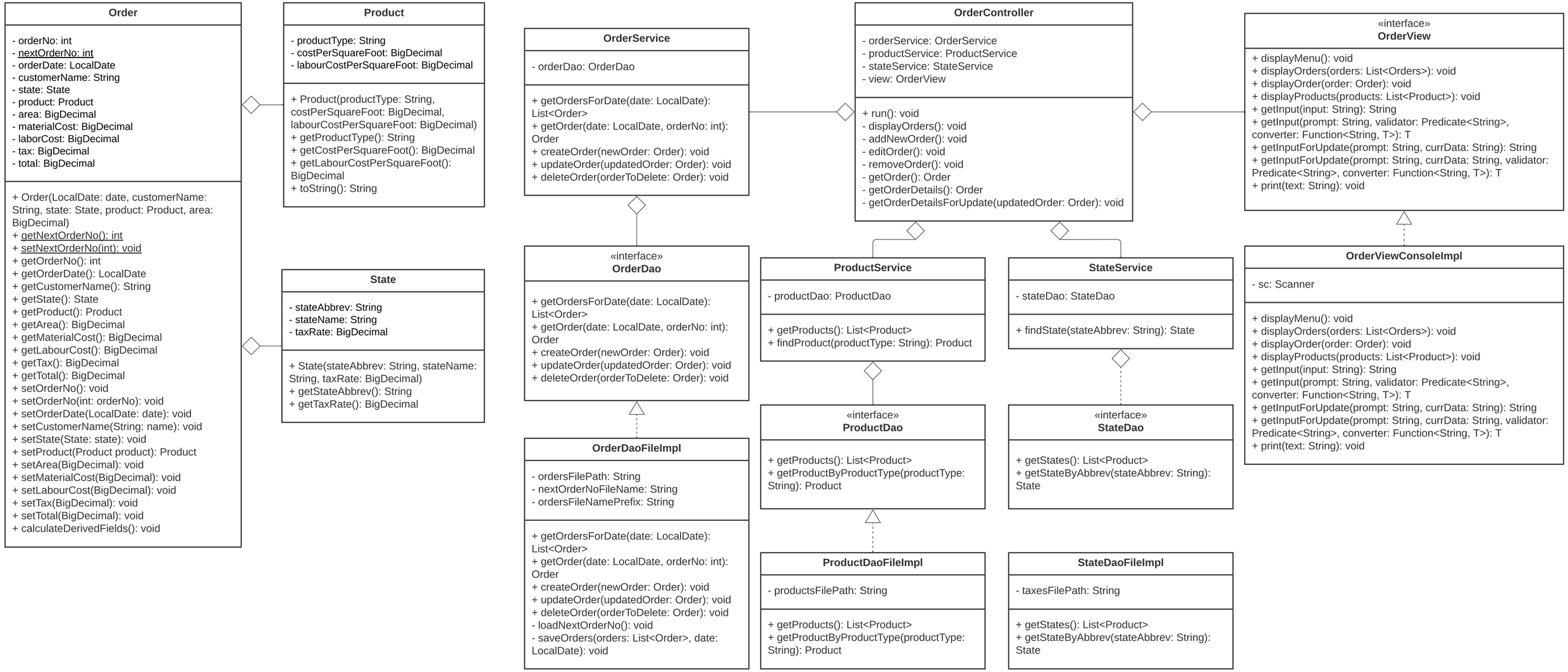
Order
- orderNo: int
- nextOrderNo: int
- orderDate: LocalDate
- customerName: String
- state: State
- product: Product
- area: BigDecimal
- materialCost: BigDecimal
- laborCost: BigDecimal
- tax: BigDecimal
- total: BigDecimal
+ Order(LocalDate: date, customerName: String, state: State, product: Product, area: BigDecimal)
+ getNextOrderNo(): int
+ setNextOrderNo(int): void
+ getOrderNo(): int
+ getOrderDate(): LocalDate
+ getCustomerName(): String
+ getState(): State
+ getProduct(): Product
+ getArea(): BigDecimal
+ getMaterialCost(): BigDecimal
+ getLabourCost(): BigDecimal
+ getTax(): BigDecimal
+ getTotal(): BigDecimal
+ setOrderNo(): void
+ setOrderNo(int: orderNo): void
+ setOrderDate(LocalDate: date): void
+ setCustomerName(String: name): void
+ setState(State: state): void
+ setProduct(Product product): Product
+ setArea(BigDecimal): void
+ setMaterialCost(BigDecimal): void
+ setLabourCost(BigDecimal): void
+ setTax(BigDecimal): void
+ setTotal(BigDecimal): void
+ calculateDerivedFields(): void
Product
- productType: String
- costPerSquareFoot: BigDecimal
- labourCostPerSquareFoot: BigDecimal
+ Product(productType: String, costPerSquareFoot: BigDecimal, labourCostPerSquareFoot: BigDecimal)
+ getProductType(): String
+ getCostPerSquareFoot(): BigDecimal
+ getLabourCostPerSquareFoot(): BigDecimal
+ toString(): String
State
- stateAbbrev: String
- stateName: String
- taxRate: BigDecimal
+ State(stateAbbrev: String, stateName: String, taxRate: BigDecimal)
+ getStateAbbrev(): String
+ getTaxRate(): BigDecimal
OrderService
- orderDao: OrderDao
+ getOrdersForDate(date: LocalDate): List<Order>
+ getOrder(date: LocalDate, orderNo: int): Order
+ createOrder(newOrder: Order): void
+ updateOrder(updatedOrder: Order): void
+ deleteOrder(orderToDelete: Order): void
«interface»
OrderDao
+ getOrdersForDate(date: LocalDate): List<Order>
+ getOrder(date: LocalDate, orderNo: int): Order
+ createOrder(newOrder: Order): void
+ updateOrder(updatedOrder: Order): void
+ deleteOrder(orderToDelete: Order): void
OrderDaoFileImpl
- ordersFilePath: String
- nextOrderNoFileName: String
- ordersFileNamePrefix: String
+ getOrdersForDate(date: LocalDate): List<Order>
+ getOrder(date: LocalDate, orderNo: int): Order
+ createOrder(newOrder: Order): void
+ updateOrder(updatedOrder: Order): void
+ deleteOrder(orderToDelete: Order): void
- loadNextOrderNo(): void
- saveOrders(orders: List<Order>, date: LocalDate): void
OrderController
- orderService: OrderService
- productService: ProductService
- stateService: StateService
- view: OrderView
+ run(): void
- displayOrders(): void
- addNewOrder(): void
- editOrder(): void
- removeOrder(): void
- getOrder(): Order
- getOrderDetails(): Order
- getOrderDetailsForUpdate(updatedOrder: Order): void
ProductService
- productDao: ProductDao
+ getProducts(): List<Product>
+ findProduct(productType: String): Product
«interface»
ProductDao
+ getProducts(): List<Product>
+ getProductByProductType(productType: String): Product
ProductDaoFileImpl
- productsFilePath: String
+ getProducts(): List<Product>
+ getProductByProductType(productType: String): Product
StateService
- stateDao: StateDao
+ findState(stateAbbrev: String): State
«interface»
StateDao
+ getStates(): List<Product>
+ getStateByAbbrev(stateAbbrev: String): State
StateDaoFileImpl
- taxesFilePath: String
+ getStates(): List<Product>
+ getStateByAbbrev(stateAbbrev: String): State
«interface»
OrderView
+ displayMenu(): void
+ displayOrders(orders: List<Orders>): void
+ displayOrder(order: Order): void
+ displayProducts(products: List<Product>): void
+ getInput(input: String): String
+ getInput(prompt: String, validator: Predicate<String>, converter: Function<String, T>): T
+ getInputForUpdate(prompt: String, currData: String): String
+ getInputForUpdate(prompt: String, currData: String, validator: Predicate<String>, converter: Function<String, T>): T
+ print(text: String): void
OrderViewConsoleImpl
- sc: Scanner
+ displayMenu(): void
+ displayOrders(orders: List<Orders>): void
+ displayOrder(order: Order): void
+ displayProducts(products: List<Product>): void
+ getInput(input: String): String
+ getInput(prompt: String, validator: Predicate<String>, converter: Function<String, T>): T
+ getInputForUpdate(prompt: String, currData: String): String
+ getInputForUpdate(prompt: String, currData: String, validator: Predicate<String>, converter: Function<String, T>): T
+ print(text: String): void