Design Document

Submitted by: Abir Lal Saha

Object Oriented Programming (OOPS)principles followed:

Plain Old Java Object (POJO):

Dresser, HotDresser, ColdDresser, Command

Abstract class Dresser contains all the abstract methods which represents actions to be performed on getting a numeric command input.

Inheritance:

HotDresser class and ColdDresser class inherits properties by extending Dresser class

Design pattern:

1) Factory Design pattern has been used for creating Dresser Instances using DresserFactory class based on the first input whether it is HOT or COLD.

2) Singleton Pattern has been used to create a DocusignUtil class, which reads the property value of each of the constant defined in enum GlobalConstant from docusign.properties file

CommandProcessor:
InputParser handler object is created which process all the input commands by invoking the processCommand() and passing the input command array as well as the dresser object.

Based on each input command, a corresponding Command object is created which sets appropriate action name and code and returns the corresponding response to it.

Business Logic validation:

Each Command object created in passed through CommandValidator validate method to conform with the rules (business logic).

Logging:
Logger framework log4j has been used to do proper logging activity throughout the application. Developers can refer logs generated in logs folder to debug any exception or issues.

Build Framework:
Maven is used to manage dependencies and generating jar file.

Custom Exception:
Exception handling has been done by creating a custom DocusignCustomException Class.

Testing Framework:

jUnit test cases have been provided for testing different scenarios of the application.

Scalabilty:

Implementation has been performed in such a way that if developers want to add more temperatures like MILD, WARM etc. or any other activity other than the provided action they can simply add corresponding constants in enum GlobalConstant file and provide associated string values in docusign.properties file. More functionalities can be added in each of the POJO classes. Command object can be modified to give various responses depending on the numeric commands. Number of numeric commands can be added as well.

Class Diagram: