UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

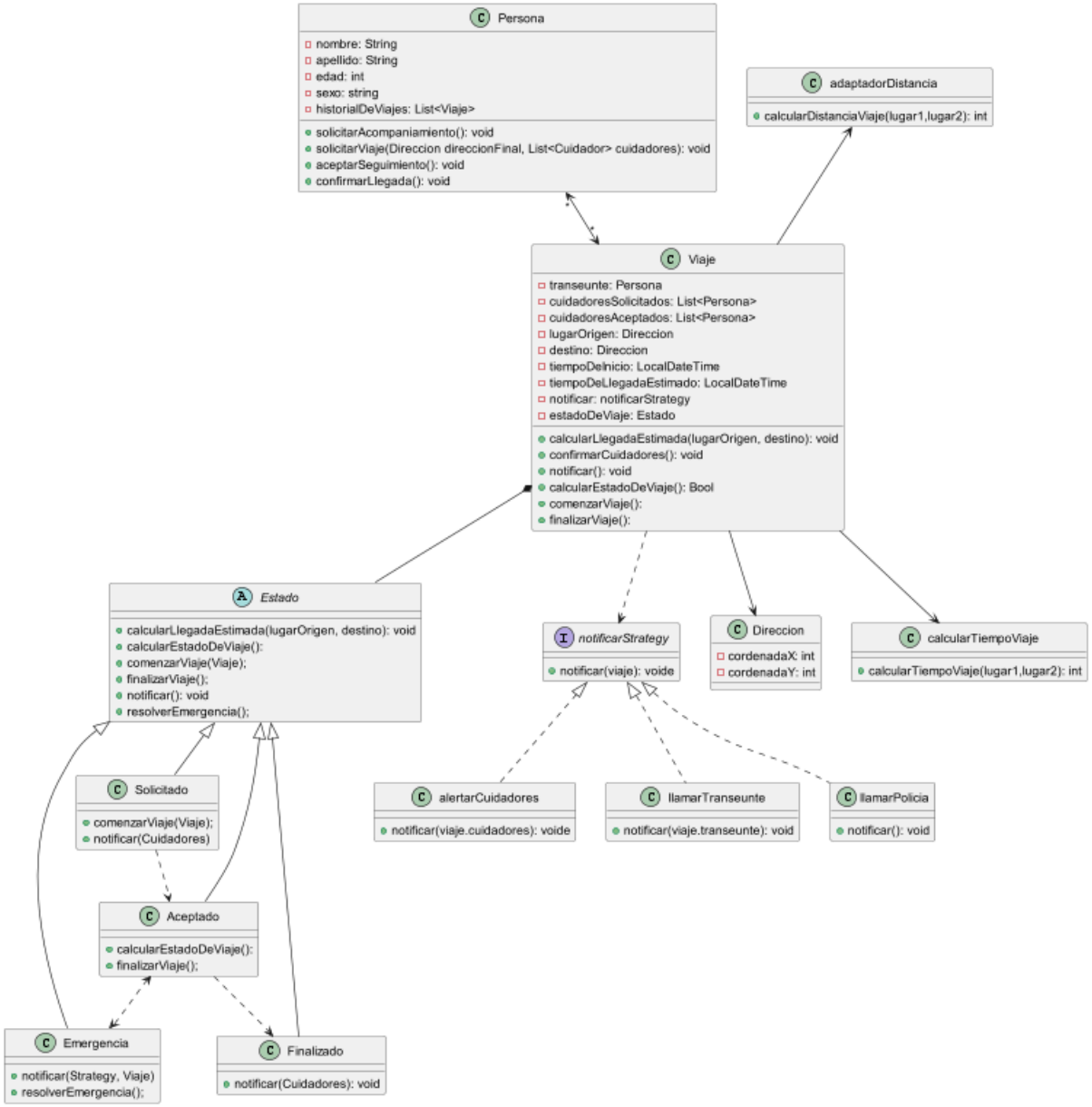
Diseño de Sistemas

Cuidandonos

- Curso : K3152 (Miércoles a la noche)
- Integrantes: Tomás Schwartz, Facundo Arakaki.

