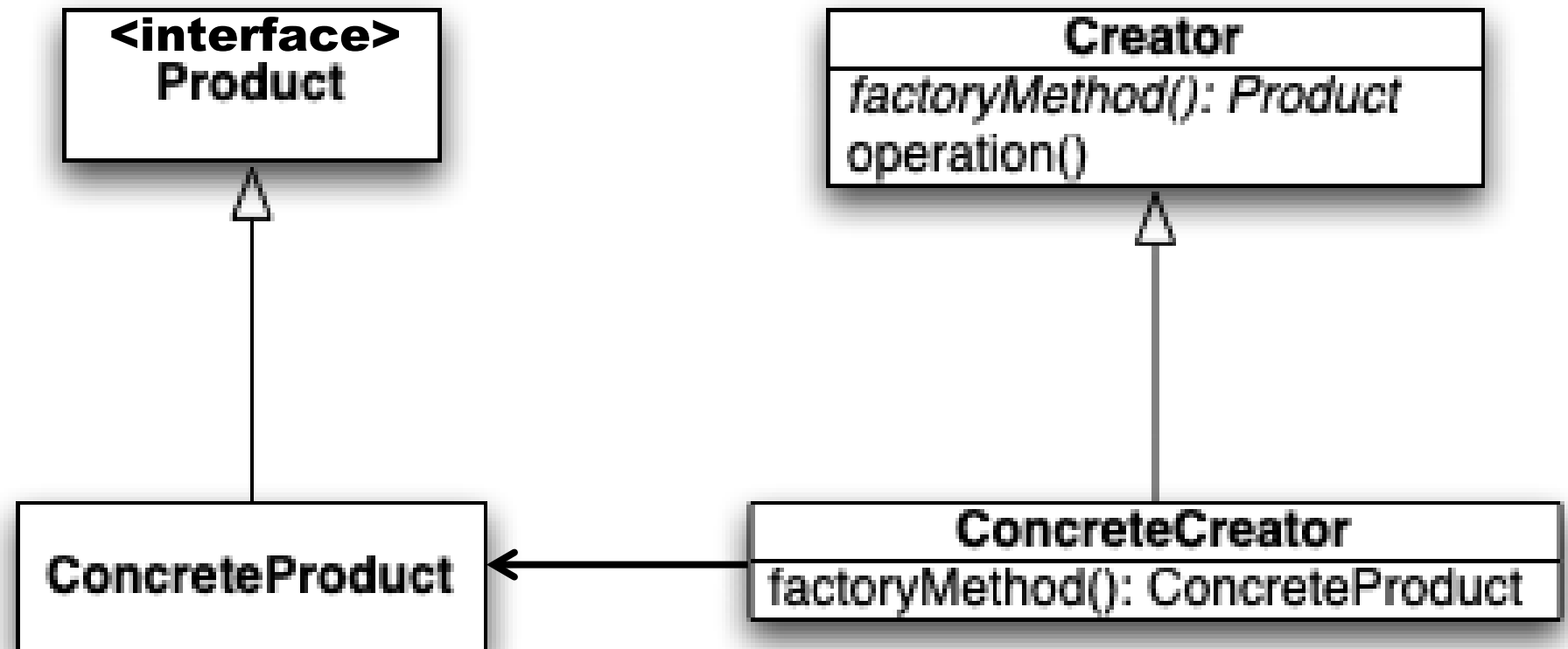
# Task 4

- Draw the UML Class Diagram for the Factory Pattern (not simple factory, not abstract factory)

AND

- Dependency Inversion Principle: Depend upon \_\_\_\_\_. Do not depend upon \_\_\_\_\_.

## Task 4 – Factory Answers



- Grading:
  - Perfect! = 2 points
  - Close! = 1 point
  - A bit too ducky = 0 points
- 
- Dependency Inversion Principle: Depend upon abstractions. Do not depend upon concrete classes.
  - 1 Point if right, 0 if otherwise

# Possible Points

- Strategy 3
- Observer 3
- Decorator 3
- Factory 3
- 12 points? Who's got it?
- Do we have a tie for first?

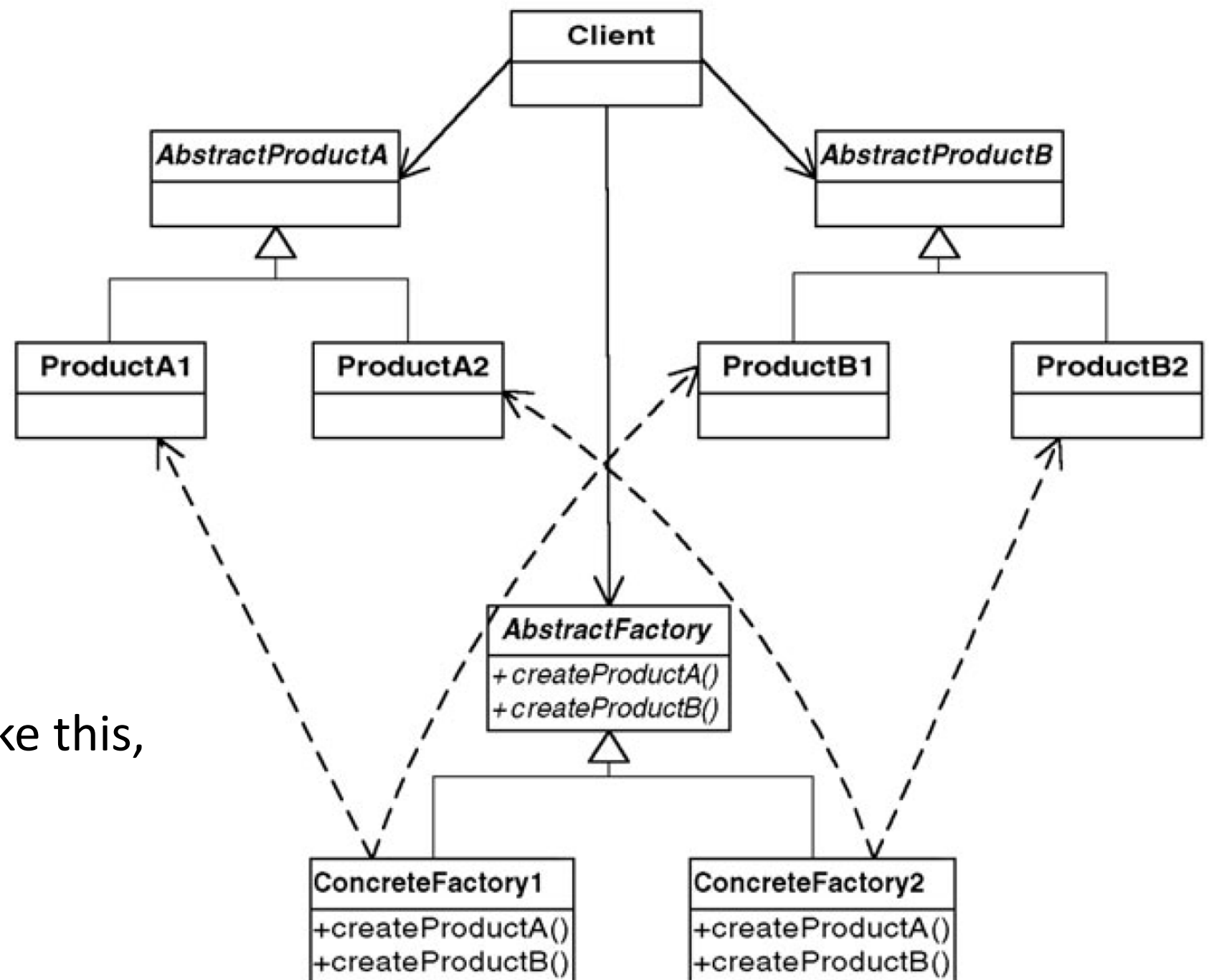
# Tiebreaker Task... Abstract Factory

- Draw the UML Class Diagram for the Abstract Factory Pattern

AND

- Factories build objects with \_\_\_\_\_, abstract factories build families of objects with \_\_\_\_\_.

# Tiebreaker – Abstract Factory Answers



- Grading:
  - May not look exactly like this, look for the parts!
  - Perfect! = 2 points
  - Close! = 1 point
  - All I see is duck = 0 points
- 
- Factories build objects with inheritance; abstract factories build families of objects with (object) composition.
  - 1 Point if right, 0 if otherwise



# Close It Up; Bring It In

- Top Scoring Player(s) – 2 Quiz Points
- Next Level Score(s) – 1 Quiz Points
- Thanks for playing!
- You may need to be able to reproduce diagrams or principles on an exam – but you'll have a sheet of note paper to help...