CSC 668 - POST GUI - Assignment 2

Group members: Mahdi (Matt) Massoodi, Pablo Escriva, Bijan Mahdavi, Marcus Wong, David Lopez, Izaac Garcilazo, Gary Straub

GitHub repository: https://github.com/sfsu-csc-668-spring-2019/post-assignment-new-group-6

Overview/Description

POST is a point of sale terminal in which customers can add store products using the GUI while being able to checkout and pay. While creating a new sale, managers are able to obtain a sales log for all the purchases made.

Discussion/Difficulties

We decided to use microservices to connect the post-server API to our POST system. We use these microservices to generate instances from within our system classes to better abstract the logic between API requests/responses and how the actual POST system works. The services are only implemented to communicate with the API. Once the system and services were built, we were able to wire up the GUI and provide the respective data accordingly. Some difficulties included issues in how much work the services needed to do with GSON. However, we ended up putting the GSON logic into the system to keep the microservices clean. Another implementation concern was wiring up the GUI with the backend system. Mainly, in how we should implement the logic in when the GUI starts and ultimately how the manager fires up the store.

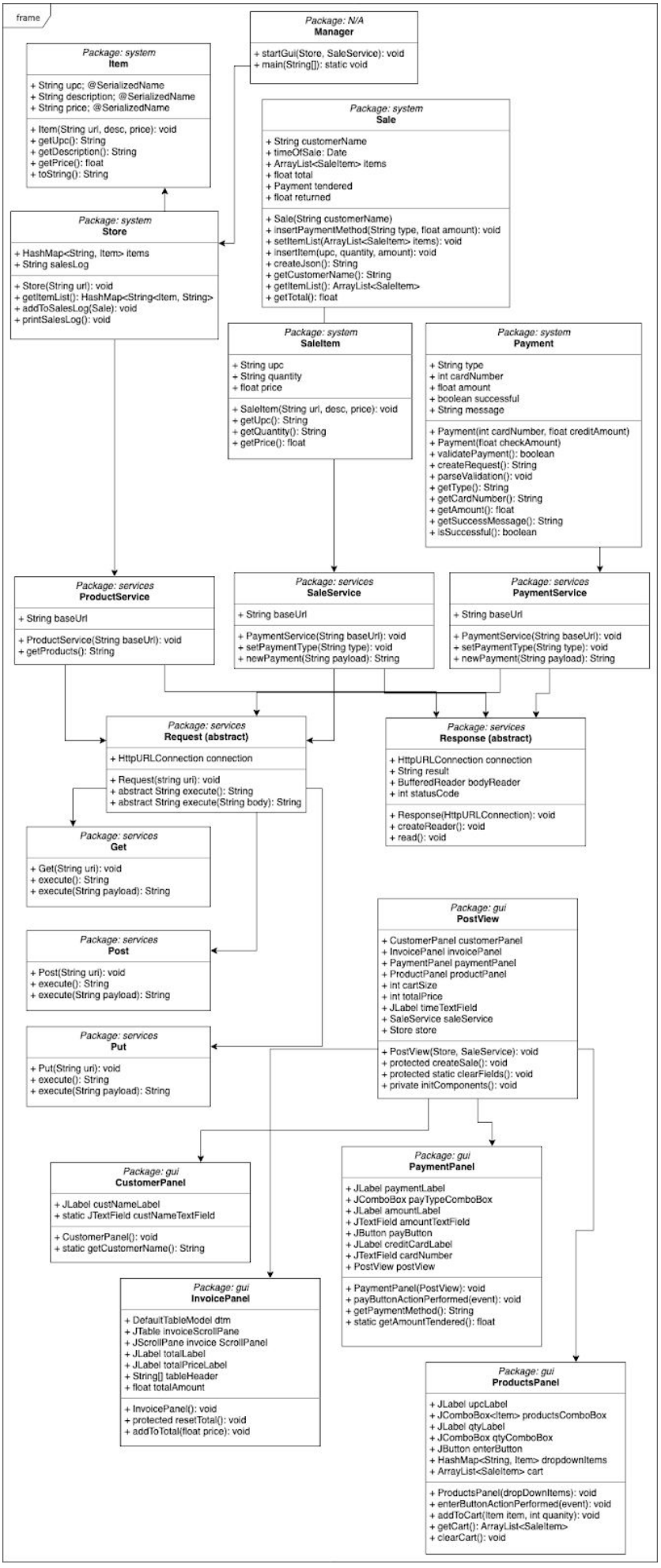
Class Design/Implementation

UML diagram link
(Screenshot in last page of the document)

In the UML diagram linked above, you can see three different packages; system, services, gui. Each one defines a specific part of our POST system. The system package contains functionality for running the GUI and handling data being passed between the services and gui, and vice versa. The services package is strictly for handling requests and responses between the API POST server. This is convenient for the system package because the abstraction cleans the functionality all together. The GUI uses the data being fetched from the system called on service methods. While on the other hand the GUI creates new records being put from the system called on their respective service methods. Our system package includes Manager (main), Store, Sale, SaleItem, Payment, and Item. These classes work throughout the entire app but mainly the GUI in bridging the gap between the services and the views. Our services package includes PaymentService, ProductService, and SaleService. Along with respective GET, POST, PUT, Request, and Response classes to communicate with the post-server API. The GUI class consists of the PostView, ProductsPanel, PaymentPanel, InvoicePanel, and CustomerPanel. These classes are used to abstract the end-user(s) input and interaction with the GUI. This allows our system to pass this data to the services for a more linear flow.

