G6046 Software Engineering

Seminar session: Class design

The task

You have been given an outline specification for a computer system to help manage stock for a new community upscaling facility.

"Margo and Barbara are starting a new community facility called 'Bargo' for upscaling and recycling household items. The idea is that members of the public can bring old sofas, chairs and electrical items that they no longer want and Bargo can then either restore the items and sell them for a profit or arrange for them to be disposed of in a safe and proper manner.

When someone donates an item to Bargo, they do not need to provide any identification. One of the Bargo team needs to record what the item is in the computer system. For sofas, they need to know how many people the sofa could seat, the condition rated from 1 (very poor) to 5 (excellent), the colour and the type of fabric. For chairs, we need the same data as sofas, except that all chairs only seat one person. For electrical items, we need a simple one line description of what the item is, the condition using the same rating system, and a record to show whether the item has been checked by an electrician to determine whether it is safe to use, as all unsafe electrical items need to be disposed of.

All items that are rated as 3 or better will be put up for sale. The computer system will need to identify which items in stock can be sold. Any item with a poorer rating, or deemed unsafe, must be disposed of. Each week, the computer system needs to produce a list of items to be disposed of. The sale items can be viewed by customers using a simple search term. In order to buy items at Bargo, customers need to open an account. To do this they need to provide a name, address and an email address. Once they have opened an account, the computer system will keep a record of all purchases they make. A customer can also reserve an item for payment within 14 days. After 14 days, the item goes back to being on sale. For convenience, Bargo staff need to be able to recover data about customer reservations by using the customer name."

You are tasked with coming up with some UML diagrams as part of the design phase for an OO code solution:

1. Draw a UML use case diagram to represent this scenario. You need to identify the relevant Actors and the key operations that they can perform.
2. Sketch out Java class definitions appropriate for coding this system. Describe the classes that will be needed and list the fields (public will suffice) and the methods that each class will need to implement. You do not need to provide any substantive source code for the classes, just their external interfaces and some outline of what they will require. You should show how your classes will interact together by means of a class diagram.
3. Once you have a class diagram, try to produce a sequence diagram (to validate your class diagram) to represent the act of a customer browsing the available items and then purchasing one of them.

Several classes will use collections of objects of other classes. Which of the available Java Collection types would be most appropriate for each of the collections? Give reasons for your choices.

Things to watch out for

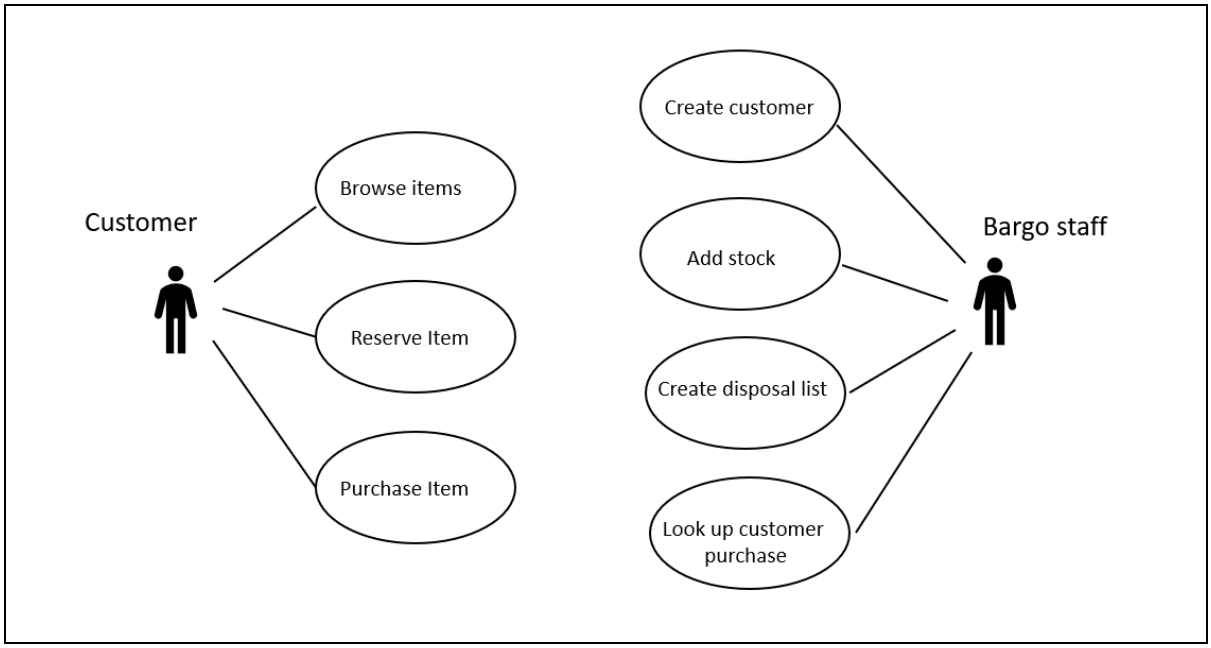
- Consider how polymorphism can help (super/sub classes). This is a key feature of OO languages.
- Don't confuse a class designed to represent a single thing with a collection class. Java allows for collections of any type.
- Aspire for high cohesion and low coupling, and consider issues like responsibility driven design.

Suggested answers for use case diagrams

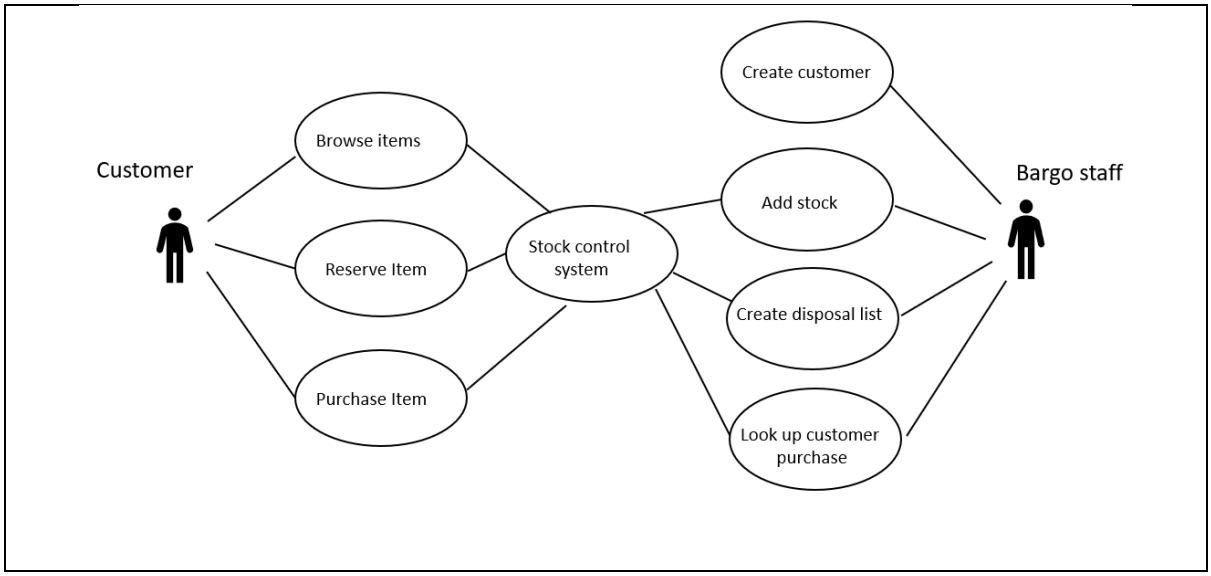
There are two actors:

- The `Bargo` staff
- The `Customer`

Each needs to be able to do different things with this application. The use case analysis provides a very high-level view of the functional requirements that emerge from the user requirements.



This is not the only possible answer. As an alternative, we could consider the existence of some kind of centralised stock management system:



This analysis might start us thinking about whether a stock control class/entity/sub-system might be useful. But that's a point for further discussion.

Suggested answers for class diagram

A possible solution might look like this (constructors and setter/getter methods are omitted for clarity):

Class	Attributes	Methods
Bargo	ArrayList<Item> stock ArrayList<Customer> allCustomers HashMap<String,Customer> purchaseRecords HashMap<String, Customer> reservedItems	ArrayList<Item> showReserved(String) createCustomer(String name, String address, String email) addStock(Item) makeDisposalList() disposePoorStock() getItemDetails(Item) makeSale(Item, Customer) makeReservation(Item, Customer)
Item	int condition boolean reserved	getItemDetails()

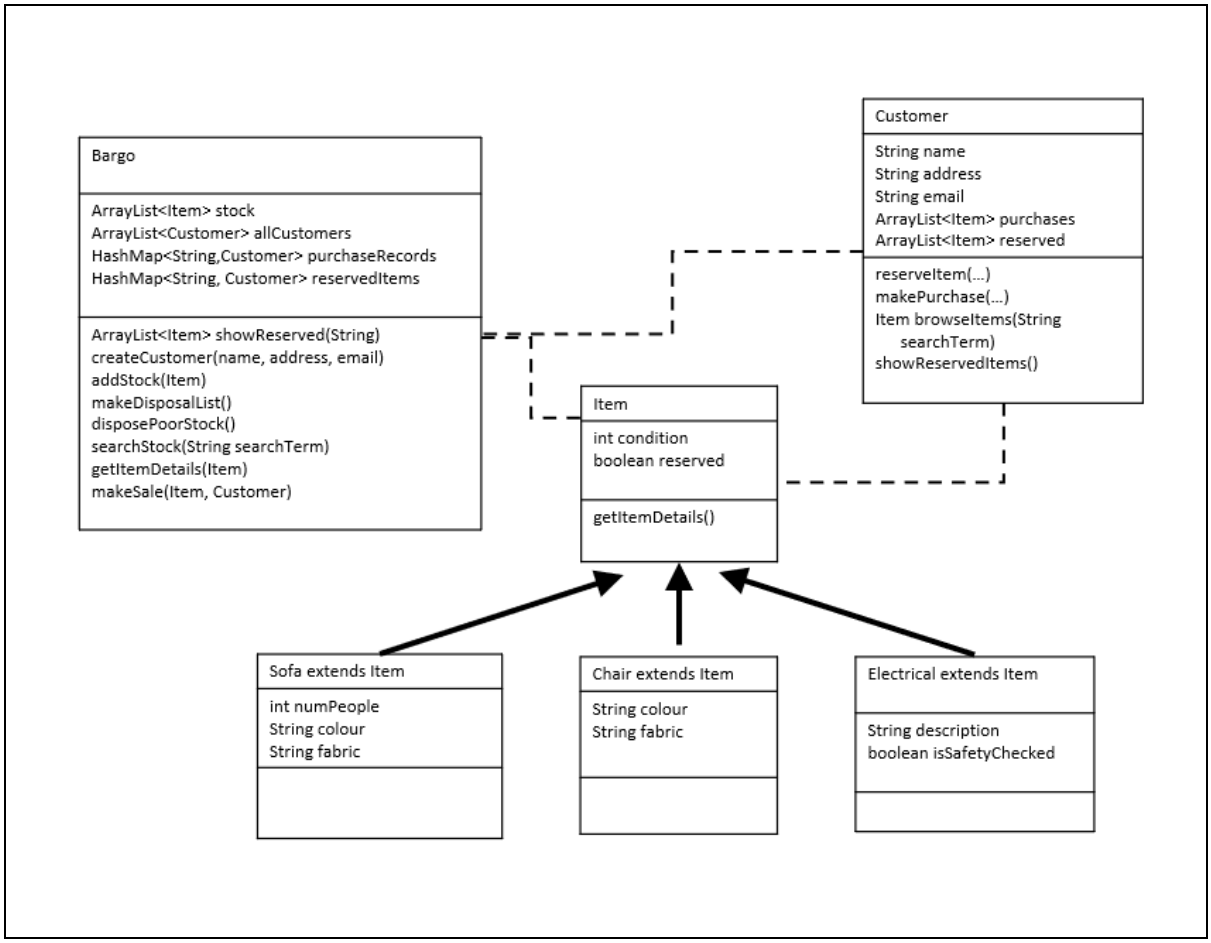
`Sofa extends Item`	`int numPeople`
	`String colour`
	`String fabric`	

`Chair extends Item`	`String colour` `String fabric`	
`Electrical extends Item`	`String description` `boolean isSafetyChecked`	

`Customer`	`String name`
	`String address`
	`String email`
	`ArrayList<Item> purchases`
	`ArrayList<Item> reserved`
		`reserveItem(…)`
		`browseItems(String searchTerm)`
		`makePurchase(…)`
		`showReservedItems()`

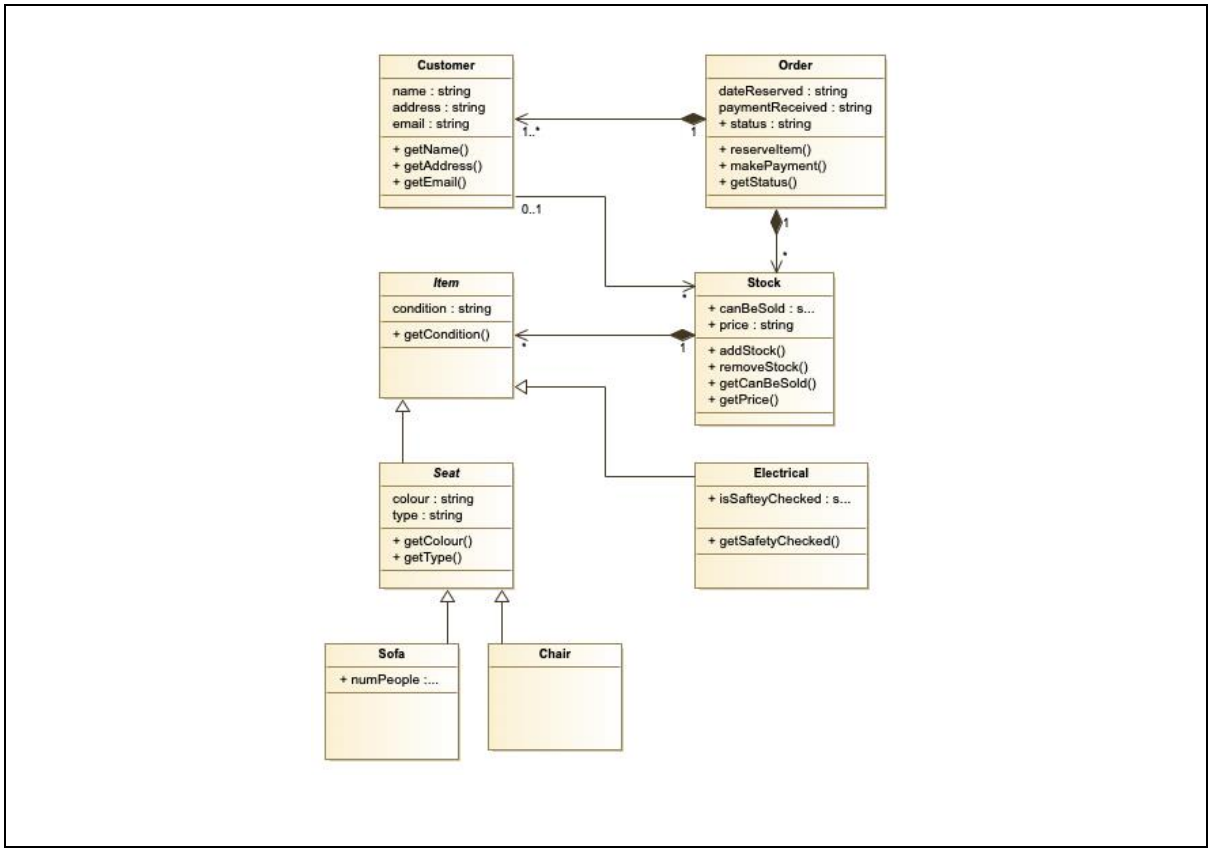
Draw a class diagram to see the extent of coupling. Note:

- The use of the `Item` superclass.
- Who was responsible for adding stock.
- Who was responsible for making purchases and reserving items.
- The use of the two collection types `ArrayList` and `HashMap`.



What do you think of this solution? On the surface it looks like it is OK, but is it as cohesive as it could be?

Here is an alternative part solution (by another member of staff) showing a different level of detail. How do you think it compares?



