Proyecto Call Center

Almundo.com

Olmedo Arcila Guzmán

Planteamiento:

Existe un Call Center donde hay 3 tipos de empleados: operador, supervisor y director. El proceso de la atención de una llamada telefónica en primera instancia debe ser atendida por un operador, si no hay ninguno libre debe ser atendida por un supervisor, y de no haber tampoco supervisores libres debe ser atendida por un director.

Solución:

La solución propuesta para el problema del ejercicio consiste en un programa escrito en Java 8, en el cual se soluciona de forma metódica el problema planteado. El programa está desarrollado utilizando el IDE NetBeans en su versión 8.2, además según los requerimientos del proyecto se usa maven como gestor del proyecto. Para la visualización de información se utilizó JavaFx para la construcción del GUI, utilizando Scene Builder 2.0.

Junto con la solución al problema se escribieron los módulos de prueba que solicitaban en el ejercicio utilizando JUnit como framework de pruebas, las pruebas unitarias se enfocaron sobre la clase Dispatcher.

Programa CallCenterFX

El programa que implementa la solución al problema, al estar escrito usando JavaFX, tiene unas particularidades de diseño como el uso del patrón MVC y el manejo de eventos.

La capa de Controlador está formada por la clase FXMLController, la cual tiene la lógica de manejo de solución al problema de forma macro, se acopla con la capa de vista mediante el archivo Scene.fxml, en el cual están definidos los objetos gráficos de la interfaz.

La capa de Vista fue desarrollada utilizando Scene Builder y consiste en una sencilla interfaz gráfica que permite visualizar de forma efectiva el estado de los empleados, las llamadas que van llegando a través del tiempo y un poco de información estadística sobre las llamadas atendidas por los empleados.

La capa del Modelo está formada por las clases que se encargan de manejar la lógica de la solución, la clase más importante para el manejo de la lógica es la clase Dispatcher, pues ella se encarga del manejo de los empleados y la recepción de nuevas llamadas. Para modelar los

empleados se utilizó una jerarquía de clases de la cual la base es la clase Empleado. Esta clase modela el comportamiento que debe tener un empleado cuanto atiende una llamada que le es asignada. En cuanto al modelado esta es una clase abstracta que además hereda de Thread, para permitir que varios objetos de esta clase trabajen de modo concurrente sin interrumpir el funcionamiento del dispatcher. De la clase Empleado se especializaron tres (3) subclases, las cuales permiten modelar cada uno de los roles de empleados que tiene el Dispatcher, siendo estas las subclases Operario, Supervisor y Director. Junto a estas está la clase Llamada que modela una llamada que entra al Call Center y tiene su respectivo tiempo de duración. Finalmente, están las clases que sirven para crear los Empleados y las llamada que requiere el programa, estas están diseñadas pensando en el patrón Factory para la creación rápida de objetos, junto con una clases utilitarias para el manejo de operaciones que permiten organización de la información siendo estas clases Util y HTMLUtil.

El diagrama de clases completo se presenta en la figura No. 1.

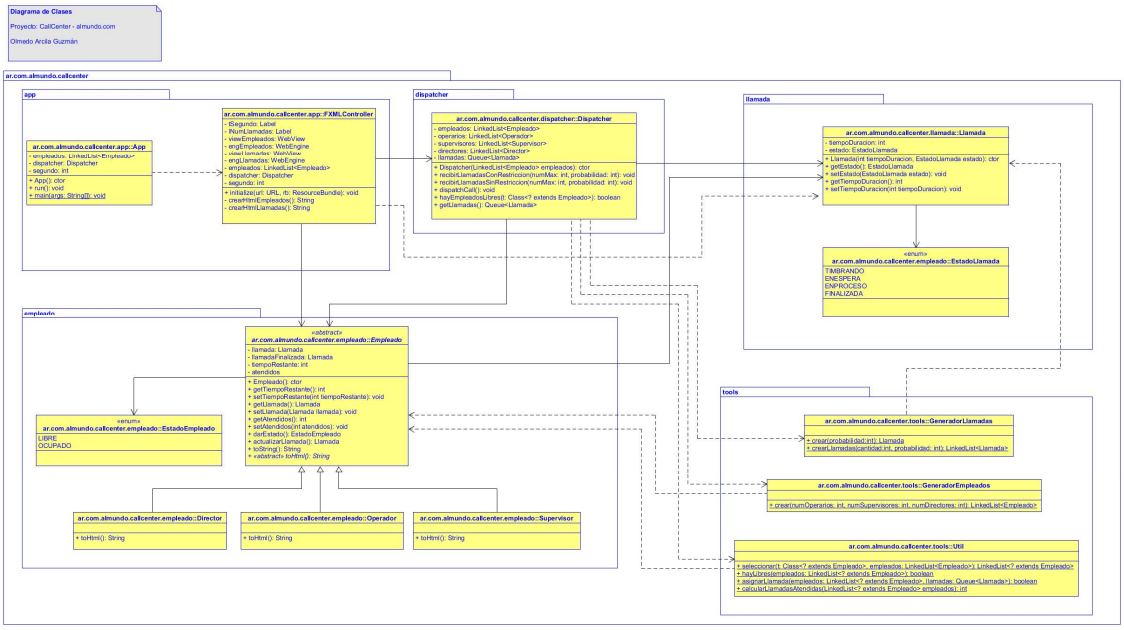

Figura No. 1: Diagrama de clases del proyecto.

