Use cases

Use case for adding components:

Actions performed by the actor	Responses from the system
1. User issues a request to add new components.	
	2. The system asks for a component name.
3. User enters component name.	
	4. If the component can be added, the system creates a new component with the given name and assigns it a generated ID. The system remembers the new component and displays the component information (including name and ID). The system then asks if the user would like to add more components.
5. User answers in the affirmative or in the negative.	
	6. If the answer is in the affirmative, the system goes to step 2., otherwise, it exits the use case.

Use case for adding a new component supplier:

Actions performed by the actor	Responses from the system
1. User issues a request to add a new component supplier.	
	2. The system asks for the component ID.
3. User enters component ID.	
	4. If the component ID is valid, the system asks for the supplier ID. Otherwise, it displays an appropriate message and exits the use case.
5. User enters supplier ID.	
	6. If the supplier ID is valid and the supplier is not already supplying this component, the system remembers that this supplier provides this component and displays a message confirming the relationship. Otherwise, it displays an appropriate error message. The system then exits the use case.

Use case for assigning components to a product:

Actions performed by the actor	Responses from the system
1. User issues a request to assign components to a product.	
	2. The system asks for the component ID.
3. User enters component ID.	
	4. If the component ID is valid, the system asks for the quantity to be assigned. Otherwise, it displays an appropriate error message and exits the use case.
5. User enters quantity to be assigned.	
	6. If the quantity > 0, and ≤ component's available stock, the system subtracts the given quantity from the component's stock and displays the updated value. Otherwise, it displays an appropriate error message. The system then exits the use case.

Use case for placing an order:

Actions performed by the actor	Responses from the system
1. User issues a request to order components.	
	2. The system asks for the component ID.
3. User enters component ID.	
	4. If the component ID is valid, the system asks for the supplier ID. Otherwise, it displays an appropriate error message and exits the use case.
5. User enters supplier ID.	
	6. If the supplier ID is valid, the system asks for the order quantity. Otherwise, it displays an appropriate error message and exits the use case.
7. User enters order quantity.	
	8. If the quantity > 0, and the supplier supplies this component, the system places a new order with a unique ID and displays its details. Otherwise, it displays an appropriate error message. The system then exits the use case.

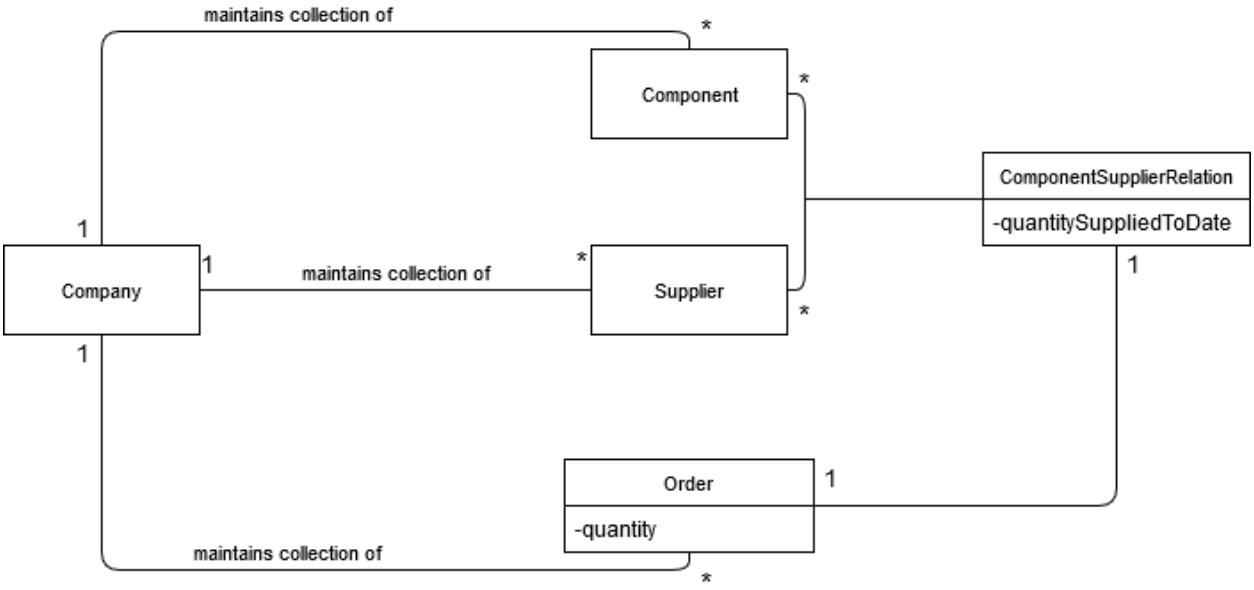
Use case for fulfilling an order:

