1) Rate your comfort/proficiency level with the following programming languages (relative to each

other) using a scale of 1-5:

(1 = Highest, 2 = Good, 3 = Basic Knowledge, 4 = Knowledge of, 5 = No Experience)

* Java - 1
* Java script(Node.js and React) - 1
* Python - 2
* C++ - 3
* C - 3
* GoLang - 5
* C# - 5
* Perl - 5
* other (please specify)

2) Rate your domain interests using a scale of 1-3:

(1 = very interested, 2 = interested, 3 = no interest)

* Server Side Development - 2
* Web Application Development - 1
* System Software Development - 2
* Test and Quality Assurance - 3
* Technical Marketing - 3

**Related coursework for Web Application and Server Side Development** - Enterprise Distributed System

**Related project for Web Application and Server Side Development** -

I developed a prototype for the food delivery **grubhub** application as an individual project.   
Github Links -   
Lab 1 (React, Node Js, MYSQL) - <https://github.com/wamiquem/Lab1-013764595/tree/master/grubhub>  
  
Lab 2 (React, Node.Js, MongoDB, Redux, Kafka, Passport.js, JWT token) -   
<https://github.com/wamiquem/Lab2-013764595>

As part of the coursework, I have deployed the application on Amazon AWS EC2.  
I also deployed the application on heroku which is up and running.

The URL for the application is [https://grubhub-proto.herokuapp.com](https://grubhub-proto.herokuapp.com/#/)

Buyer login details: Username/Password - [wamique@sjsu.edu](mailto:wamique@sjsu.edu)/test123

As part of the group project, our team is required to develop a prototype of the **twitter** web application which will be completed within a month.

