Advanced Software Engineering F21AS

Stage 1 - Development Plan

Date: 31/1/2019
Group 2

Authors: Ioannis Athanasiou ia50@hw.ac.uk; Nanduri Ram rnn1@hw.ac.uk; Schatton Bartosz bps2@hw.ac.uk; Thomas Triffterer tt63@hw.ac.uk

The application is being designed for a coffee shop which allows customers to make and submit orders using an interface in the shop. The manager can view a summary report when the interface is closed.

Work Plan

For the actual development process, it was decided that the following tasks would need to be undertaken, considering the below mentioned structures and assumptions.

		WEEKS							
	Duration	**W1 (7/Jan)**	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8 (30/Feb)**
Design & Diagrams	3w	■	■	■					
Implementation	2w				▨	▨			
Unit Testing	1w						▨		
Quality Assurance	1w							▨	
Documentation/Report	1w								▨

The **planned iterative development** methodology will be utilized for the implementation of this stage. Furthermore, the team will be cooperating through the **Github** platform by using the Git versioning system.

Data Structures

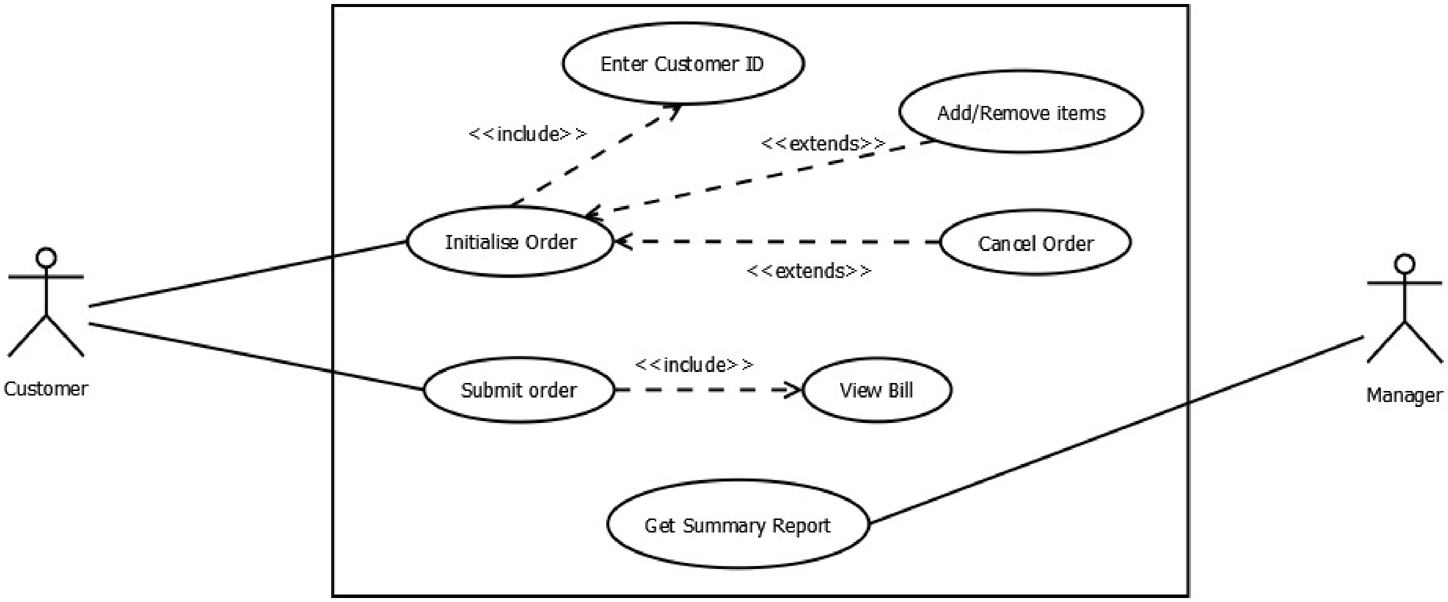
- `Inventory` - TreeMap (key: itemId, value: Item object)
  - keeps items sorted using the key (itemId: category+name)
  - relatively quick to search
  - adding and removing inventory items is expected to be rare
- `Orders` - ArrayList of Order objects
  - keeps orders sorted in the order they are added
  - easy to add new orders
  - iterating through orders only required at application exit
- `OrderItems` - ArrayList of Item objects
  - keeps order items in the order they were added
  - easy to add a new item to the end of the list
  - easy to access and remove items by index

Assumptions

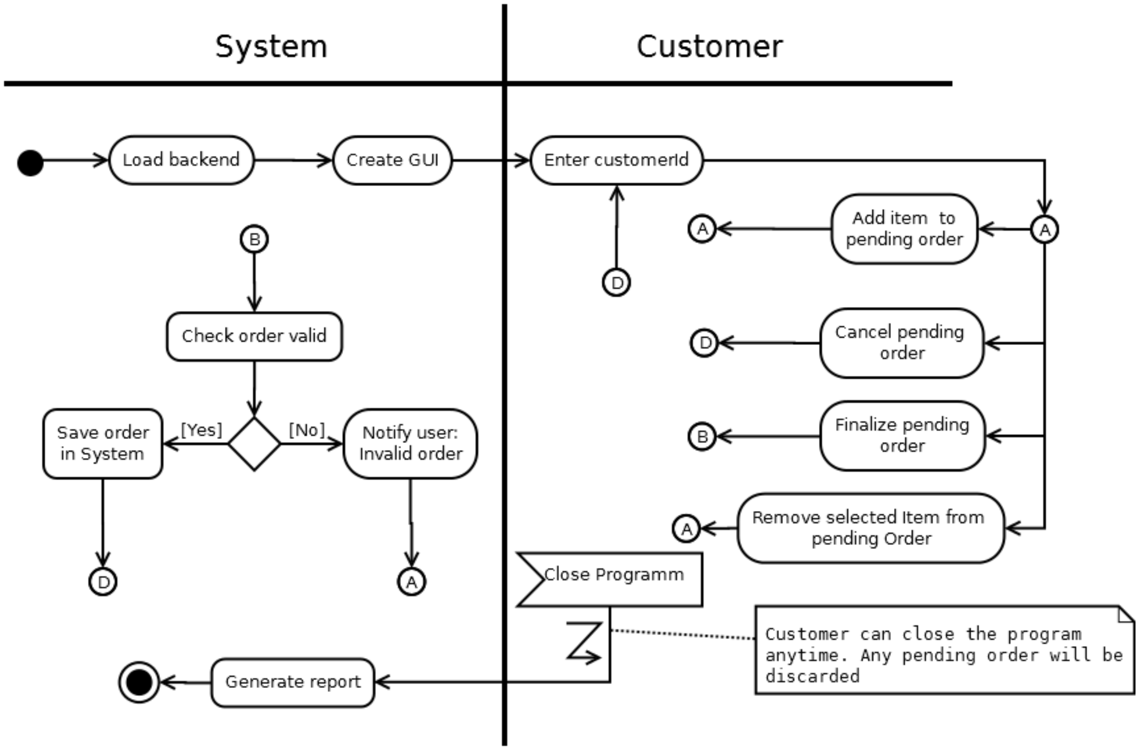
- Prices will *not be changed*, since that would cause inconsistencies between generated reports of different time periods.

- For the `TreeMap` data structure, items will be having an `id` with a name prefix along the lines of `<item_name><unique_id>` so as to provide both an alphabetically (pre)sorted list of items in the inventory, as well as efficient retrieval through their id being used as a key for the TreeMap. This further means that we do not expect item name's to be changed.
- Getter/Setter methods & data structures for method-local variables are *omitted* from the UML diagram as their usage will be dependent on the implementation and each will be used as-needed.
- The `main` running method of our application, as well as its corresponding class, are not shown in the UML since they have no ontological value.