Actions performed by the actor	Responses from the system
1. User issues a request to mark an order as fulfilled.	
	2. The system asks for the ID of the outstanding order.
3. User enters the order ID.	
	4. If the order ID is valid, the system updates the stock of the component and the total quantity of this component supplied by this supplier. Otherwise, it displays an appropriate error message. The system then exits the use case.

Use case for displaying all outstanding orders:

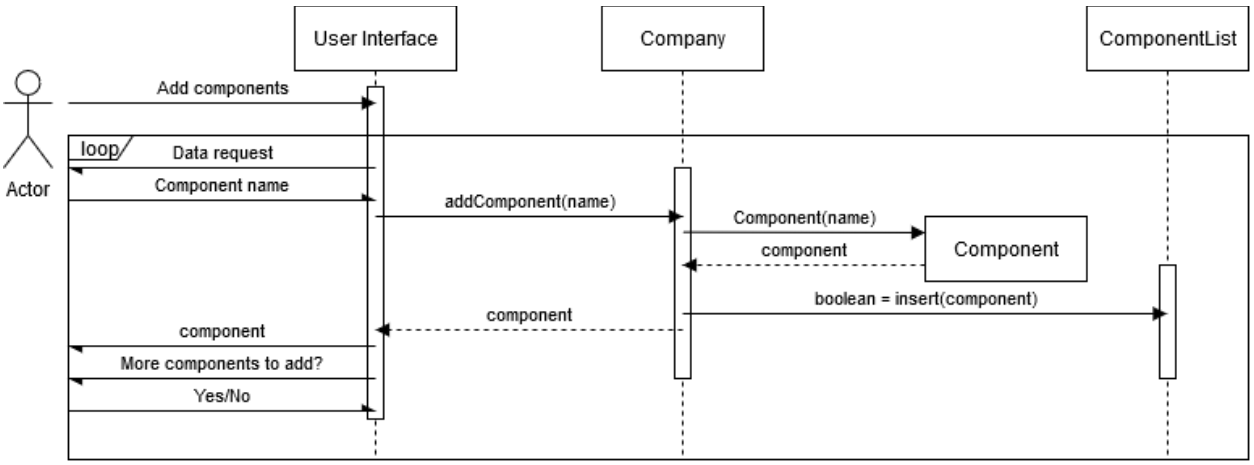
Actions performed by the actor	Responses from the system
1. User issues a request to display all outstanding orders.	
	2. The system displays information about all the outstanding orders. For each outstanding order, the names of the component and supplier are displayed, along with the order ID and the quantity ordered.

