### Step 0: User Stories

* Display Orders
* Add an Order
* Edit an Order
* Remove an Order
* Export all data

### Step 1: required classes

* UserIO (helper used by the View class to interact with the console)
* View class (used by Controller to handle user interaction)
* OrderController (this class orchestrates the program)
* App (this class has a main method that instantiates AddressBookController and calls the execute method)
* Order (DTO)
* Tax (DTO)
* Product (DTO)
* OrderBookDao (DAO)
  + OrderBookDaoFileImpl
  + OrderBookPersistenceException
* OrderBookAuditDao (DAO)
  + OrderBookAuditDaoFileImpl
* OrderBookTaxDao (DAO)
  + OrderBookTaxDaoFileImpl
* OrderBookProductDao (DAO)
  + OrderBookProductDaoFileImpl
* OrderServiceLayer (Service)
  + OrderServiceLayerImpl
  + OrderBookTaxException
  + OrderBookProductException
  + OrderBookInvalidDateException
  + OrderBookInvalidAreaException

### Step 2: Define properties/methods for each class

* UserIO
  + Add BigDecimal method
  + Add LocalDate method
* Order DTO
  + Private members
    - private LocalDate orderDate
    - private int orderNumber
    - private String customerName
    - private String state
    - private BigDecimal taxRate
    - private String productType
    - private BigDecimal area
    - private BigDecimal costPerSquareFoot
    - private BigDecimal laborCostPerSquareFoot
    - private BigDecimal materialCost
    - private BigDecimal laborCost
    - private BigDecimal tax
    - private BigDecimal total
  + Private methods
    - private void calculateMaterialCost()
    - private void calculateLaborCost()
    - private void calculateTax()
    - private void calculateTotal()
  + Public methods
    - public Order(LocalDate orderDate, String customerName, String state, String productType, BigDecimal area)
    - public void CalculateFields()
    - getters/setters
* Tax DTO
  + Private fields:
    - private String stateAbbreviation
    - private String stateName
    - private BigDecimal taxRate
  + Public methods:
    - public Tax(String stateAbbreviation, String stateName, BigDecimal taxRate)
* Product DTO
  + Private fields:
    - private String productType
    - private BigDecimal costPerSquareFoot
    - private BigDecimal laborCostPerSquareFoot
  + Public methods
    - public Product(String productType, BigDecimal costPerSquareFoot, BigDecimal laborCostPerSquareFoot)
* OrderBookDao
  + Order addOrder(Order order)
  + Order editOrder(Order editedOrder)
  + Order removeOrder(LocalDate orderDate, int orderNumber)
  + Order getOrder(LocalDate orderDate, int orderNumber)
  + List<Order> getAllOrders(LocalDate orderDate)
* OrderBookDaoFileImpl
  + Public members:
    - Constants for file persistence
  + Private members:
    - private Map<Integer, Order> orders
  + Private methods:
    - private Order unmarshallOrder(String orderAsText)
    - private String marshallOrder(Order anOrder)
    - private void loadBook(LocalDate orderDate)
    - private void writeBook(LocalDate orderDate)
  + Public methods:
    - Overridden from dao
* OrderBookTaxDao:
  + Tax getTax(String state)
* OrderBookTaxDaoFileImpl
  + Public members:
    - Constants for file persistence
  + Private members:
    - private Map<String, Tax> taxes
  + Private methods:
    - private Tax unmarshallTax(String taxAsText)
    - private void loadTaxes()
* OrderBookProductDao:
  + Product getProduct(String productType)
* OrderBookProductDaoFileImpl:
  + Public members:
    - Constants for file persistence
  + Private members:
    - private Map<String, Product> products
  + Private methods:
    - private Product unmarshallProduct(String productAsText)
    - private void loadProducts()
* OrderBookServiceLayer:
  + void createOrder(Order order)
  + List<Order> getAllOrders(LocalDate orderDate)
  + Order getOrder(LocalDate orderDate, int orderNumber)
  + Order removeOrder(LocalDate orderDate, int orderNumber)
  + Order editOrder(Order order)
* OrderBookServiceLayerImpl:
  + Business Logic:
    - Order date must be in the future
    - Fields must not be blank
    - State must exist in tax file
    - Product must exist in product file
    - Area must be positive decimal, min 100
  + Private members:
    - private OrderBookDao dao;
    - private OrderBookTaxDao taxDao;
    - private OrderBookProductDao productDao;
    - private OrderBookAuditDao auditDao;
  + Public methods:
    - Constructor w DI
    - Overriden methods
  + Private methods:
    - private void validateOrder(Order anOrder)
    - private void validateTax(String state)
    - private void validateProduct(String productType)
    - private void validateDate(LocalDate date)
    - private void validateArea(BigDecimal area)
* OrderBookView:
  + Private members:
    - final UserIO io
  + Public methods:
    - Constructor with DI

// for getting user input

* + - public int printMenuAndGetSelection()
    - public LocalDate getDateChoice()
    - public int getOrderNumber()
    - public Order getNewOrderInfo()
    - public Order getEditOrderInfo()
    - public int getRemoveConfirmation()

// for displaying info to user

* + - public void displayOrders(List<Order> orders)
    - public void displayOrder(Order order)

// for displaying success/failures/other alerts

* + - public void displayPlaceOrderBanner()
    - public void displayOrderSuccessBanner()
    - public void displayDisplayOrdersBanner(LocalDate date)
    - public void displayEditOrderBanner()
    - public void displayEditOrderSuccessBanner()
    - public void displayRemoveOrderSuccessBanner()
    - public void displayErrorMessage(String errorMsg)
    - public void displayExitBanner()
    - public void displayUnknownCommandBanner()
* OrderBookController:
  + Private members:
    - private OrderBookView view
    - private OrderBookServiceLayer service
  + Public methods:
    - Constructor with DI
    - public void run()
  + Private methods:
    - private int getMenuSelection()
    - private void addOrder()
    - private void editOrder()
    - private void removeOrder()
    - private void displayOrders()
    - private void unknownCommand()
    - private void exitMessage()