# Consigna

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.

# Requerimientos

* Debe existir una clase Dispatcher encargada de manejar las llamadas, y debe contener el método dispatchCall para que las asigne a los empleados disponibles.
* El método dispatchCall puede invocarse por varios hilos al mismo tiempo.
* La clase Dispatcher debe tener la capacidad de poder procesar 10 llamadas al mismo tiempo (de modo concurrente).
* Cada llamada puede durar un tiempo aleatorio entre 5 y 10 segundos.
* Debe tener un test unitario donde lleguen 10 llamadas

# Extras/Plus

* Dar alguna solución sobre qué pasa con una llamada cuando no hay ningún empleado libre.
* Dar alguna solución sobre qué pasa con una llamada cuando entran más de 10 llamadas concurrentes.
* Agregar los tests unitarios que se crean convenientes.
* Agregar documentación de código.

# Tener en cuenta

* El proyecto debe ser creado con Maven.
* De ser necesario, anexar un documento con la explicación del cómo y porqué resolvió los puntos extras, o comentarlo en las clases donde se encuentran sus tests unitarios respectivos.

# Solución

Codigo fuente: https://github.com/ElkinMantilla/AlmundoTest/

Se creó la clase Dispatcher que vendría a ser un deamon que recibe llamadas y se le asignan empleados, esta clase asigna asincrónicamente las llamadas en espera a los empleados disponibles quienes van a atender dichas llamadas. Cuando un empleado termina de atender una llamada se vuelve a cargar ese tipo de empleado a la cola de employeeAvailable.

Se tiene dos colas que almacenan en una las llamadas y en otra los empleados disponibles descriptas de la siguientes formas:

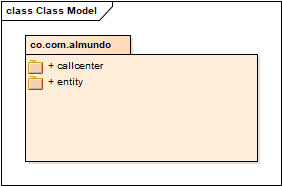
Para la llamadas se utiliza la cola java.util.concurrent.LinkedBlockingQueue.LinkedBlockingQueue que mantiene el hilo en espera cuando se quiere sacar un elemento (empleado) de una cola vacía.

Para los empleados disponibles se utiliza java.util.concurrent.PriorityBlockingQueue.PriorityBlockingQueue para manejar la prioridad en el que los empleados son tomados por el Dispatcher. La clase Employee implementa la interfaz Comparable, la cual permite ordenar a los empleados por la prioridad que tengan.

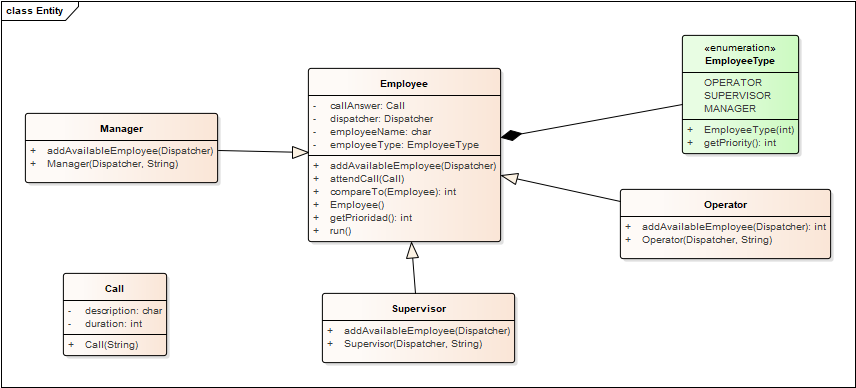
Para las llamadas, se creó una clase, la cual tiene un atributo de duración que cuando se crea el objeto se le asigna un tiempo radón entre 5 y 10 y es posible asignarle tiempo fijo después de creado el objeto con el método setDuration(int duration)

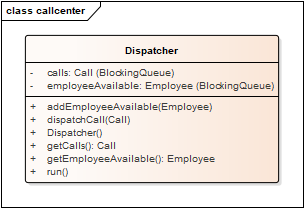
Con estas clases, se solucionan los Extra/plus.

1. Diagrama de paquete

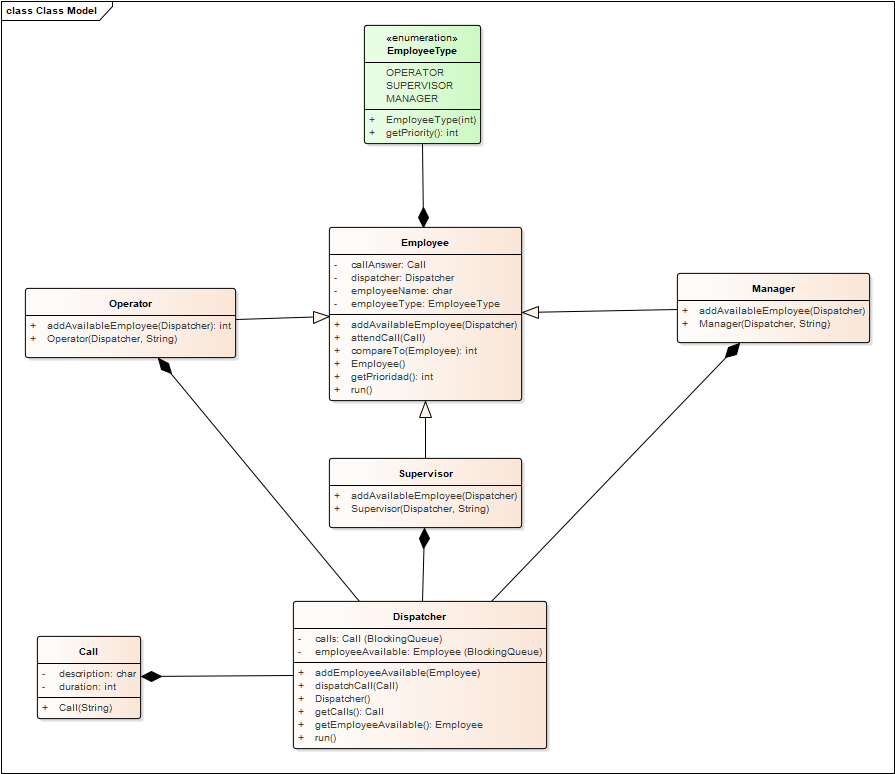


1. Diagrama de clases





1. Diagrama de relaciones



1. Diagrama de Secuencia