La solución propuesta se basa en una solución de atención de las llamadas entrantes con una estructura FIFO, basada en una cola de llamadas entrantes al Call Center, de tal forma que todas las llamadas entrantes son puestas en la cola de espera para que no se pierda ninguna llamada. Una vez se tienen todas las llamadas que llegan en un instante determinado se procede a asignar a los empleados según la disponibilidad siguiendo de forma estricta el siguiente orden: i) primero se asignan a los operarios, ii) en caso de no haber operarios disponibles las llamadas se le asignan a los supervisores, iii) en cado que tampoco haya un supervisor disponible la llamada se le asigna a los directores y iv) en caso que todos los usuarios estén ocupados la llamada queda en la cola a la espera de un empleado disponible según el criterio establecido.

La lógica de la descripción anterior la realiza la clase Dispatcher.

La clase Distpacher

Esta clase está compuesta por una lista de llamadas, en la cual se van a colocar todas las llamadas que llegan al Call Center, esta lista funciona como una Cola y de ella solo van a ser llamados los métodos add (encola al final de la lista) y pool (desencola de la cabeza de la lista). Luego tienen una lista principal de empleados, en la cual se almacenan en orden los empleados según su tipo, primero están los operarios, luego los supervisores y al final los directores, esto se hace para tener control sobre el orden de asignación de llamadas, junto a esta están unas listas auxiliares que tienen solo empleados del mismo tipo, es decir una lista de operarios, otra de supervisores y una última de directores.

Una vez están creados los empleados y catalogados en sus respectivas listas, se pueden empezar a atender llamadas, las llamadas se generan de forma aleatoria en el método recibirLlamadasSinRestriccion(int numMax, int probabilidad). En este método se crea un número de llamadas controlador por numMax que es un tope superior y la variable probabilidad sirve como elemento que limita la creación de las llamadas de tal forma que el número de llamadas creadas en una invocación no es necesariamente igual a numMax pero si es menor a este. Las llamadas que se crean se almacenan en la cola de espera y de allí son tomadas en el método dispacthCall(). En este método se asignan las llamadas según el criterio definido, primero se verifica si hay operarios disponibles, luego si hay supervisores, luego directores y finalmente sino hay nadie disponible se dejan las llamadas en la cola de espera.

Las clases Empleado, Operador, Supervisor y Director

Estas clases forman una jerarquía de herencia, en la cual Empleado es la clase padre y las clases Operador, Supervisor y Director son subclases. La lógica de atención de una llamada está implementada en la clase Empleado, pues las llamadas se atienden de igual forma sin importar quien la toma, pero para efectos de visualización cada una implementa el método abstracto toHTML(), definido en Empleado, así se logra que cada empleado se vea diferente en la GUI.

Empleado además de ser la clase principal de la jerarquía, es un hilo, lo cual permite que cada empleado tenga su propia línea de tiempo permitiendo que las llamadas sean atendidas de forma concurrente durante la ejecución del programa y así el dispatcher solo se preocupe por preguntar si hay empleados disponibles.

Las clases Util y UtilHTML

Estas son unas clases utilitarias muy importantes, pues en ellas se implementan operaciones que permiten separar la lógica de la aplicación de otras funciones que solo sirven para catalogar, hacer conteos, y formatear información que será presentada en la GUI. Con estas clases se busca una mejor legibilidad del código escrito buscando tener Clean Code en todo momento.

La clase FXMLController

Esta es la clase en la cual se junta, el modelo de datos, la interfaz gráfica y el control del programa, aquí se crea el objeto dispatcher y para el manejo del tiempo se crea un objeto de la clase Timeline el cual permite manejar de forma muy sencilla una línea de tiempo en la cual para la creación de llamadas, la asignación a los empleados y hacer el conteo de la información que se presenta en la GUI.

Timeline permite junto con los Empleados manejar de forma concurrente el manejo de las llamadas que llegan al Call Center.

Timeline implementa el método handle, el cual se invoca una vez se termina el tiempo de espera que se le asignó al objeto de Timeline. En la implementación de este programa, como se dijo anteriormente, se utilizó Java 8, el método handle desaparece pues se hace uso de la notación lambda que permite usar métodos anónimos.

En el método se tiene el control de la aplicación pues allí se invocan los métodos en los cuales se crean las llamadas, se asignan a los empleados y se saca la información que se presenta al usuario de la aplicación.

Test

Para verificar que el programa cumple con los requerimientos se escribieron dos métodos de prueba para ver qué sucede cuando llegan llamadas y no hay operarios disponibles. Los métodos son testRecibirLlamadasConRestriccion() y testRecibirLlamadasSinRestriccion(). En ambos métodos se crean 10 empleados () y se generan 20 llamadas con probabilidad de 100%.

En el método testRecibirLlamadasConRestriccion se crean las 20 llamadas de las cuales se atienden 10 que son asignadas a los empleados disponibles pero se pierden 10 llamadas que no pueden ser atendidas por nadie.

En el método testRecibirLlamadasSinRestriccion, por otro lado se crean también 20 llamadas, de las cuales se atienden 10, pero las llamadas no atendidas en el momento de su llegada son almacenadas en la cola de espera para ser procesadas en cuanto los empleados empiecen a desocuparse de las llamadas que tienen asignadas.