3) **Restaurant ordering application**

Language - **JAVA**

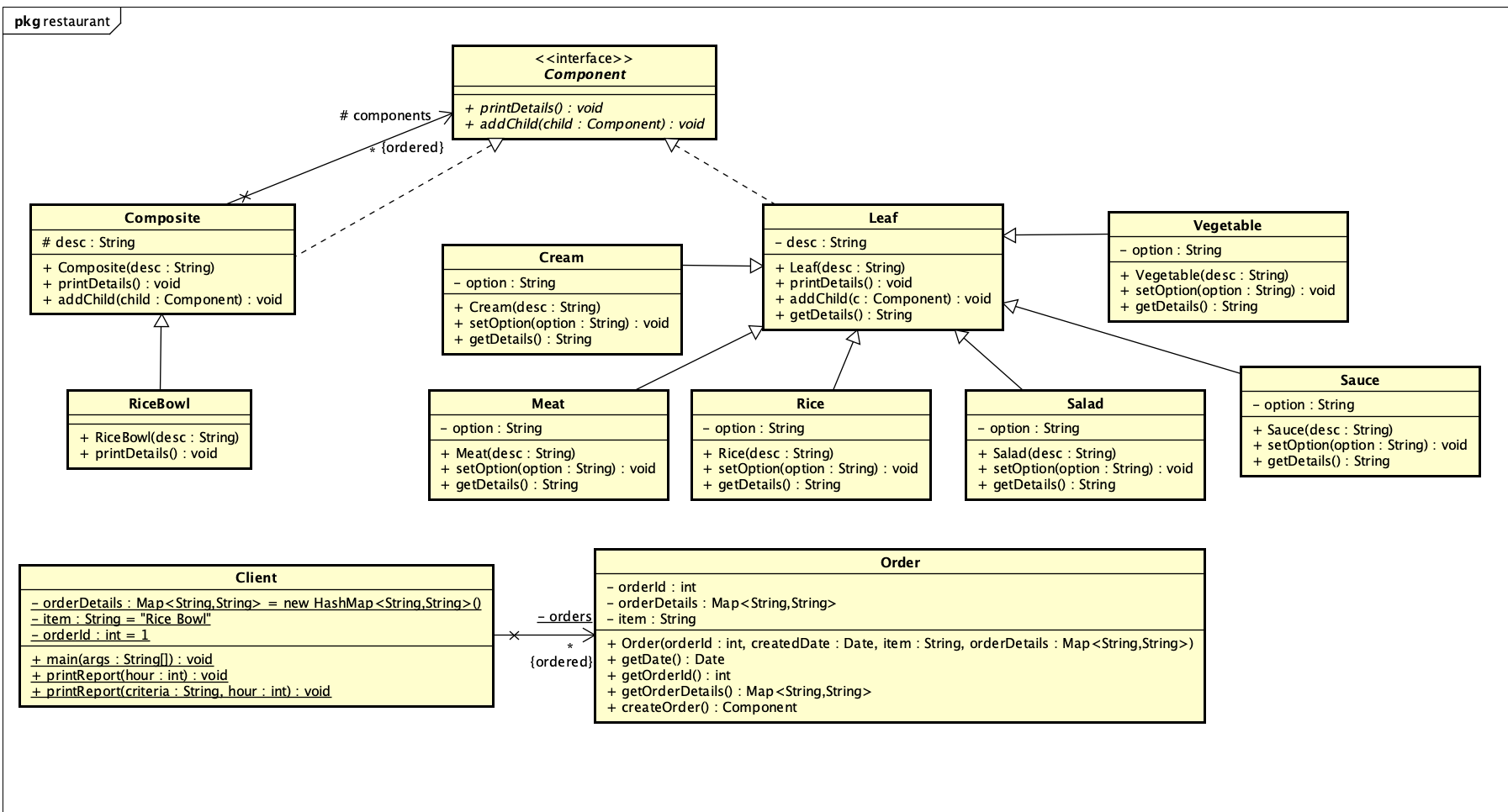
**Assumptions:**

* The client will ensure that proper validation of all mandatory and correct options(eg: Meat is mandatory and correct option is either Chicken or Vegetable but not both).
* The client will take care of the spell checks for all the available options.
* The client will keep track of the Order Id and pass it to the module.
* The client always sends valid order details to the createOrder() function.
* Assuming that this is a client - server model, all validations needs to be done on frontend itself, for instance, the order should contain all the available options for rice bowl like meat and sauce. Furthermore, the suboption should be validated by the client itself, for instance, Chicken option in Meat etc. The client passes the order details and the server processes the order.