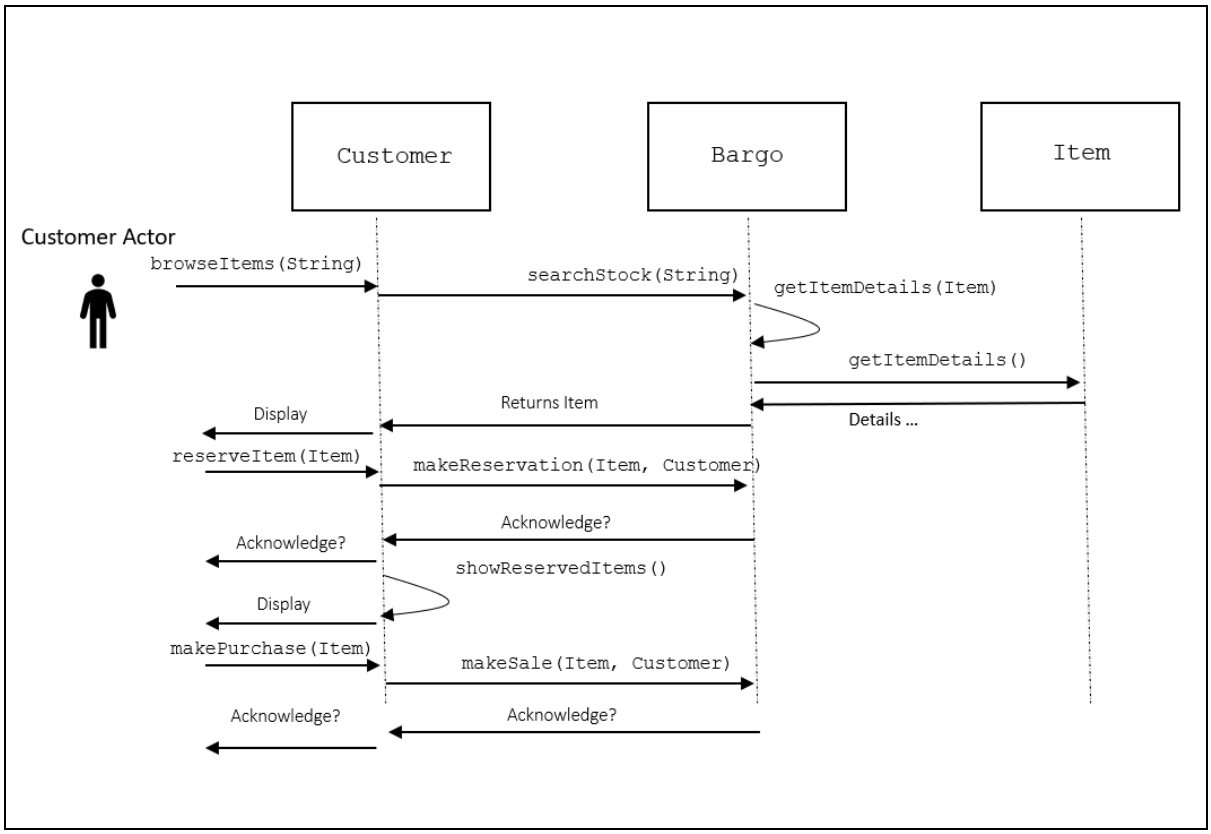
Suggested answer for sequence diagram

We want to validate the class diagram (I am using the first example here) to determine whether it is "fit for purpose". Fit for purpose means that it allows us sufficient confidence that we could deliver the use cases. That could be a single use case, or a combination of use cases – it does not really matter. We just need to consider whether we have done "enough" design at this stage.

We want to represent the act of a customer browsing the available items and then purchasing one of them. This involves:

- Browsing for an item using a simple search term.
- Selecting an item for purchase (effectively reserving an item)
- (Optionally) checking the items we selected/reserved.
- Making the actual purchase.

This spans several use cases from our original analysis.



Some points to note:

- For a method call, the arrow points INTO the class/entity that provides it. So here, `browseItem()` must be a method provided by the `Customer` class.
- For an intra class method call, the arrow loops around. This represents a method in a class that does not interact with other classes.

We can make this as simple or as detailed as we think necessary to get the job done.

Dr Kingsley Sage
Khs20@sussex.ac.uk