Punto 2 - Modelo de Dominio

1. Modelar el dominio presentado utilizando el paradigma orientado a objetos, comunicando su solución mediante un diagrama de clases. Si utiliza patrones de diseño, indíquelos y justifique su uso. NOTA: Puede ayudarse para comunicar, además, con código, pseudo-código, prosa u otros diagramas (diagrama de secuencia, de estados, entre otros).



Cuando un transeunte comienza el viaje, elige sus acompañantes y el método de notificación. El viaje se crea en estado “solicitado”, y se moverá a “Aceptado” cuando al menos un cuidador acepte el viaje. Una vez en aceptado, utilizando el método calcularEstadoDeViaje(), se comparará el tiempo actual con el tiempo estimado, y si

detecta algun imprevistó, notificará según haya definido el transeunte. Una vez se resuelva la emergencia, vuelve a aceptado. Cuando llegue al destino, y el transeunte confirme que llegó, el estado pasará a finalizado y notificará a todos los cuidadores que aceptaron seguir el viaje.

Para este trabajo, utilizamos 3 patrones:

Primero, al utilizar la API vamos a tener que adaptar nuestro programa a los requerimientos de la API, el cual va a recibir dos distancias y las va a adaptar para que la API nos devuelva entonces un int, es decir, la distancia entre los dos puntos.

Segundo, detectamos que el notificar emergencia dependía de la estrategia definida por el transeunte, pero que no variaba durante el viaje, sino que era definida al principio del viaje. Por eso, dependiendo del valor inicial, actuaría de forma distinta el notificar. Para eso, utilizamos el patron strategy, el cual dependiendo de la estrategia definida por el transeunte, podría entonces cada estrategia actuar de forma distinta.

Tercero, el estado del viaje iba a tener comportamientos distintos según cómo esté, pero además, debería ir cambiando a lo largo del trayecto. Por ello utilizamos un patron state, el cual nos permite ir cambiando el estado del viaje, y con ello, definir el comportamiento interno del mismo.

2. Ahora un transeúnte también podrá escoger un destino con varias paradas; esto es: Posición actual -> primer destino -> segundo destino -> ... -> destino final. Para esto, se deberá especificar la dirección exacta de cada destino y el orden en el que se recorrerán. Además, el usuario deberá especificar si se detendrá N minutos en cada parada, o si irá avisando punto a punto su estado de “salud” (si llegó bien). Si se especifica que se va a detener en cada parada, entonces el sistema deberá ir calculando las demoras aproximadas por secciones (demora de A->B, demora B->C, etc.); caso contrario, se deberá hacer un cálculo aproximado total. Extienda su solución para que soporte este nuevo requerimiento. Además, muestre mediante código o pseudocódigo cómo implementaría el cálculo de demora aproximado.

Para abordar la nueva necesidad, añadimos un atributo denominado formaDeViaje, el cual será un Enum con tres valores posibles: si es viaje directo (es decir, no hay

paradas), si se detendrá en cada parada (en este caso, también se guardará en otro atributo los N minutos en cada parada), o si confirmará en cada parada si llegó bien.

Aprovechando el paradigma de objetos, delegaremos la responsabilidad de "comenzarViaje" al valor que tomará dicho enum. Los valores que este podrá tomar son:
- VIAJE_DIRECTO: esto será el default. Es decir, es lo mismo que el punto anterior.
- VIAJE_CON_DEMORA: donde el usuario deberá ingresar la demora que realizará en cada parada y el sistema sumará los tiempos de todas las secciones del viaje para calcular la duración total.
- VIAJE_NOTIFICANDO_SALUD: Implementará un algoritmo que se verá de la siguiente manera:

```
comenzarViaje(){
        confirmarCuidadores();

        for(Int i = 0; i<length(viaje), i++) {
                calcularLlegadaEstimada(destino[i], destino[i+1]);
                self.estado.comenzarViaje(viaje);
                setter_estado(aceptado);
}
}
```

En esta opción, confirmarCuidadores desempeña el mismo papel que en el punto 1: verifica que al menos haya un cuidador asignado para supervisar el viaje del transeúnte. Luego, utilizamos un bucle for para iterar sobre la cantidad de paradas solicitadas por el transeúnte. En cada iteración, se calcula el tiempo estimado para llegar de una parada a la siguiente, se inicia cada sub-viaje y se establece el estado aceptado. Este estado controla el estado de "salud" del transeúnte, es decir, si ha completado ese sub-viaje o si se encuentra en una situación de emergencia.