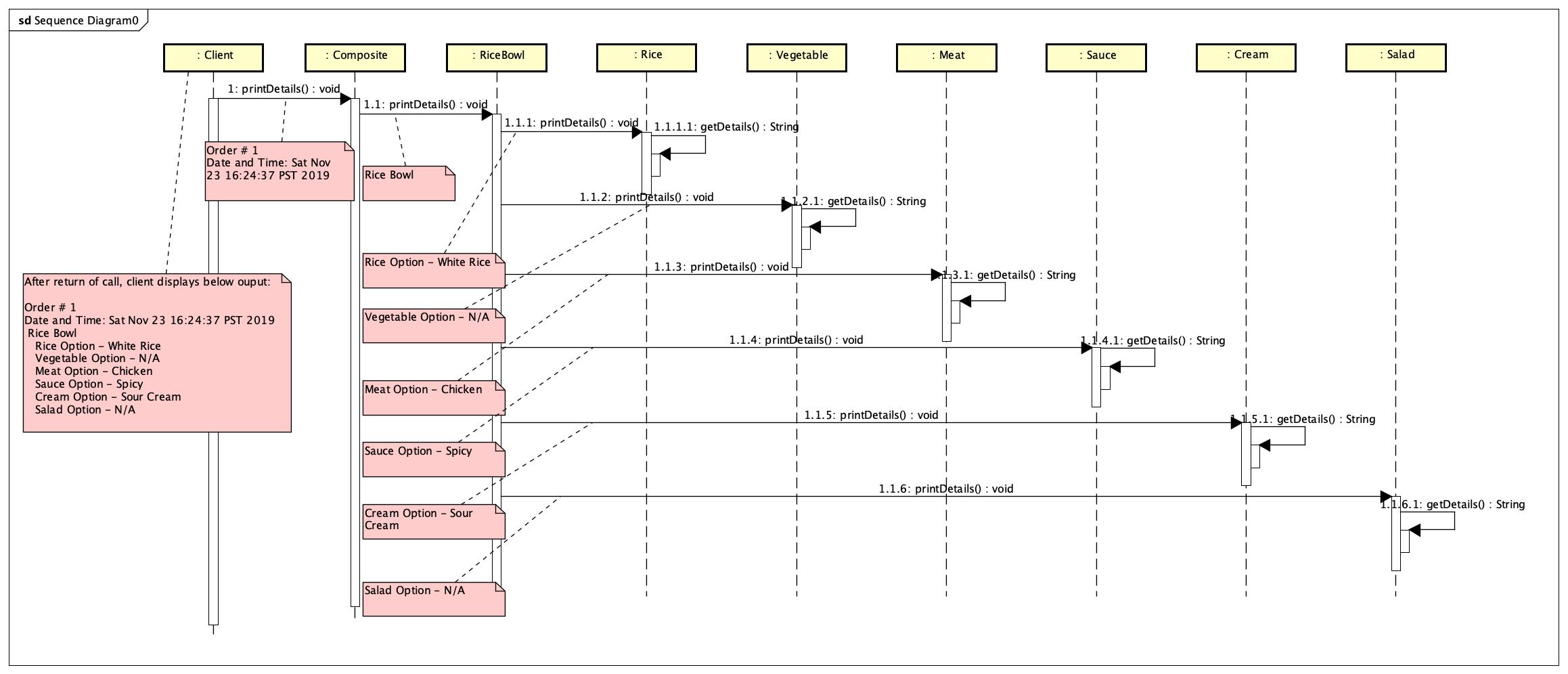
**Design:**

* The client(main method) will send all the order details with all selected options for the rice bowl.
* The client will call the createOrder() method implemented by the Order class in order to create the order.
* I have used the **Composite design pattern** in the implementation to keep the design for the item Rice Bowl and options simple, readable and to enforce the key OOP design principle **‘Single Responsibility Principle’**.
* When printing the order details, the composite design pattern will take care of aggregating multiple results into a single result.
* Composite design pattern will also allow reusability and expansion. For example, if in the future some option like beans needs to be added, we just need to add an extra subclass of the class Leaf and no other changes in the code will be needed.
* I have demonstrated various OOP key concepts like Inheritance, Polymorphism, Abstraction and Encapsulation in the implementation.
* Also, demonstrated the usage of Java in built data structures like HashMap and ArrayList.
* For reports, I have considered appending individual order objects to the List of objects which can be referred to print report summary.
* I have kept the printReport method open for the client to specify the hours within which the report needs to be generated.
* The Client takes 7 orders, print details of the created order and then prints the reports.
* Below is the class diagram and sequence diagram of the implementation. The sequence diagram depicts the message flow between various objects for printing details of an order.

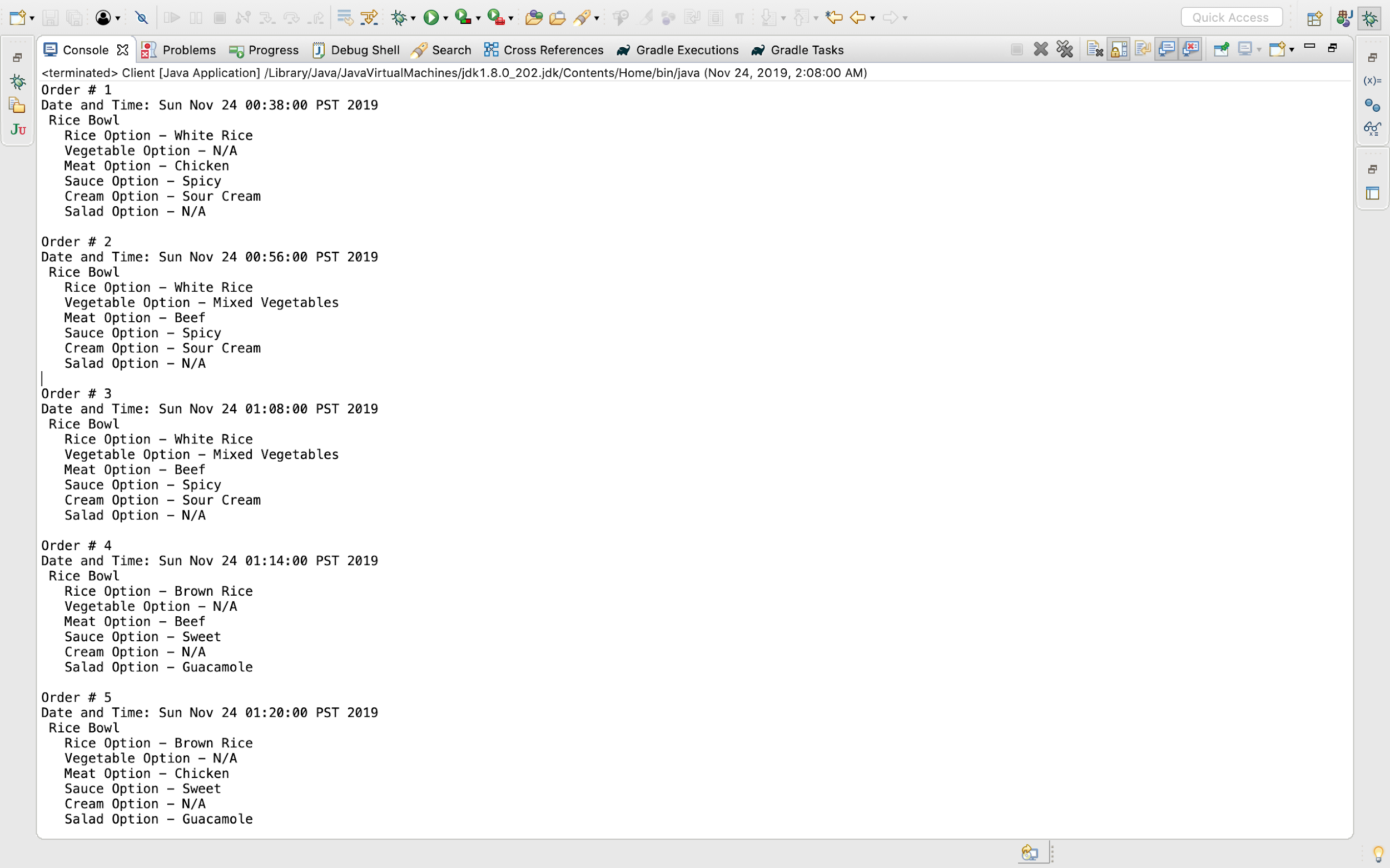
**Class Diagram:**

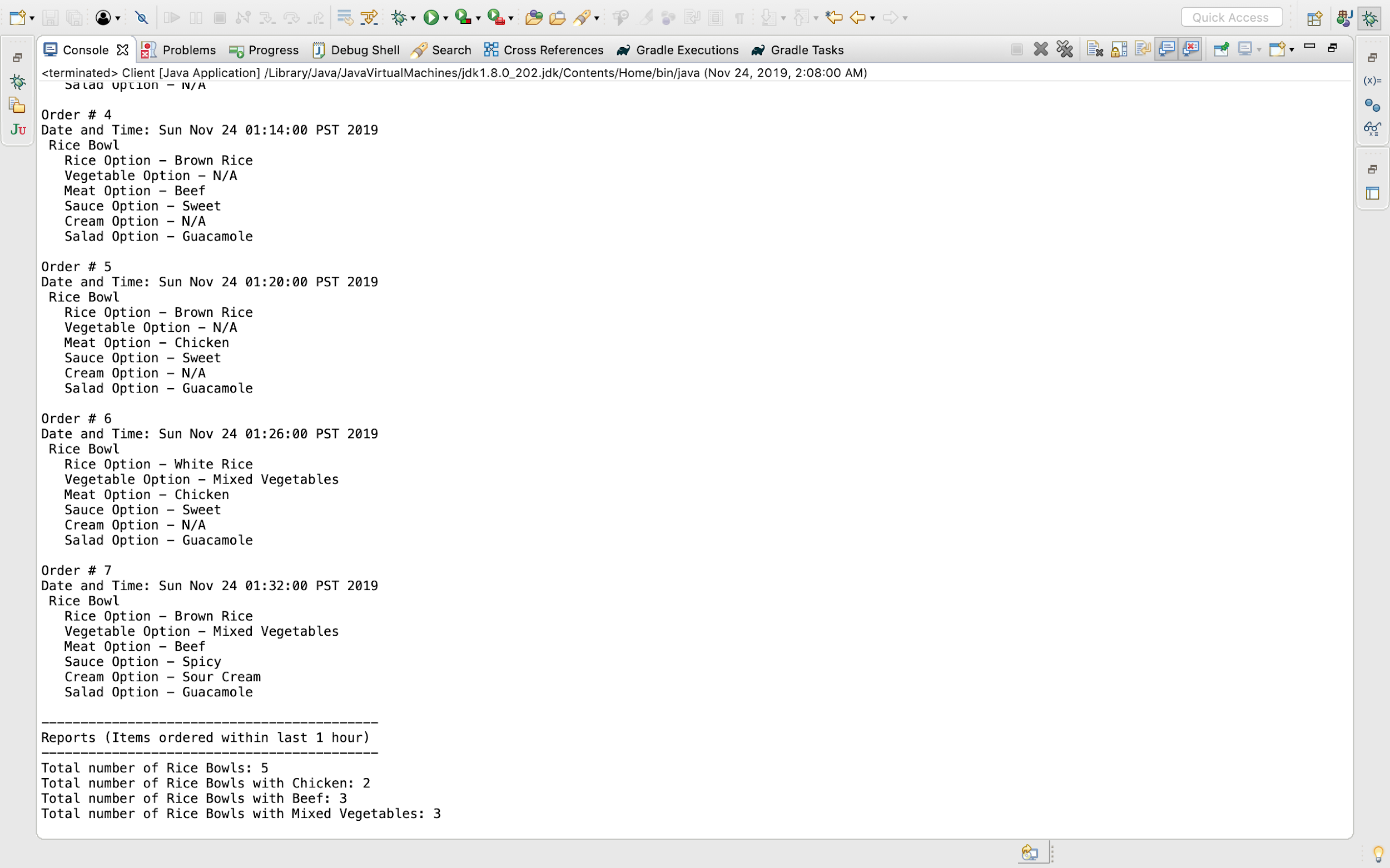
****

**Sequence Diagram:**

****

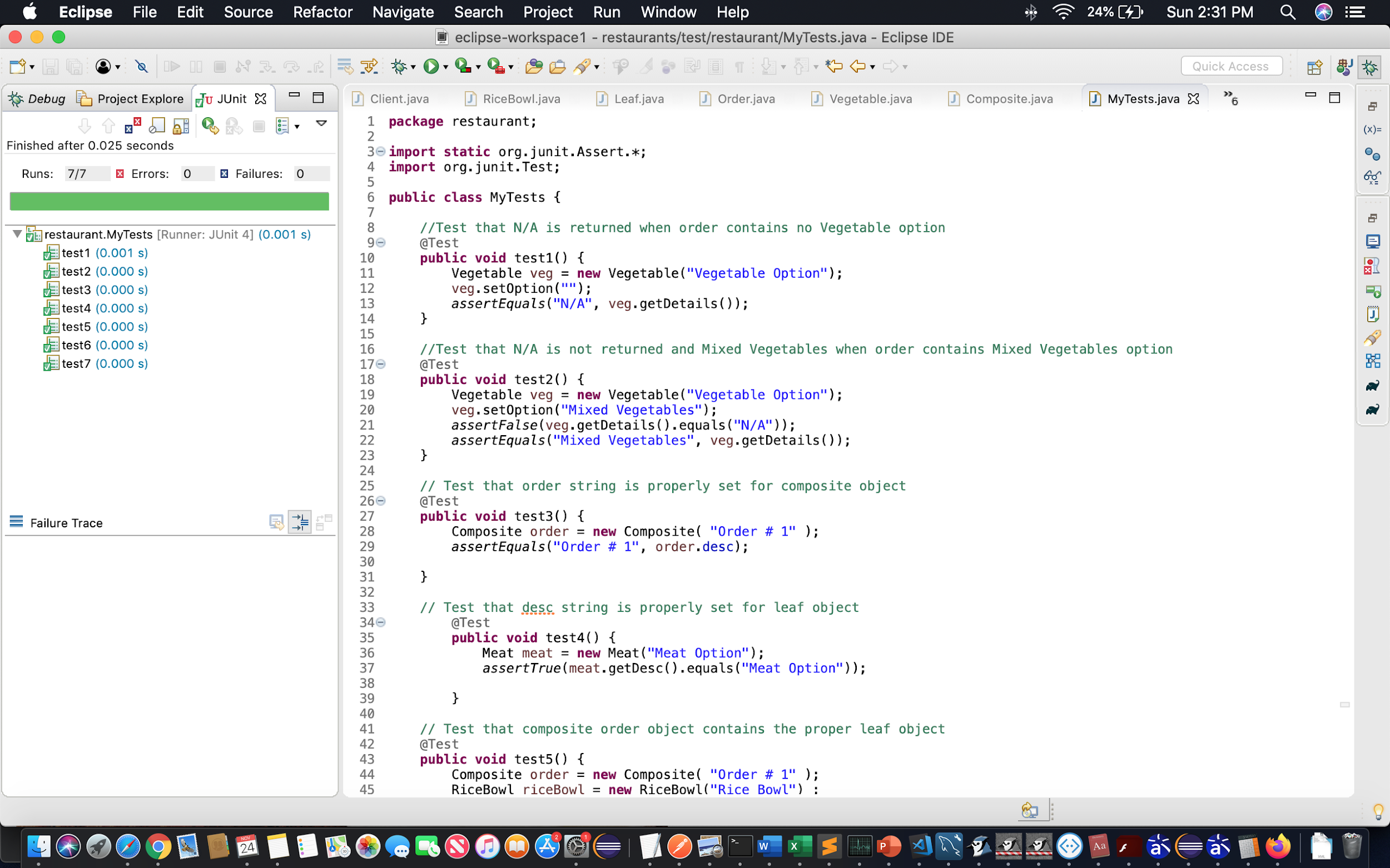
**Output of Execution:**

****

****

**Unit Test Cases:**

* I have written JUnit test cases to check for all individual module. As all leaf subclasses are similar, I have written unit tests for 1 leaf subclass.
* Unit test cases are also included in the attached zip file**.**
* Modules works as expected as all unit test cases are passing. Please find below screenshot of tests run:

****