Solution design:

* The facade pattern.

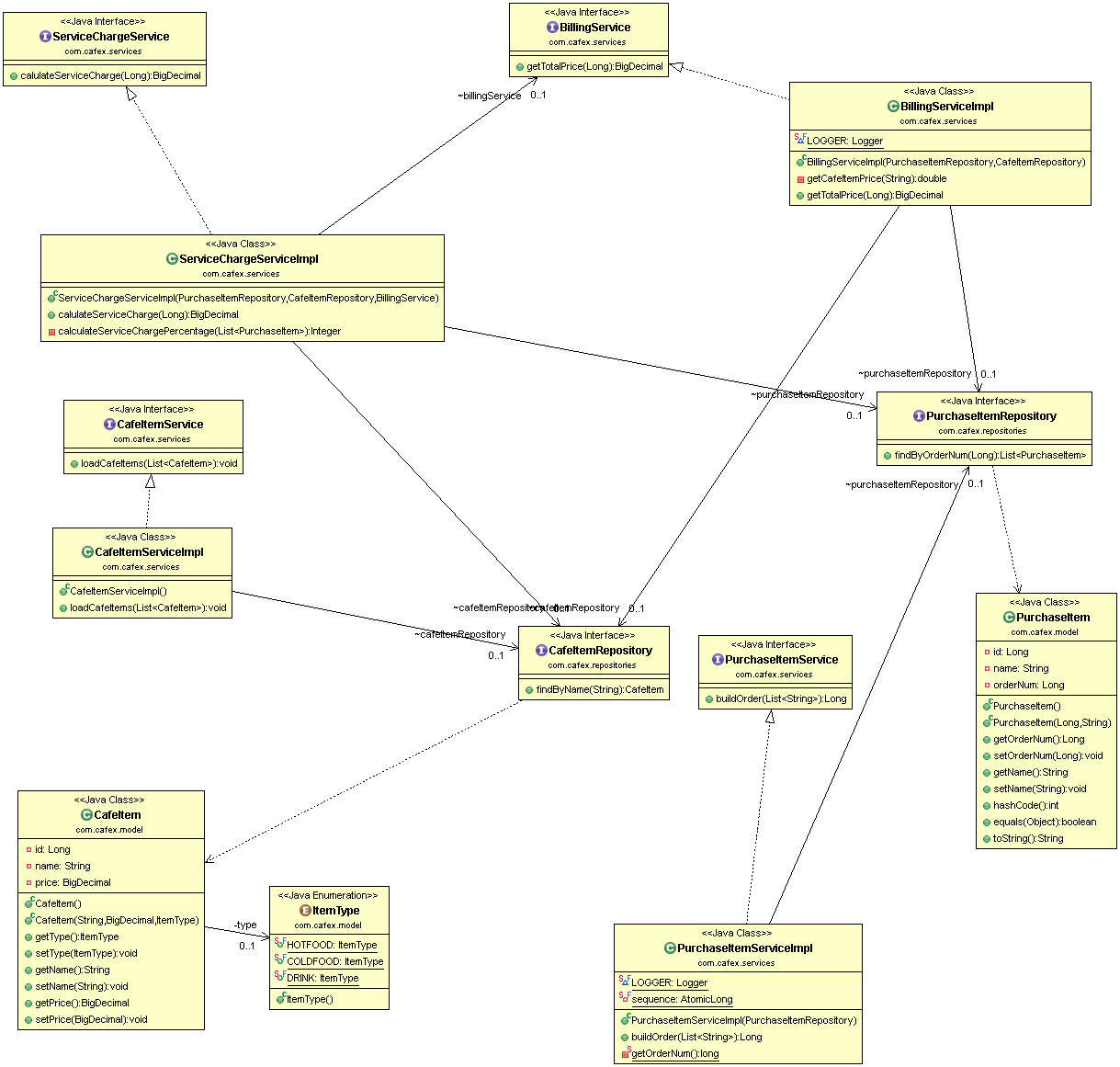
The facade pattern (also spelled façade) is a software design pattern commonly used with object-oriented programming. The name is by analogy to an architectural façade.A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can make a software library easier to use, understand and test. Facade provides convenient methods for common tasks,make the library more readable, for the same reason, reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system,wrap with a single well-designed API.

* Composite pattern.

The composite pattern is a partitioning design pattern, describes a group of objects that is treated the same way as a single instance of the same type of object. The intent of a composite is to "compose" objects into tree structures to represent part-whole hierarchies. Implementing the composite pattern lets clients treat individual objects and compositions uniformly

Technical design:

Class diagram



Build:

* Development Technology stack – Java8, Spring Boot, Spring and Spring data repository
* Test technology stack - Mockito
* Maven is used as a build tool.
* Eclipse Oxygen is used for development tool.
* GIT master branch is used for code repository.

Test:

* Download the source code and navigate to root (demo) directory which has POM.
* Run the Spring boot application with demo profile

mvn spring-boot:run -Drun.arguments="arg1,arg2"

(mvn spring-boot:run -Drun.arguments="Steak Sandwich,Cheese Sandwich")

a. Loads cafe reference data

b. Perform the test.

* To run the JUNIT test cases, please run the command below from root directory. Please note that data repositories are mocked to test individual components.

mvn test