Conceptual Class Diagram

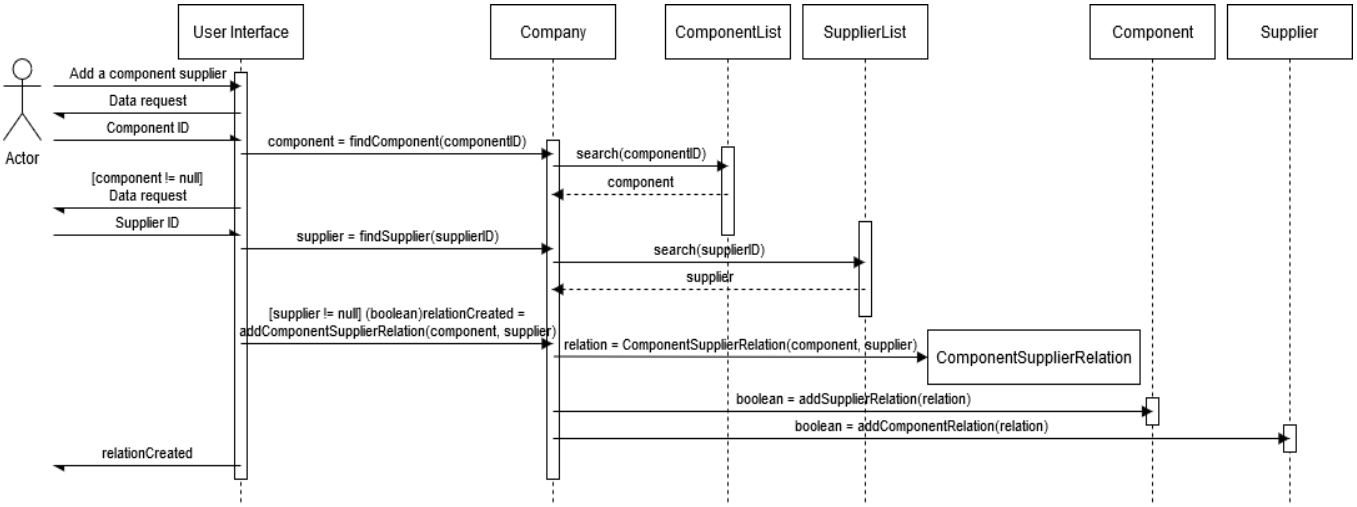


Sequence diagrams

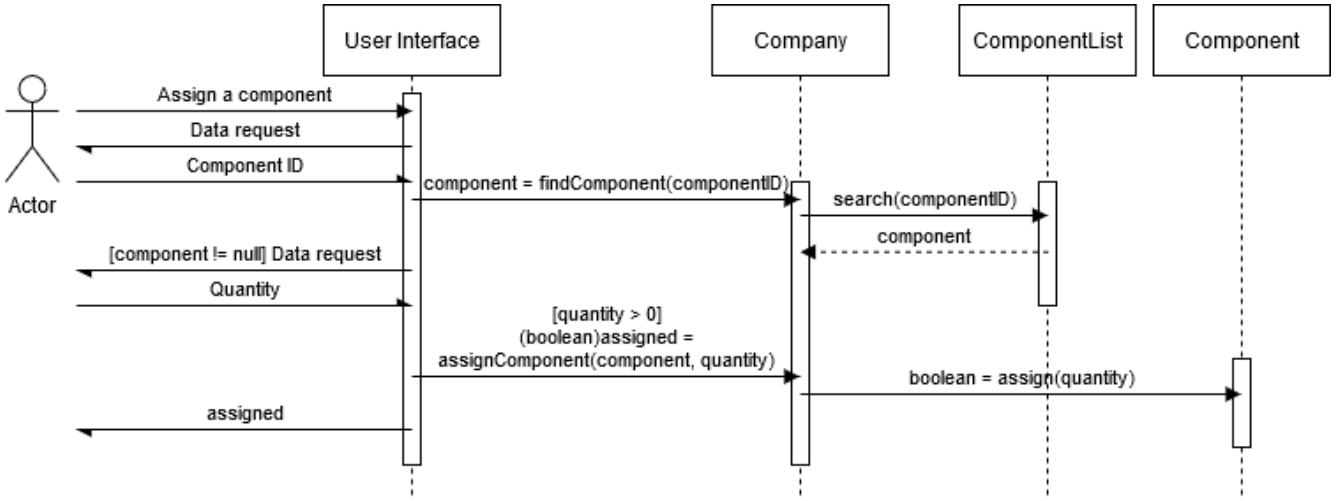
Sequence diagram for adding components:



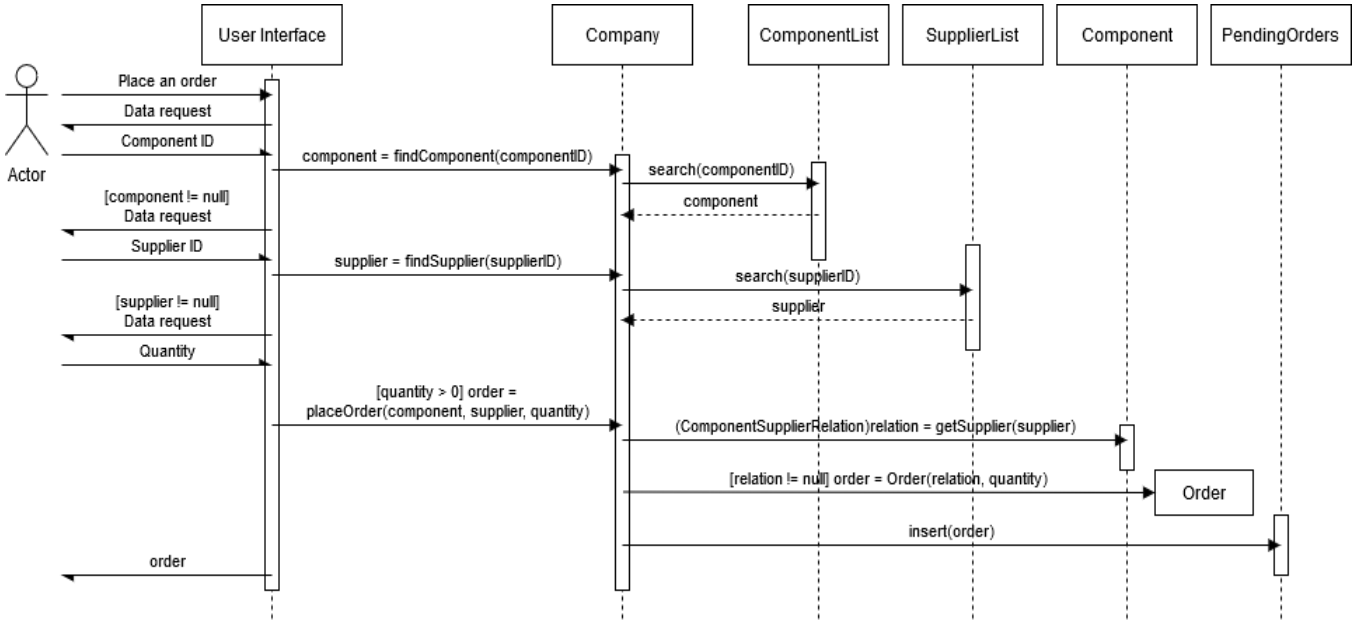
Sequence diagram for adding a component supplier:



