*Process MeNtOR 3.o*

*Uni-SEP*

Publisher-Subscriber System

CS2212B – 2019

Group 4

**Design Document**

|  |  |
| --- | --- |
| Version: | 1.5 |
| Print Date: | 4/7/2019 |
| Release Date: | 4/7/2019 |
| Release State: | Final |
| Approval State: | Approved |
| Approved by: | Group4 |
| Prepared by: | Xiaoxuan Yang  Yu Wu  Cheng Chen  Lishan Huang  Shuming Zhang |
| Reviewed by: | Group4 |
| File Name: | Group4\_SDD.doc |

**Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary of Changes** |
| 0.5 | 4/5/2019 | Lishan Huang | Brainstorm code implement |
| 1.0 | 4/6/2019 | Lishan Huang | Finished all use case requirement |
| 1.5 | 4/7/2019 | Lishan Huang | Debug and Formatting |

# Document Sign-Off

|  |  |  |
| --- | --- | --- |
| **Name (Position)** | **Signature** | **Date** |
| LiShan Huang |  | 4/7 |
| Cheng Chen |  | 4/7 |
| Yu Wu |  | 4/7 |
| XiaoXuan Yang |  | 4/7 |
| Shuming Zhang |  | 4/7 |

Table of Contents

[1 Introductions 4](#_Toc5544086)

[1.1 Overview 4](#_Toc5544087)

[1.2 Resources - References 5](#_Toc5544088)

[2 Major Design Decisions 6](#_Toc5544089)

[3 Architecture 7](#_Toc5544090)

3.1 The Component Diagram

3.2 The Component Description

[4 Detailed Class Diagrams 8](#_Toc5544091)

[4.1 UML Class Diagrams 8](#_Toc5544092)

[4.3 Package Description 11](#_Toc5544093)

1. Introductions
   1. Overview

The programming tasks for this project are summarized as follows:

1. **Implement the Event Factory and three or more Concrete Event Types**

2. **Implement three or more Concrete States for subscribers**

3. **Implement three or more Concrete Strategies for publishers**

4. **Implement the Singletons for** Channel Access Control

Channel Creator

Channel Discovery

Channel Event Dispatcher

Subscription Manager

5. **Notify subscriber only if the subscriber is not blocked in a Channel**

This SDD included:

1. **Major Design Decisions**
2. **Architecture**
3. **UML Class Diagram**
   1. Resources - References

<https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern>, and <https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg15pub.htm>

1. Major Design Decisions

**The 1st decision:** we understand state as related to subscriber, so we merged the two classes.

**The 2nd decision**: the operation instructions are based on a console non-graphical interface.

**The 3rd decision:** we enable eclipse as a compilation software

**The 4th decision:** We found that using Arrarylist to store information during the final phase of the project was inefficient, especially in finding the relationship between each subscriber's ID and Event. On April 6th we overthrew the original data structure and finally chose HashMap to store various data.

1. Architecture

This project uses the publisher-subscriber architecture style.

### The Component Diagram for the system is show below:

A close up of a piece of paper

Description automatically generated

### The Component Diagram Description:

|  |  |  |
| --- | --- | --- |
| **Strategy** | doPublish and create strategy | IStrategy |
|  |  | StrategyFactory |
|  |  | StrategyName |
|  |  |  |
| **Publisher** | Create publisher | AbstractPublisher |
|  |  | PublisherFactory |
|  |  | ConcretePublisher |
|  |  | PublisherType |
|  |  |  |
| **Event** | createEvent | AbstractEvent |
|  |  | EventType |
|  |  | EventFactory |
|  |  | EventTypeA |
|  |  | EventMessage |
|  |  | EventIDMaker |
|  |  | EventTypeB |
|  |  |  |
| PubSubServer | Subscribe and unsubscribe | EventDispatcher |
|  | postEvent | ChannelPoolManager |
|  | notifySubscriber | ChannelCreator |
|  | createChannel | ChannelDiscovery |
|  | blockSubscriber | SubscriptionManager |
|  | unblockSubscriber | Channel Access Control |
|  |  | AbstractChannel |
|  |  | Channel |
|  |  |  |
| Subscriber | Createsubscriber | AbstractSubscriber |
|  |  | Subscriber Factory |
|  |  | ConcreteSubscriber |
|  |  | SubscriberType |
|  |  |  |
| State | addEventintofavorlist | IState |
|  | HandleEven | StateName |
|  | createState | StateFactory |
|  |  |  |
| Orchestration | createsubscriber | PubSubEntity |
|  | createpublisher | OrchestrationTest1 |
|  |  | Channel.chl |
|  |  | Input.txt |
|  |  | State.sts |
|  |  | Strategy.sts |

1. Detailed Class Diagrams

## UML Class Diagrams

A close up of text on a white background

Description automatically generated

### The tables below show the list of data members

***Subscribers***

Data Members

|  |
| --- |
| **protected** IState state; |

Methods

|  |
| --- |
| **public** **void** setState(StateName stateName) |
| **public** **void** alert(AbstractEvent event, String channelName) |
| **public** **void** subscribe(String channelName) |
| **public** **void** unsubscribe(String channelName) |
| **public** **static** AbstractSubscriber createSubscriber(SubscriberType subscriberType, StateName stateName) |
| **public** **SubscriberType** type() |

***States.Subscribers***

Data Members

|  |
| --- |
| **protected** LinkedList<AbstractEvent> favor |

Methods

|  |
| --- |
| **public** **void** handleEvent(AbstractEvent event, String channelName) |
| **public** **boolean** transport () |
| **public** **boolean** food() |
| **public** **boolean** unset() |
| **public** **boolean** all() |
| **public** StateName state() |

***Publishers***

Data Members

|  |
| --- |
| **protected** IStrategy publishingStrategy |

Methods:

|  |
| --- |
| **public** **void** publish(AbstractEvent event) |
| **public** **void** publish() |
| **protected** ConcretePublisher(IStrategy concreteStrategy) |
| **public** **static** AbstractPublisher createPublisher(PublisherType publisherType, StrategyName strategyName) |

***pubSubServer***

Data Members

|  |
| --- |
| **public** **static** ChannelAccessControl *instance*; |
| **private** Set<AbstractSubscriber> subscribers ; |
| **private** String channelTopic; |
| **private** Queue<AbstractEvent> events ; |

Methods:

|  |
| --- |
| **protected** **void** publishEvent(AbstractEvent event); |
| **protected** **void** subscribe(AbstractSubscriber subscriber); |
| **protected** **void** unsubscribe(AbstractSubscriber subscriber);  **public** String getChannelTopic() |
| **public** **static** AbstractPublisher createPublisher(PublisherType publisherType, StrategyName strategyName) |
| **private** **void** notifySubscribers(AbstractEvent event) |

***strategies.publisher***

Data Members

|  |
| --- |
| **int** publisherId; |
| AbstractEvent event; |
| **int** strategy; |
| List<String> channelListPub; |
| List<String> mainChannelList; |
| **private** DefaultStrategy ds; |

Methods:

|  |
| --- |
| **public** **void** doPublish(**int** publisherId) |
| **public** **void** doPublish(AbstractEvent event, **int** publisherId) |
| **public** **int** getpid() |
| **public** List<String> getMainList() |
| **public** FoodStrategy() |
| **public** **static** IStrategy createStrategy(StrategyName strategyName) |
| **public** TransportStrategy() |

## Package Description

* + 1. Event Package

This package is used to create events and give each event a unique kind. Each event has its own publisher and payload.

* + 1. Orchestration

This package is used to test our codes.

* + 1. Publisher

This package is used to create new publishers and publish Strategy-based events.

* + 1. pubSubServer

This package has the following features: 1. Add the published event to the corresponding channel. 2. notify the eligible users, while controlling the subscription and unsubscribing in the channel.

* + 1. states.subscriber

This package has the following features: Subscriber's current status tag: food, transport, unset, all.

* + 1. strategies.publisher

Use hashcode to track each type of strategy and contact the publisher. The events that are published depend on the specific strategy.

* + 1. subscribers

Set and generate subscriber ID, categorize based on subscriber type