Output/Tests

Pictures and test cases


frame
Package: N/A
Manager
+ startGui(Store, SaleService): void
+ main(String[]): static void
Package: system
Item
+ String upc; @SerializedName
+ String description; @SerializedName
+ String price; @SerializedName
+ Item(String url, desc, price): void
+ getUpc(): String
+ getDescription(): String
+ getPrice(): float
+ toString(): String
Package: system
Sale
+ String customerName
+ timeOfSale: Date
+ ArrayList<SaleItem> items
+ float total
+ Payment tendered
+ float returned
+ Sale(String customerName)
+ insertPaymentMethod(String type, float amount): void
+ setItemList(ArrayList<SaleItem> items): void
+ insertItem(upc, quantity, amount): void
+ createJson(): String
+ getCustomerName(): String
+ getItemList(): ArrayList<SaleItem>
+ getTotal(): float
Package: system
Store
+ HashMap<String, Item> items
+ String salesLog
+ Store(String url): void
+ getItemList(): HashMap<String<Item, String>
+ addToSalesLog(Sale): void
+ printSalesLog(): void
Package: system
SaleItem
+ String upc
+ String quantity
+ float price
+ SaleItem(String url, desc, price): void
+ getUpc(): String
+ getQuantity(): String
+ getPrice(): float
Package: system
Payment
+ String type
+ int cardNumber
+ float amount
+ boolean successful
+ String message
+ Payment(int cardNumber, float creditAmount)
+ Payment(float checkAmount)
+ validatePayment(): boolean
+ createRequest(): String
+ parseValidation(): void
+ getType(): String
+ getCardNumber(): String
+ getAmount(): float
+ getSuccessMessage(): String
+ isSuccessful(): boolean
Package: services
ProductService
+ String baseUrl
+ ProductService(String baseUrl): void
+ getProducts(): String
Package: services
SaleService
+ String baseUrl
+ PaymentService(String baseUrl): void
+ setPaymentType(String type): void
+ newPayment(String payload): String
Package: services
PaymentService
+ String baseUrl
+ PaymentService(String baseUrl): void
+ setPaymentType(String type): void
+ newPayment(String payload): String
Package: services
Request (abstract)
+ HttpURLConnection connection
+ Request(string uri): void
+ abstract String execute(): String
+ abstract String execute(String body): String
Package: services
Response (abstract)
+ HttpURLConnection connection
+ String result
+ BufferedReader bodyReader
+ int statusCode
+ Response(HttpURLConnection): void
+ createReader(): void
+ read(): void
Package: services
Get
+ Get(String uri): void
+ execute(): String
+ execute(String payload): String
Package: services
Post
+ Post(String uri): void
+ execute(): String
+ execute(String payload): String
Package: services
Put
+ Put(String uri): void
+ execute(): String
+ execute(String payload): String
Package: gui
PostView
+ CustomerPanel customerPanel
+ InvoicePanel invoicePanel
+ PaymentPanel paymentPanel
+ ProductPanel productPanel
+ int cartSize
+ int totalPrice
+ JLabel timeTextField
+ SaleService saleService
+ Store store
+ PostView(Store, SaleService): void
+ protected createSale(): void
+ protected static clearFields(): void
+ private initComponents(): void
Package: gui
CustomerPanel
+ JLabel custNameLabel
+ static JTextField custNameTextField
+ CustomerPanel(): void
+ static getCustomerName(): String
Package: gui
PaymentPanel
+ JLabel paymentLabel
+ JComboBox payTypeComboBox
+ JLabel amountLabel
+ JTextField amountTextField
+ JButton payButton
+ JLabel creditCardLabel
+ JTextField cardNumber
+ PostView postView
+ PaymentPanel(PostView): void
+ payButtonActionPerformed(event): void
+ getPaymentMethod(): String
+ static getAmountTendered(): float
Package: gui
InvoicePanel
+ DefaultTableModel dtm
+ JTable invoiceScrollPane
+ JScrollPane invoice ScrollPanel
+ JLabel totalLabel
+ JLabel totalPriceLabel
+ String[] tableHeader
+ float totalAmount
+ InvoicePanel(): void
+ protected resetTotal(): void
+ addToTotal(float price): void
Package: gui
ProductsPanel
+ JLabel upcLabel
+ JComboBox<Item> productsComboBox
+ JLabel qtyLabel
+ JComboBox qtyComboBox
+ JButton enterButton
+ HashMap<String, Item> dropdownItems
+ ArrayList<SaleItem> cart
+ ProductsPanel(dropDownItems): void
+ enterButtonActionPerformed(event): void
+ addToCart(Item item, int quanity): void
+ getCart(): ArrayList<SaleItem>
+ clearCart(): void