|  |  |
| --- | --- |
| **NOMBRE** | Denis Rodriguez Mendoza |
| **Proyecto**: emailClientPatterns | **IDE**: IntelliJ IDEA – CE **Diagramador:** BlueJ |
| **Lenguaje**: Java – EE 8 |  |

**DETALLE DE CLASES**

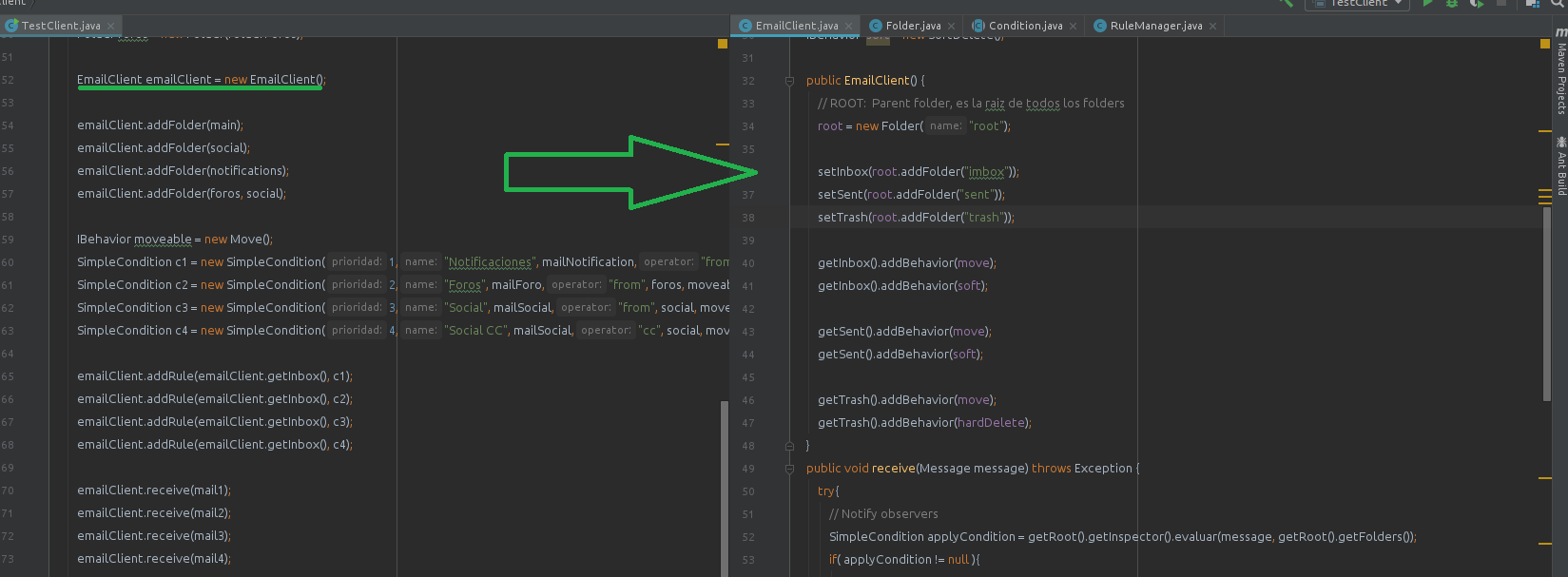
|  |  |  |
| --- | --- | --- |
| **Package** | **class** | **Observación** |
| **Behaviors** | IBehavior | Iterface de comprotamientos |
| HardDelete | Eliminar message de la app |
| SoftDelete | Comportamiento de mover el message a TRASH |
| Move | Comportamiento de mover elemento |
| **elements** | IContainer | Interface contenedor |
| Folder | Estructura de folder |
| Message | Estructura de message |
| EmailClient | Cliente de la app |
| RuleManager | Manager que aplica las rules asociadas a los folders creados |
| **rules** | Condition | Abstract de condiciones que se crearan en la app, se tiene planificado que la app soporte condiciones simples y condiciones Complejas |
| SimpleCondition | Estructura de una condición simple |

**Diagrama de clases:**

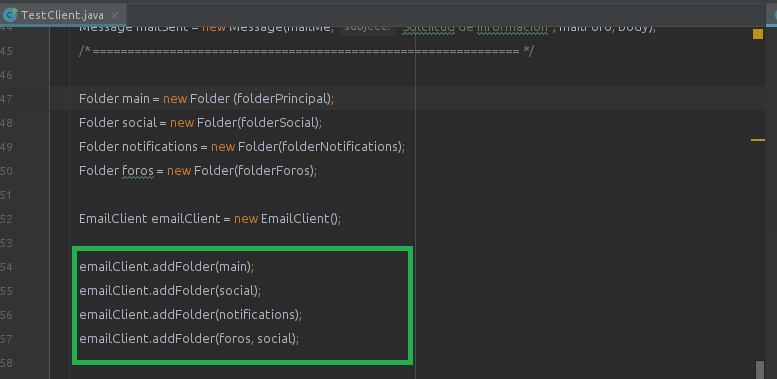
**Descripción de implementacion:**

Se tiene la main class “TestClient”, la cual se encarga de preconfigurar toda la app y ejecutar toda la secuencia