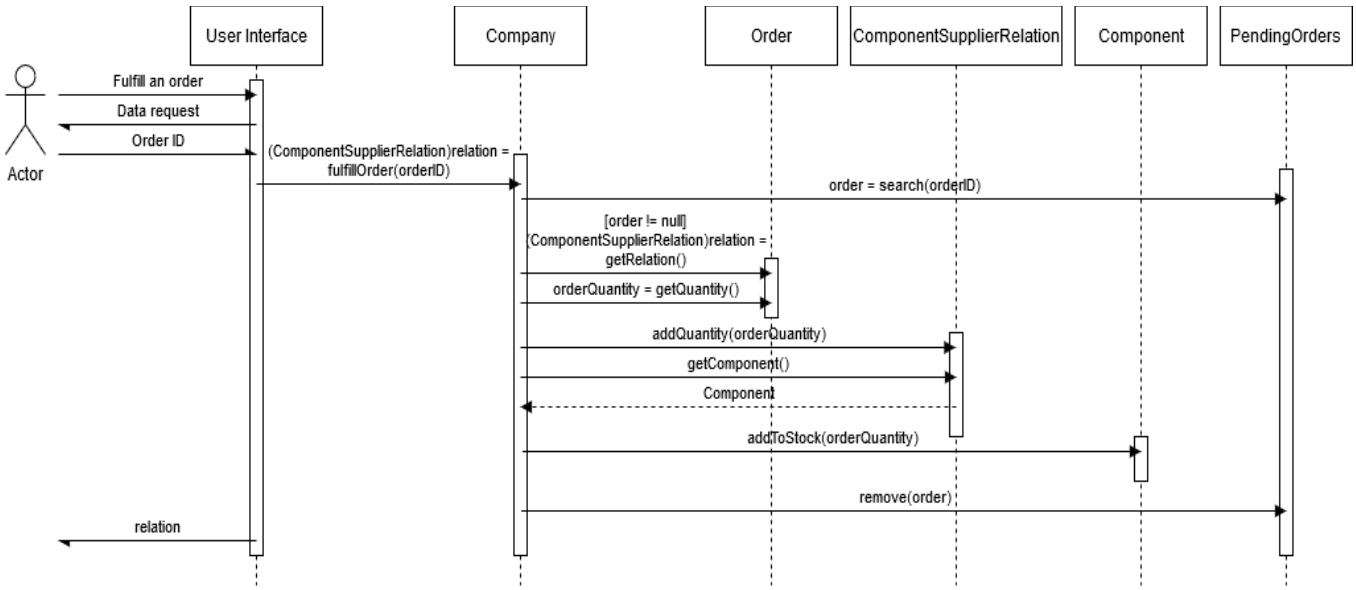
Sequence diagram for assigning components to a product:



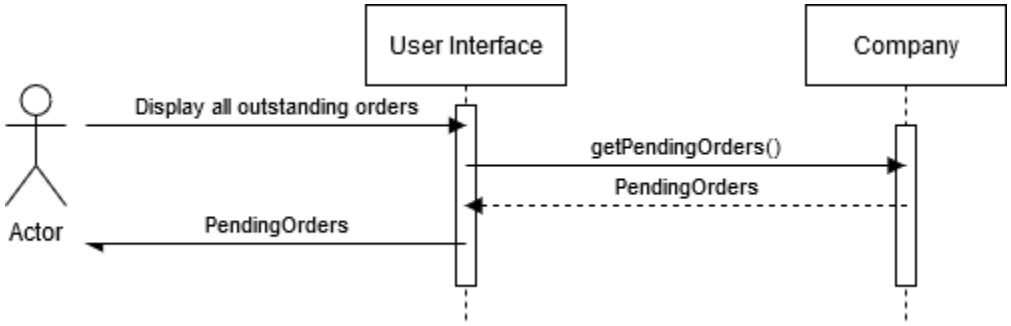
Sequence diagram for placing an order:



Sequence diagram for fulfilling an order:

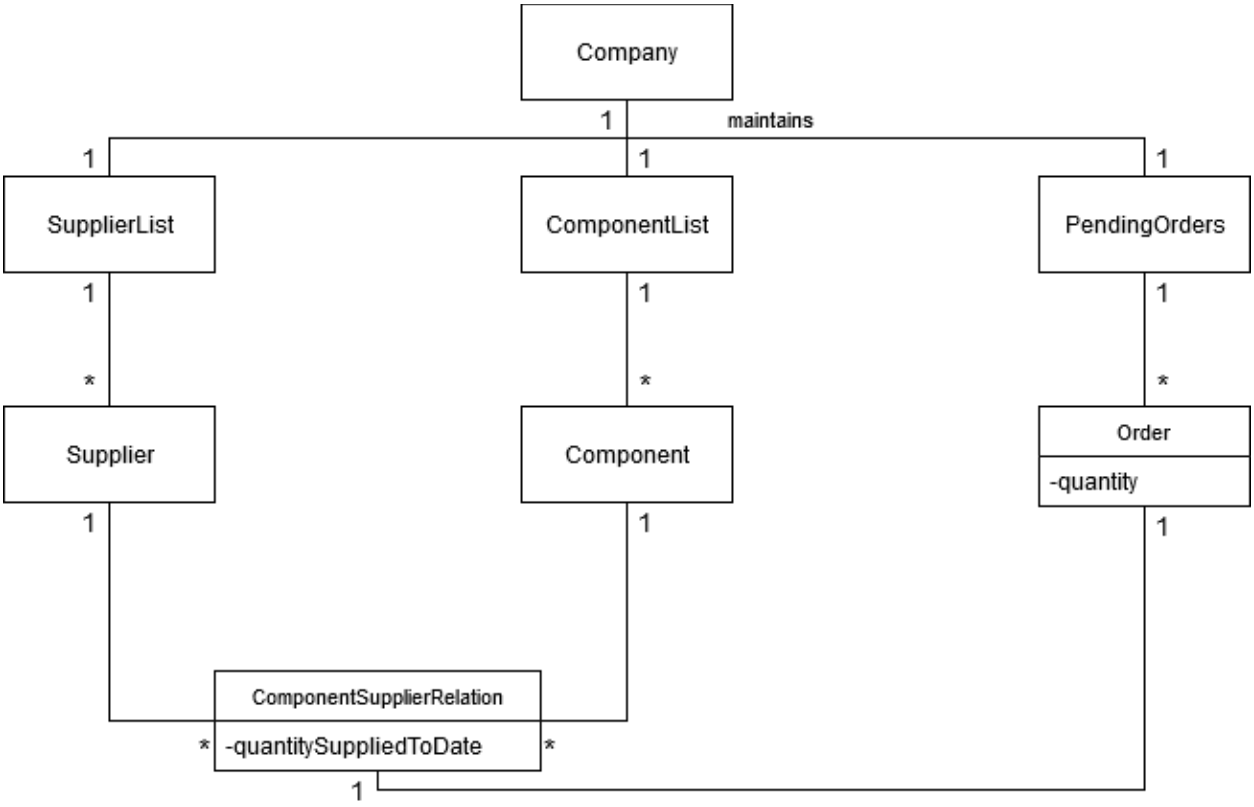


Sequence diagram for displaying all outstanding orders:

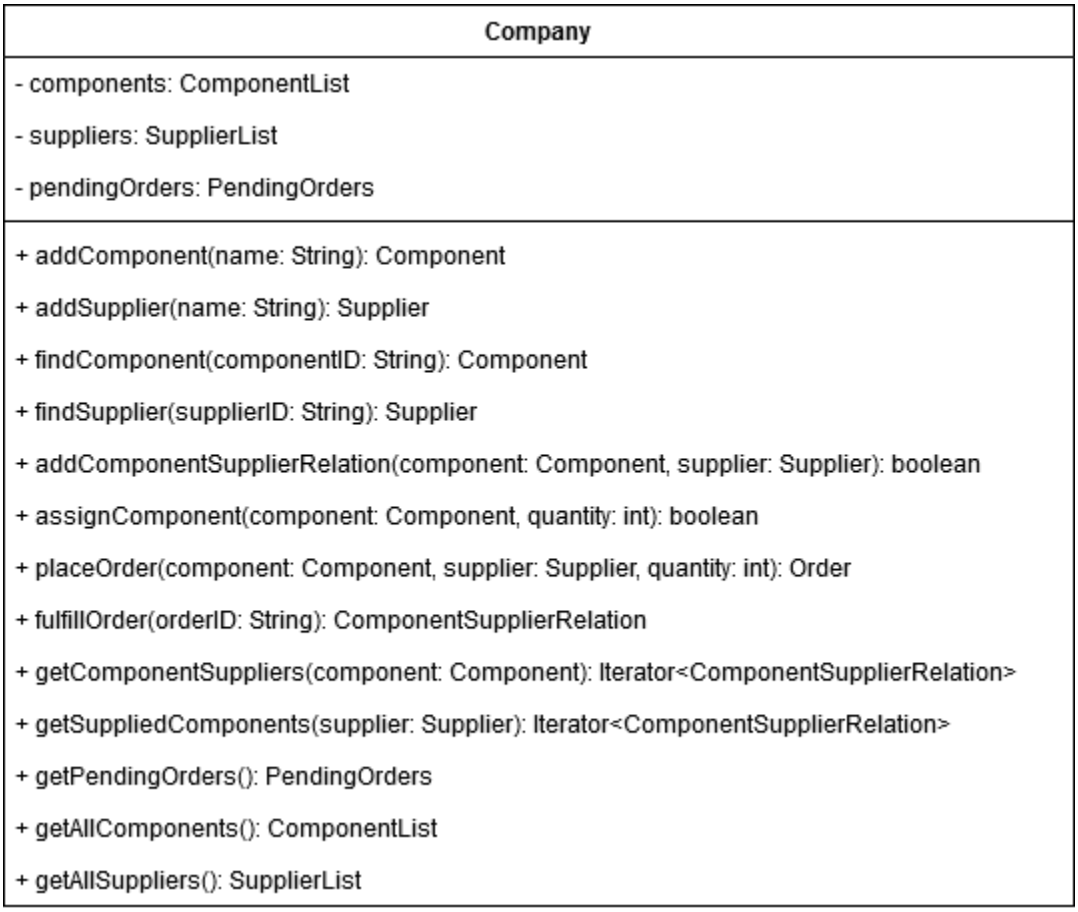


Physical Class Diagram

Overall class diagram:



Class diagram for Company:



Class diagram for Component:

Component
- name: String - id: String - stock: int - supplierRelations: HashSet<ComponentSupplierRelation>
+ Component(name: String): Component + getName(): String + getId(): String + getStock(): int + addToStock(quantity: int): void + getSupplierRelations(): HashSet<ComponentSupplierRelation> + assign(quantity: int): boolean + getSupplier(supplier: Suppiler): ComponentSupplierRelation + getAllSuppliers(): Iterator<ComponentSupplierRelation> + toString(): String

Class diagram for Supplier:

Supplier
- name: String - id: String - componentRelations: HashSet<ComponentSupplierRelation>
+ Supplier(name: String): Supplier + getId(): String + getName(): String + getComponentRelations(): HashSet<ComponentSupplierRelation> + addComponentRelation(relation: ComponentSupplierRelation): boolean + getAllComponents(): Iterator<ComponentSupplierRelation> + toString(): String

Class diagram for Order:

Order
- relation: ComponentSupplierRelation - quantity: int - id: String
+ Order(relation: ComponentSupplierRelation, quantity: int): Order + getId(): String + getQuantity(): int + getRelation(): ComponentSupplierRelation + toString(): String

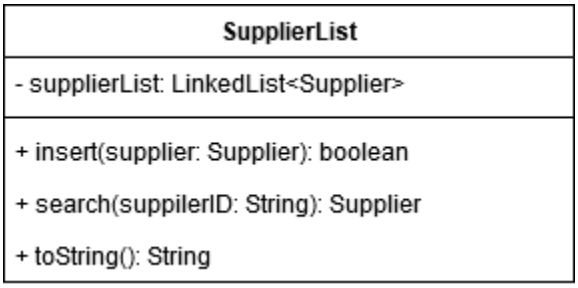
Class diagram for ComponentSupplierRelation:

ComponentSupplierRelation
- component: Component - supplier: Supplier - quantitySuppliedToDate: int
+ ComponentSupplierRelation(component: Component, supplier: Supplier): ComponentSupplierRelation + getQuantitySuppliedToDate(): int + getComponent(): Component + getSupplier(): Supplier + addQuantity(quantity: int): void + equals(object: Object): boolean + hashCode(): int + toString(): String

Class diagram for ComponentList:

ComponentList
- componentList: LinkedList<Component>
+ insert(component: Component): boolean + search(componentID: String): Component + toString(): String

Class diagram for SupplierList:



Class diagram for PendingOrders: