- FloorMastery.UI
    - Folder: Workflows
        - Display Order
        - Add
        - Edit
        - Remove
    - Console.IO
    - Menu.cs
    - Program.cs
- FloorMastery.Tests
    - FileRepoTests
        - Stuff to test for:
            - can access Tax file AND can format file
            - can create new order file
            - can format order file correctly
            - can add/edit/remove orders
            - can access Products file & format
- FloorMastery.Models
    - Orders
        - public class Orders
            - {
                - public int OrderNumber {get ; set; }
                - public string CustomerName {get ; set; }
                - public string State {get; set; }
                - public decimal TaxRate {get; set; }
                - public string ProductType {get; set; }
                - public decimal Area {get; set; }
                - public decimal CostPerSquareFoot {get; set;}
                - public decimal LaborCostPerSquareFoot {get; set;}
                - public decimal.materialCost {get; set;}
                - public decimal.laborCost {get; set;}
                - public decimal tax {get; set;}
                - Public decimal total {get; set;}
            - }
    - Tax
    - Product
    - Interfaces
        - IOrder
        - ITax
        - IProduct
- FloorMastery.Data
    - Crrud method of design
        - Create/add object of whatever type you're handling ...so in add case you're "creating" a new order.
        - Read all--> return anything im asking for

            - Read One by parameter (ID, name, etc.)
            - Update/Edit --> find something already in database and update it.
            - Delete from file --> whatever is told
        - Test Mode - Fake Orders
        - Test - taxes
        - test - products
        - Production Mode
        - production - taxes
        - production - products
- FloorMastery.BLL
    - OrderManager Factory
        - public status OrderManager Create()
            - { switch(mode)
                - case "OrderTest":
                - return new OrderManager (new OrderTestRespository());
                - case "OrderProduction":
                - return new OrderManager(new OrderProductionRepo(path));
                - default:
                    - throw new Exception("Mode value in app config is not valid");
            - }
    - CalculateTotals
        - MaterialCost
            - public class decimal MaterialCost(Orders Area, Orders CostPerSquareFoot){
                - MaterialCost mc = new MaterialCost;
                - mc = area * costpersquarefoot;
                - return mc;
                - }
        - LaborCost
            - public class decimal LaborCost(Orders Area, Orders LaborCostPerSquareFoot){
                - Laborcost lc = new LaborCost;
                - lc = (Area * LaborCostPerSquareFoot);
                - return lc;
                - }
        - Tax
            - public class decimal Tax(path to taxrate){
            - tax = ((MaterialCost + LaborCost)*(taxRate/100))
            - return tax
            - }
        - Total
            - public class decimal Total () {
            - Total toootal = new Total;
            - toootal = (MaterialCost + LaborCost + Tax)
            - return toootal

        - }
- Responses
    - public class Response
    - { public bool Success {get; set; }
        - public string Message { get; set;}
        - }
    - DisplayOrder Response
        - public class DisplayOrder Response : Response
            - { public Order Order {get; set;} }
    - AddOrder Response
        - public class AddOrderResponse : Response
            - { public OrderDate newOrderDate { get; set; }}
            - {public string CustomerName {get; set;}}
            - {public string State {get; set;}}
            - {public enum ProductType {get; set;}}
            - {public decimal Area {get; set;}}
    - EditOrder Response
    - Remove Response
- Rules
    - Folder Add Order Rule
        - //Make sure to instantiate a response class for Add Order
            - **Side note response must represent an "order" so yo can response.CustomerName, response.OrderDate etc etc.
        - // If Order Date != Current or past date
            - the response will be false
            - response will print a message saying its not a valid order
        - //If Customer Name is blank
            - response will be false
            - response will bring a message reminding them not to leave it blank
        - //Take user input and compare against tax file
            - if state does not exist
                - response false
                - response Message "state tax information is not available"
        - //Verify Area >= 100
            - if below
                - response false
                - response Message "Area is too small please make it at least 100sq. ft"
        - Prompt Product Type
            - //Make Enum **maybe in another class** associated with materialCost, LaborCost, ProductType
        - If all is good then...

- response.OrderDate = newOrderDate
- response.CustomerName = newCustomerName
- etc. etc.
- make userInput equal appropriate response attribute

- Instantiate tax, laborcost, materialCost classes with relevant userInput as parameters to produce calculations
- return AddOrder response

- Editing
  - //instantiate a response class
  - //If date = null
    - response.Success = fail
    - response.Message = order is not found
  - If OrderDate = null
    - response.Success = fail
    - response.Message = please enter a correct date
  - 4 methods of change
    - CustomerName
      - if customerName == " "
        - leave old Name
      - if customerName has an input (parameter filled)
        - change name
    - State
      - if state == " "
        - leave old state
      - if state has input
        - change state
    - ProductType
      - if blank
        - leave old product type
      - switch(Enum)
        - make new Enum the current Enum
    - Area
      - if area blank
        - leave old area
      - if area has new number that is positive and above 100
        - change area
  - Initiate new calculations with new values
    - response.bla = response.newbla
  - return edited order
- Removing
  - initiate response
  - Check for date

        - if date doesn't exist
            - response.Success = fail
            - response.Message = Please enter new Date
- Check for Order Number
    - if order number doesn't exist
        - response.Success = fail
        - response.Message = Please enter a correct order number
- Check for user answer
    - if Yes
        - Loop through orders to match date and order#
            - Find order in List
            - delete order
    - if No
        - return to menu // do nothing