1. Crea una instancia de la clase EmailClient, la cual al crearse crea por defecto los folders base: root dentro del cual están los folders inbox, sent y trash, root es un folder que nunca se ve y al cual no se le pueden añadir emails, solo folders. Patron composite usado para poder crear la jerarquía de folder y asi hacer que cada folder tenga las mismas características que su padre, excepto el folder root



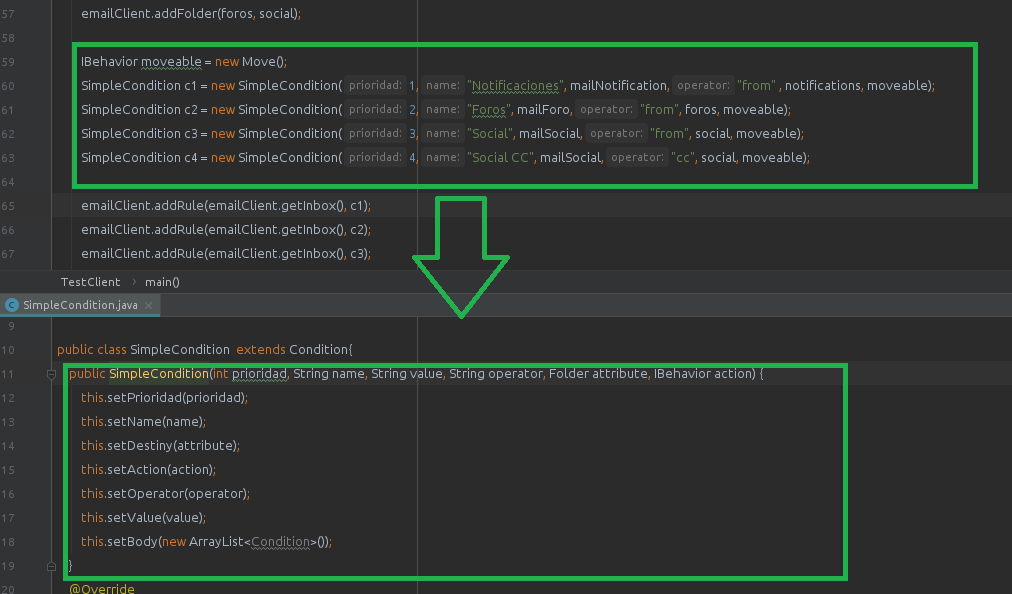
1. Se crean los folder de primer nivel: DESTACADOS, SOCIAL, NOTIFICATIONS, FOROS



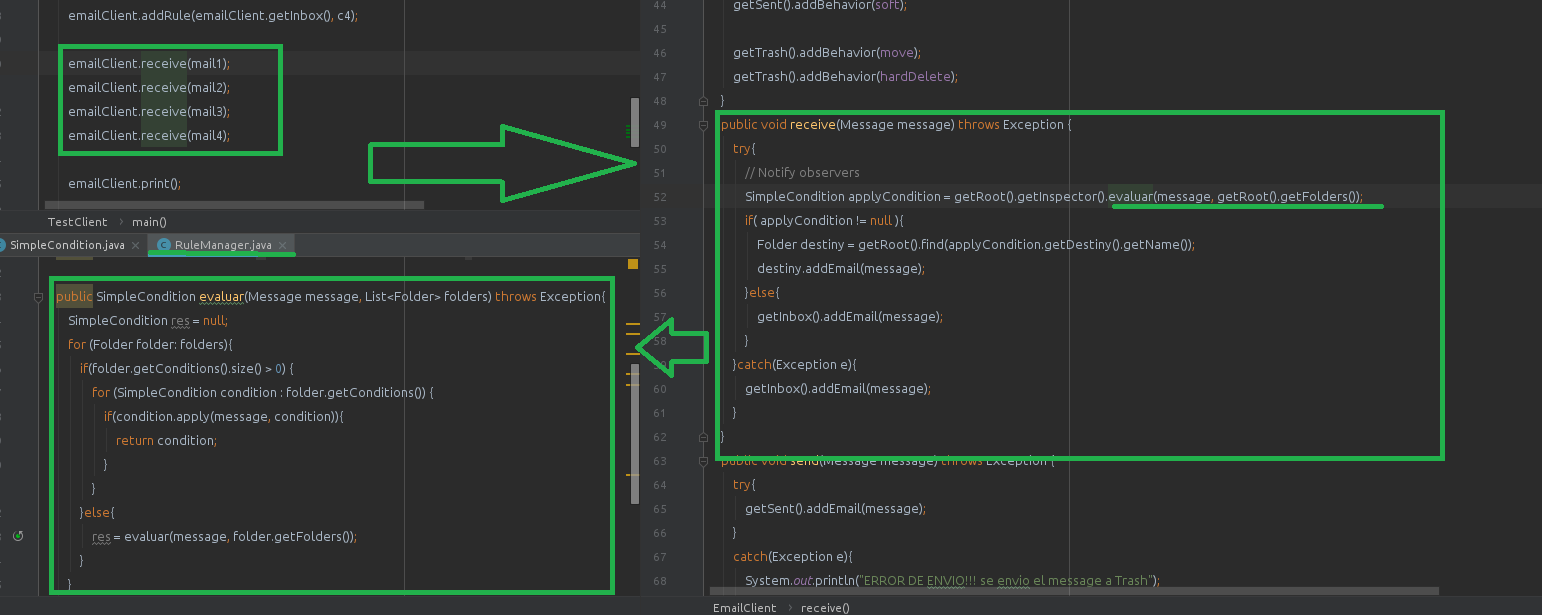
1. Se crean emails de 4 de prueba, ya que en este realese no se logró implementar la obtención automática de emails



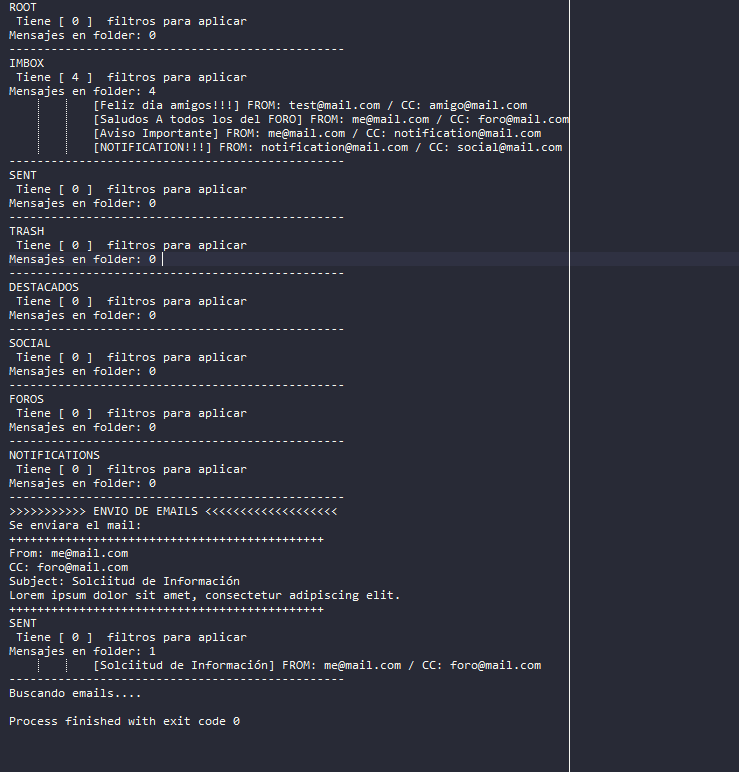
1. Crea las 4 reglas y las aplica al folder “inbox”, ya que según el enunciado del problema todos los mails llegan a este folder, es lógico pensar que en primera instancia solo se aplican las reglas a este folder. Aplicando template method se crean la abstract class “Condition” el cual tiene el esqueleto que las subclases deberán implementar, tal el caso de la clase “SimpleCondition” que estiende este clase abstarcta e implementa su método apply().



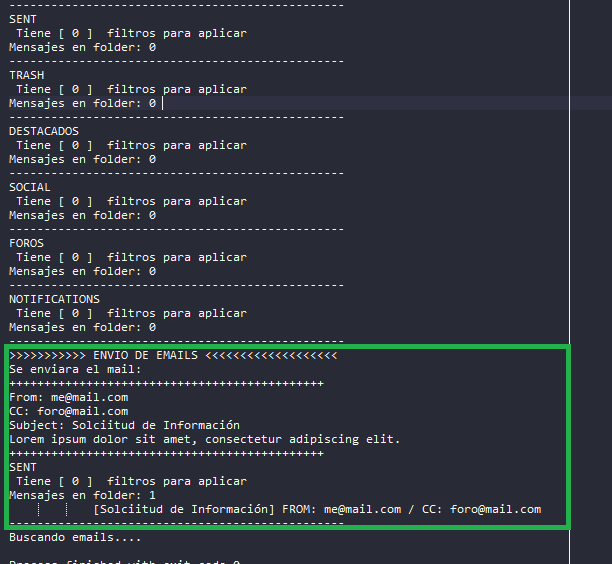
1. Se hace el envió de email, por medio del método receive de la clase EmailClient, el cual al ser un **observador** notifica a sus **observados**(folder) que llegaron nuevos mails y deben actuar para aplicar sus reglas, en el lado de los folder existe un **ruleManager** el cual se encarga de evaluar cada email con la lista de rules del folders. Como resultado de la evaluación se obtendrá la acción(IBehavior) que se debe tomar con el email, para ubicarlo en la carpeta que corresponda según la regla que se aplique.



1. Se mostrara el estado de la recepción y la aplicación de reglas

****

1. Finalmente para la corrida se enviara un mail el cual debera terminar en el folder SENT

****

Corrida de ejemplo: