

MavenCallCenter

DOCUMENTACIÓN DEL PROYECTO

Lina Marcela Malaver Gómez
INGENIERA DE SISTEMAS | COLOMBIA

Contenido

| | |
|--|---|
| ESTRUCTURA GENERAL DEL PROYECTO..... | 2 |
| DIAGRAMA DE CLASES | 3 |
| EXPLICACIÓN | 3 |
| Test Procesar10LlamadasAlMismoTiempo | 3 |
| Test EntraLlamadaYNoHayEmpleadoDisponible..... | 3 |
| Test EntranMasDe10LlamadasConcurrentes | 4 |

ESTRUCTURA GENERAL DEL PROYECTO





















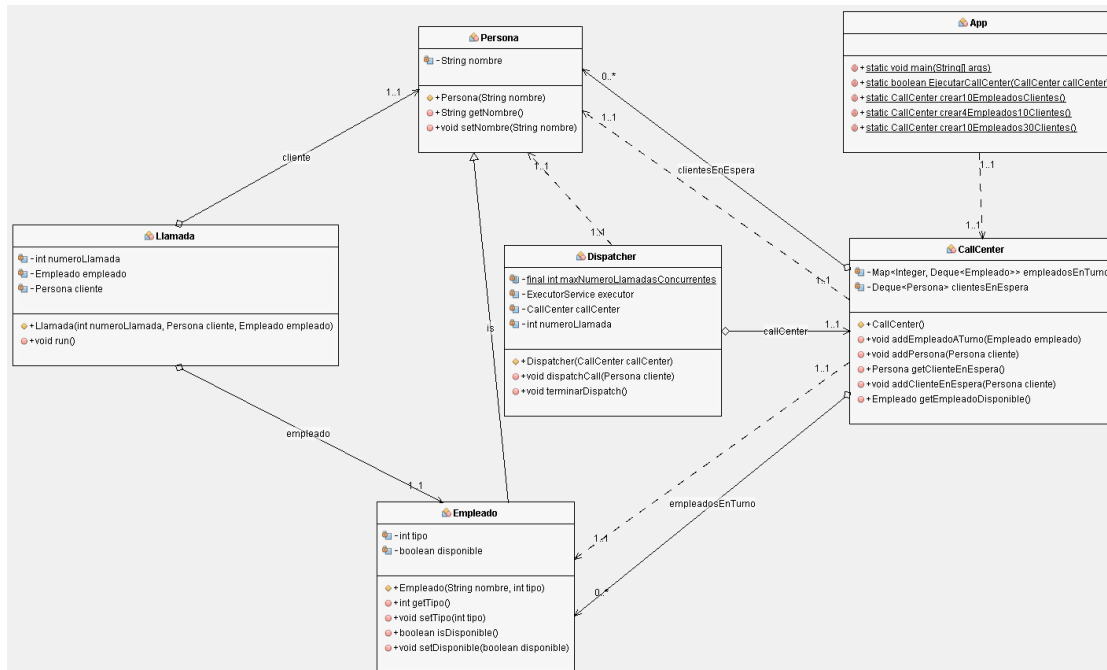
- ▼  MavenCallCenter
 - ▼  src/main/java
 - ▼  controller
 - >  Dispatcher.java
 - ▼  MavenCallCenter
 - >  App.java
 - ▼  model
 - >  CallCenter.java
 - >  Empleado.java
 - >  Llamada.java
 - >  Persona.java
 - ▼  src/test/java
 - ▼  MavenCallCenter
 - >  AppTest.java
 - >  JRE System Library [J2SE-1.5]
 - >  Maven Dependencies
 - >  JUnit 4
 - >  src
 - >  target
 - >  pom.xml

DIAGRAMA DE CLASES



EXPLICACIÓN

Test Procesar10LlamadasAlMismoTiempo

Esta es la solución al requerimiento principal:

1. Se crean 10 empleados, 10 clientes y todos se añaden al Modelo CallCenter

```
CallCenter callCenter = App.crear10EmpleadosClientes();
```

2. Se llama al método `App.EjecutarCallCenter(callCenter)` que hace las llamadas por medio de `dispatchCall` el cual a través de `Runnable` y `executor` permite el procesamiento de 10 llamadas al mismo tiempo. Al final devuelve `true` si no ocurrió ningún error o `false` en caso de haber un problema.

```
Assert.assertTrue(App.EjecutarCallCenter(callCenter));
```

Test EntraLlamadaYNoHayEmpleadoDisponible

Esta es la solución propuesta para qué pasa con una llamada cuando no hay ningún empleado libre. De manera general esto se soluciona en `callCenter.addClienteEnEspera(cliente);`:

```
public void dispatchCall(Persona cliente) {
    Empleado empleadoDisponible =
callCenter.getEmpleadoDisponible();
    if (empleadoDisponible != null) {
        //Al ejecutar la llamada se imprime el numeroLlamada el
cual aumenta si se asigna de lo contrario no aumenta
        Runnable llamada = new Llamada(numeroLlamada, cliente,
empleadoDisponible);
```

```

        executor.execute(llamada);
        numeroLlamada++;
    }

    else {
        //No hay empleados disponibles para tomar la llamada, por
        tanto se añade a clientes en espera
        callCenter.addClienteEnEspera(cliente);
    }
}

```

En el cual se añade a clientes en espera (.offerFirst lo añade a la cabeza) y no se sigue con otra llamada hasta que esa es atendida:

```

/*Metodo usado cuando no hay empleados disponibles para tomar la llamada,
 * por tanto se añade a clientes en espera (.offerFirst lo añade a la
 * cabeza) */
public void addClienteEnEspera(Persona cliente) {
    clientesEnEspera.offerFirst(cliente);
}

```

1. Se crean 4 empleados, 10 clientes y todos se añaden al Modelo CallCenter

```
CallCenter callCenter = App.crear4Empleados10Clientes();
```

2. Se llama al método App.EjecutarCallCenter(callCenter) que hace las llamadas por medio de dispatchCall el cual a través de Runnable y executor permite el procesamiento de 10 llamadas al mismo tiempo pero que al no haber empleados disponibles para atenderlas a tiempo se añade a clientes en espera hasta que hay un empleado que pueda contestar la llamada. Al final devuelve true si no ocurrió ningún error o false en caso de haber un problema.

Test EntranMasDe10LlamadasConcurrentes

Esta es la solución propuesta para qué pasa con una llamada cuando entran más de 10 llamadas concurrentes. De manera general se sigue el comportamiento explicado en el test anterior, pero se tiene en consideración que el máximo número de llamadas al tiempo permitido es 10 por lo que no crea un nuevo Thread sino que espera a que un thread este disponible y lo asigna nuevamente si hay más llamadas por atender y deja de hacerlo cuando todas las llamadas son contestadas.

Para efectos del test:

1. Se crean 10 empleados, 30 clientes y todos se añaden al Modelo CallCenter

```
CallCenter callCenter = App.crear10Empleados30Clientes();
```

2. Se llama al método App.EjecutarCallCenter(callCenter) que hace las llamadas por medio de dispatchCall el cual a través de Runnable y executor permite el procesamiento de 10 llamadas al mismo tiempo pero que al no haber empleados disponibles para atenderlas a tiempo se añade a clientes en espera hasta que hay un empleado que pueda contestar la llamada. A la vez puesto que se supera el numero de Thread permitidos entonces se reutilizan los existentes apenas son liberados. Al final devuelve true si no ocurrió ningún error o false en caso de haber un problema.