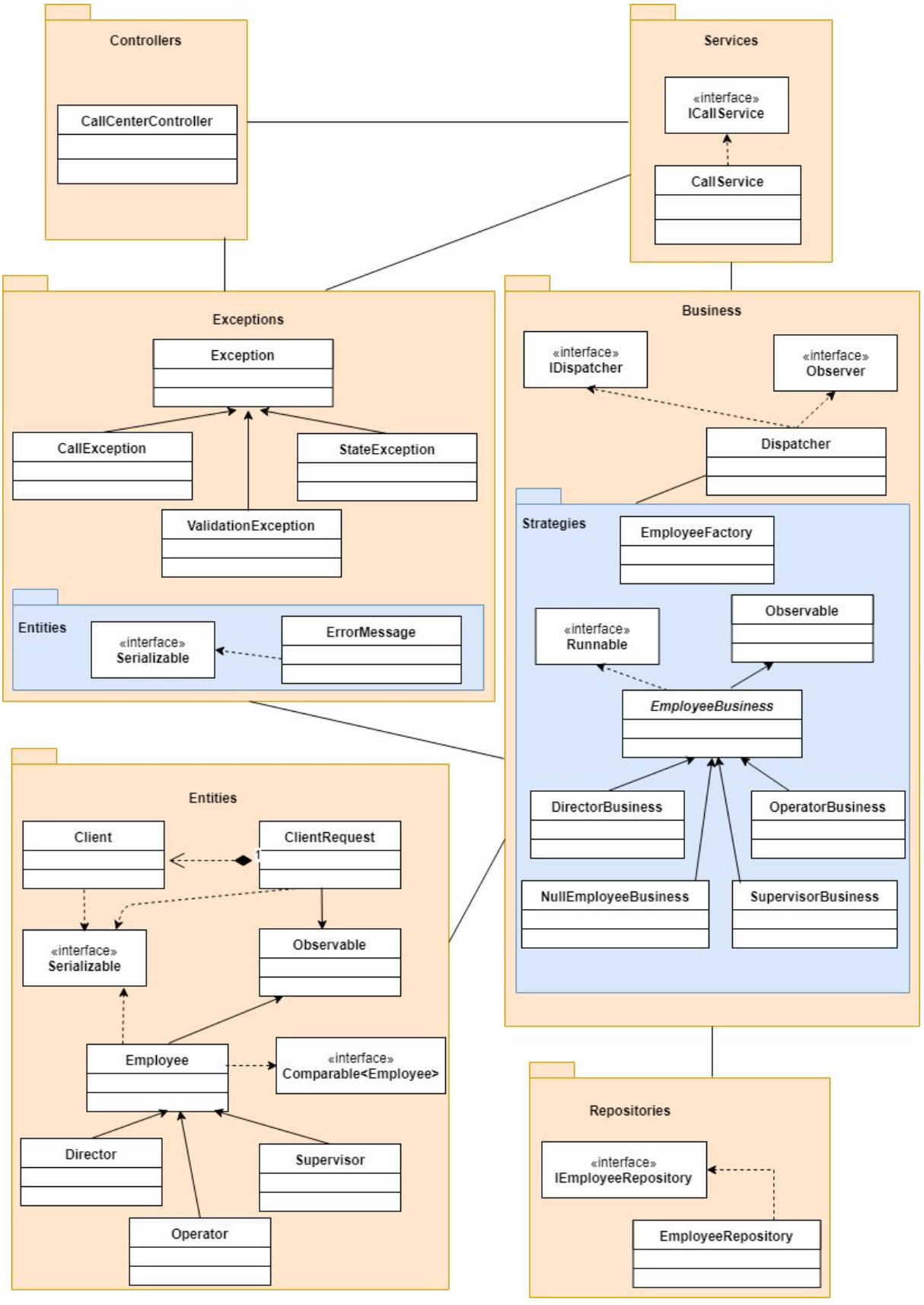
Informe desarrollo

Para poder resolver el problema planteado se creó la siguiente estructura:



Cada llamada que ingresa, es atendida por el método dispatchCall, de la clase Dispatcher. Si no existen empleados disponibles o se llegó al máximo de llamadas para ser procesadas en simultáneo, se encolan a la espera de poder ser atendidas.

Se creó un clase base del tipo Employee, donde se tienen los atributos compartidos por todos los empleados. De ésta extienden los diferentes tipos de empleados que se tiene: Supervisor, Operator y Director. Si bien no se le agregaron atributos puntuales a cada uno, se deja la estructura para poder hacerlo a futuro.

Un empleado es de tipo Observable, implementando así el patrón de diseño observer. De esta manera, el empleado notifica al Dispatcher (observer) cuando su estado ha sido modificado. El Dispatcher cuenta con dos estructuras de datos, una para los empleados ocupados (Set<Employee>) y otra para los disponibles (SortedSet<Employee>). La implementación del patrón de diseño observer permite mantener ambas estructuras siempre actualizadas.
Se usan dos estructuras de datos para ser más eficientes al momento de seleccionar un empleado para atender una llamada. Por este mismo motivo, Employee extiende de Comparable, para que a medida que se vayan guardando en un TreeSet se vayan ordenando por tipo de empleado (TypeEmployee), para quedar primero los operadores, después los supervisores y por último los directores.

Por otro lado, para la atención de las llamadas se creó la clase EmployeeBusiness, la cual implementa la interfaz Runnable, debido a que se van a ejecutar en hilos separados, permitiendo la concurrencia de estas. Esta clase es del tipo observable, con la idea de que se informe cuando el hilo terminó de ejecutarse, para que en caso de que hayan llamadas en espera de ser atendidas, comience el proceso de atención.
A su vez, usa el patrón template para armar la estructura base para la atención a la llamada, y por medio del patrón strategy, se ejecuta la estrategia de atención que adopta cada tipo de empleado.
Se usó el patrón de diseño Null Object para crear una estrategia por defecto.

Se han mockeado 10 empleados, en estado disponible, a efectos de realizar pruebas.

Se agregó un rest controller que expone un servicio para ingresar una llamada:

- Url: http://localhost:8080/callcenter/
- Tipo de request: POST
- Header:
    - Client ---> seria el codigo de cliente.
- Body:
  {
  "dni": 12555666,
  "name": "Eliana"
  }

- Response:
    - Si no hubo errores: HttpStatus: 201
    - Si hubo error:
      HttpStatus: 400

```
{
    "httpStatus": 400,
    "message": "El cliente debe de informar su nombre.",
    "code": 400
}
```