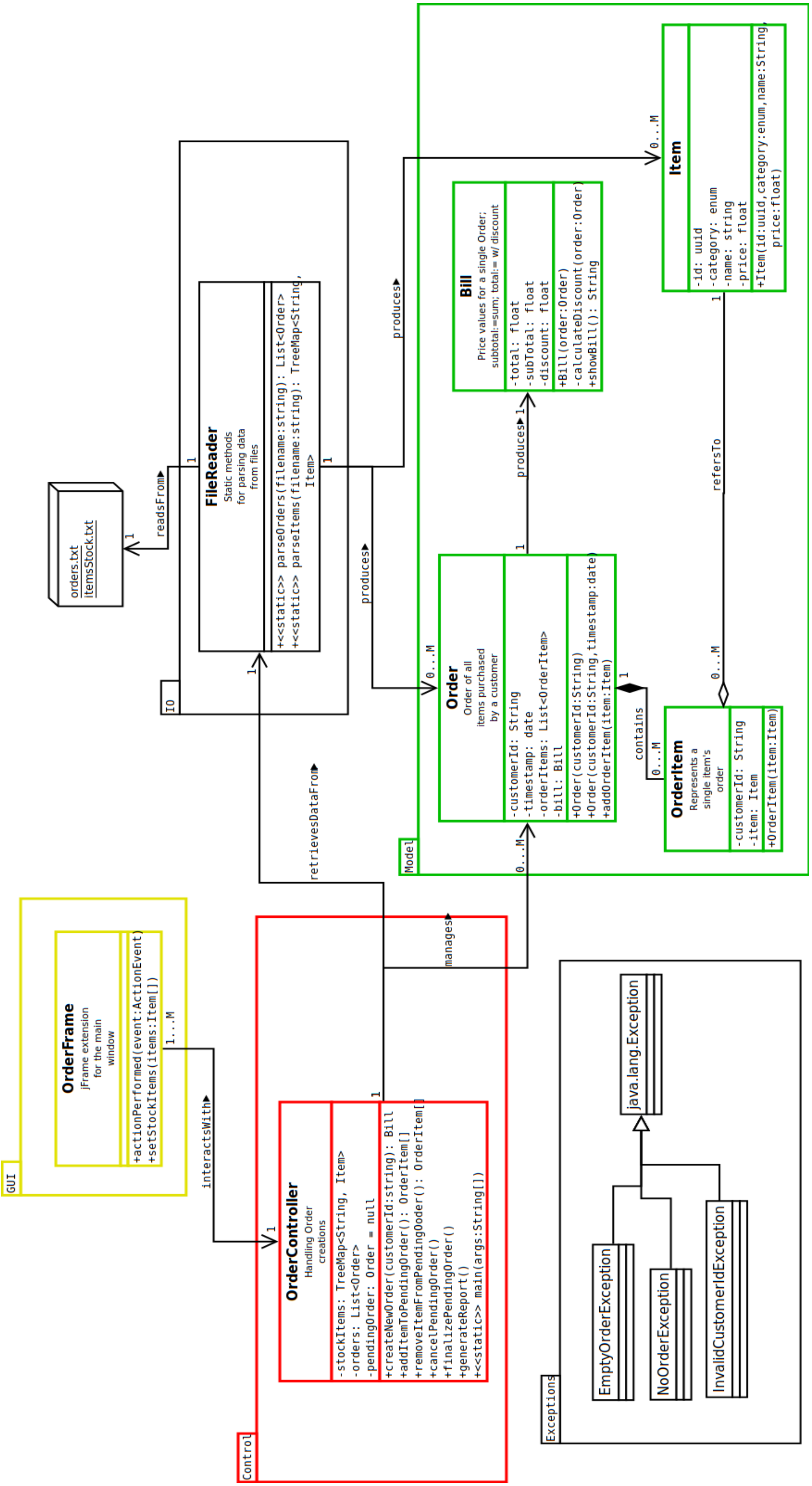
Use Case Diagram



Activity Diagram



Class Diagram



IO
orders.txt
itemsStock.txt
readsFrom▸
1
1
FileReader
Static methods
for parsing data
from files
+<<static>> parseOrders(filename:string): List<Order>
+<<static>> parseItems(filename:string): TreeMap<String, Item>
1
produces▸
produces▸
1
retrievesDataFrom▸
Model
Bill
Price values for a single Order;
subtotal:=sum; total:= w/ discount
-total: float
-subTotal: float
-discount: float
+Bill(order:Order)
-calculateDiscount(order:Order)
+showBill(): String
produces▸
1
1
Item
-id: uuid
-category: enum
-name: string
-price: float
+Item(id:uuid,category:enum,name:String, price:float)
0...M
1
refersTo
0...M
Order
Order of all
items purchased
by a customer
-customerId: String
-timestamp: date
-orderItems: List<OrderItem>
-bill: Bill
+Order(customerId:String)
+Order(customerId:String,timestamp:date)
+addOrderItem(item:Item)
0...M
1
contains
0...M
OrderItem
Represents a
single item's
order
-customerId: String
-item: Item
+OrderItem(item:Item)
0...M
manages▸
GUI
OrderFrame
jFrame extension
for the main
window
+actionPerformed(event:ActionEvent)
+setStockItems(items:Item[])
1...M
interactsWith▸
1
1
Control
OrderController
Handling Order
creations
-stockItems: TreeMap<String, Item>
-orders: List<Order>
-pendingOrder: Order = null
+createNewOrder(customerId:string): Bill
+addItemToPendingOrder(): OrderItem[]
+removeItemFromPendingOoder(): OrderItem[]
+cancelPendingOrder()
+finalizePendingOrder()
+generateReport()
+<<static>> main(args:String[])
Exceptions
java.lang.Exception
EmptyOrderException
NoOrderException
InvalidCustomerIdException