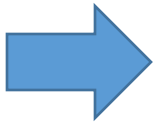
Expression Tree Code Demo (cont.)

Cpt S 321

Washington State University

Pointers for solutions ExpressionTreeCodeDemo

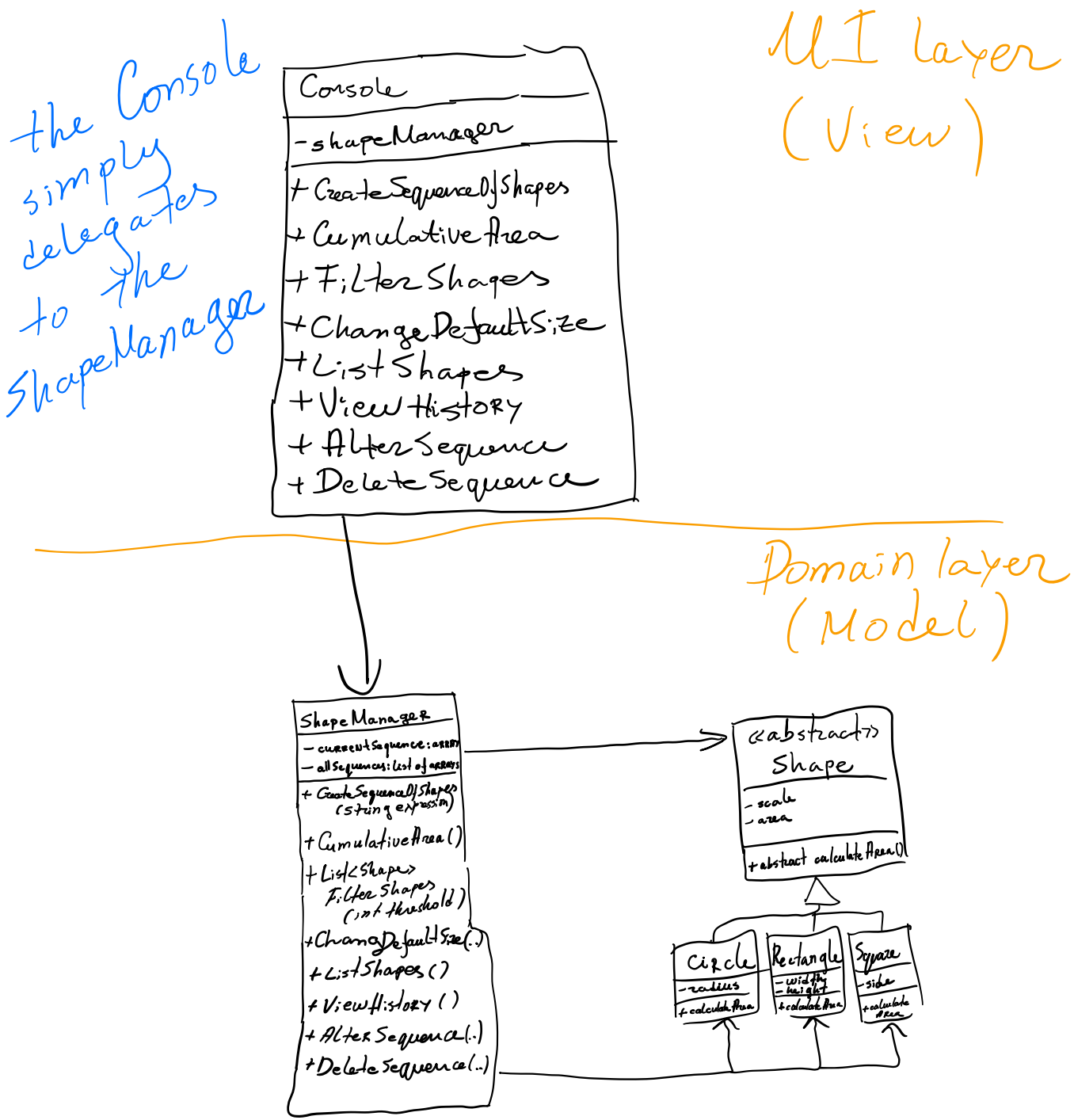
- **Isolate the use of hard coded operators**
  - Extract all uses of operators in a separate class?
  - How to name this one?
  - Data structure?
  - Decouple the different operators from the parsing and building the expression tree



Factory method pattern

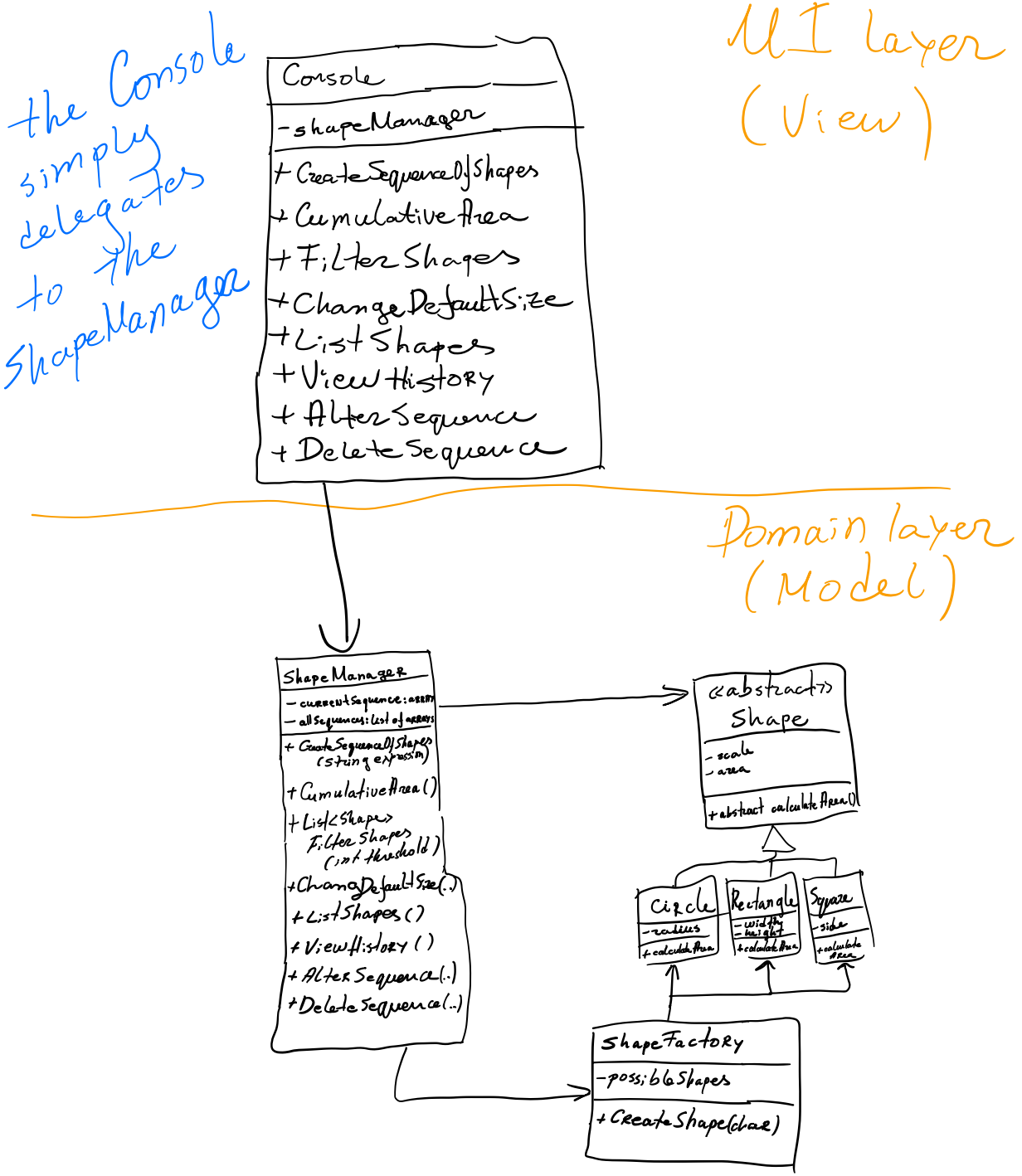
Factory method/Concrete Factory/Simple Factory

- One of the creational design patterns proposed by the ‘Gang of Four’
- Deals with the problem of creating objects with a common supertype or interface without everyone needing to know which exact type will be instantiated.
- Typically used when a class can’t anticipate the type of objects it must create (or should not!)
- Solution:
  - Create a “Pure Fabrication” Factory class
  - The Factory class is then the only one that will know which specific subtype will be instantiated: it will contain a factory method that will do this job
  - Other classes will only deal with the common supertype (class or interface).



Example: NO Factory method

- Remember the shapes example?
- In our initial design we had the ShapeManager creating all shapes in the CreateSequenceOfShapes method
- As a result, the ShapeManager was coupled to all child classes of class Shape



Example: Using the Factory method

- We assign the responsibility of creating all shapes to the ShapeFactory class
- CreateSequenceOfShapes in ShapeManager will only delegate to CreateShapes in the ShapeFactory
- As a result, the ShapeManager only needs to know about the abstract Shape class and not the subclasses

Factory Method in the Expression Tree Code Demo

- We want to make the method responsible for creating the expression tree oblivious of the different operators (to allow easy extension).
- How?
  - Create a Factory class (ex. **OperatorNodeFactory**) with a factory method :

    public OperatorNode **CreateOperatorNode**(char op)
  - Move the logic for the creation of operator nodes in CreateOperatorNode
  - The client (i.e., the ExpressionTree class) will only know about OperatorNode and not the different subclasses
- Don’t forget to adapt your test cases and run them – they should all pass before you continue!