Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2022

Alumno:	SAIAGO ROJAS, Brayan Joaquin
Número de padrón:	104967
Email:	bsaiago@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un servicio de telefonía en Pharo, utilizando los conceptos del paradigma de POO vistos hasta ahora en el curso.

2. Supuestos

2.1. Restriccion a cantidades negativas

No podrán asignarse cantidades negativas (> 0) a las duraciones de las llamadas, a la tarifa de las llamadas internacionales, al abono mensual y al costo de la llamadas, puesto que no tiene ningún tipo de sentido lógico que dichos campos contengan esos valores.

2.2. 2 tipos de descuentos (Descuento nulo y Descuento Por duracion)

Todas las llamadas registradas (Nacionales o Internacionales) pueden tener uno de los 2 descuentos posibles. El descuento nulo que hace referencia a un descuento sin valor cuando la duración de la llamada no supera los 30 Min , es decir, es equivalente a un descuento inválido, mientras que por el otro lado, El descuento por duración es un descuento que equivale a un 10 por ciento menos en el precio de la llamada, y se aplica cuando la duración de la llamada es mayor o igual 30 minutos de duración.

2.3. Los Datos entrantes deben ser del tipo de dato esperado

Para cualquier llamada (Nacional o Internacional) los datos que se ingresan pueden ser inválidos (no estar dentro del rango pedido, y no pasa nada porque las excepciones los controlan) pero no pueden ser de distinto tipo de dato con lo esperado, es decir, que si por ejemplo se intenta registrar una llamada con duración de tipo de dato string, el comportamiento del programa es indefinido, puesto que la duración debe ser un número en el caso del ejemplo expuesto anteriormente.

2.4. El país de destino de la llamada internacional no puede ser vacío

Esto hace referencia a que cuando se registra una llamada internacional, el país de destino a donde se hizo la llamada no puede vacío, puesto que a la empresa le interesa saber el país de dicha llamada para poder establecer un precio justo para las tarifas internacionales, o simplemente para establecer promociones a través de las estadisticas obtenidas en base a la información de todas las llamadas.

2.5. Las llamadas nacionales hacen refencia a el territorio Argentino

Cuando se registra una llamada nacional con su duración y horario. Por defecto el país de destino será Argentina, el cual no puede ser cambiado de ninguna manera posible, es decir, que todas las llamadas son dentro del territorio Argentino.

3. Detalles de implementación

3.1. Pilares del POO

Los principales pilares del paradigma POO son: Herencia, Polimorfismo, Abstracción, Encapsulamiento y Delegación.

El uso de herencia se manifiesta en RegistroLlamada con la finalidad de reutilizar código, puesto que ambos registros (RegistroLlamadaNacional y RegistroLlamadaInternacional) funcionan igual con una única diferencia en su comportamiento, la ventaja en el uso de la herencia se centra principalmente en la reutilización de código entre clases y la usamos en este caso porque se cumple la relación 'es un', una desventaja es por ejemplo: al modificar una clase padre que afecta el funcionamiento de las subclases.

El uso de polimorfismo se observa en esta misma clase, cuando se envia el mensaje costoLlamada, y cada clase hija realiza un comportamiento distinto ante ese mensaje, la principal ventaja de resolver los problemas a través de soluciones polimorficas, es que al querer ampliar la implementacion se vuelve muchisimo más sencillo, ya que solo hay que agregar la nueva clase, agregar la firma del método que va a tener ese nueva clase en la clase padre (metodo abstracto) y en la clase hija, y por último implementar dicho método en la clase hija que se desea agregar. La desventaja de esto es que hay una importante limitación: el tipo de la referencia (clase abstracta) limita los métodos que se pueden utilizar y los atributos a los que se pueden acceder.

Por otra parte la delegación puede verse en mucho lugares, como por ejemplo en RegistroLlamada cuando le transfiere responsabilidades (delega una responsabilidad para que la resuelva) a DescuentoNulo o por duración según sea el caso, así también con TeleAlgo derivando ciertas responsabilidades a RegistroLlamada. De esta manera cuando se hacen dichas delegaciones, podemos observar que también entra en juego el concepto de encapsulamiento, ya que nunca se expone el estado del objeto, debido a que la comunicación es a través de mensajes, nunca se pide el estado de un objeto para trabajar en base a eso, pues eso no nos importa, ya que eso rompe el encapsulamiento (tell don't ask). ventajas de esta caracteristica es que a la hora de modificar las implementaciones solo hay que modificar la parte en la que se busca cambiar el comportamiento, al no haber código repartido por todos lados esto se vuelve una tarea mucho mas sencilla. La desventaja es que con algunos problemas se suele tornar bastante complicada la solución a través de delegación, y lo que conviene es usar otro pilar POO, como por ejemplo Herencia o polimorfismo.

A continuacion se mostrarán algunos fragmentos del código que se utilizo para resolver las especificaciones pedidas por el problema en cuestión a resolver (TP 1). En dichos ejemplos se podrán observar como se manifiestan los conceptos de POO mencionados anteriormente y conceptos dados a lo largo de las clases relacionados a las lecturas obligatorias (Tell Don't Ask, Shy Code, Getter Eradicator, "The Art of Enbugging', entre otras).

Ejemplo 1: Se desea calcular el gasto final, entonces lo que se hace es que el registro de llamadas polimórfico envía el mensaje costoLlamada, el cual van a entender todos los hijos. Para agregar vemos que cuando se envia ese mensaje, no se sabe cual hijo es el que va a responder, lo único que se sabe es que un hijo seguro responda, y aca es donde entra en juego el polimorfismo, y la abstracción (la que permite reedefinir un mismo mensaje en cada hija).

```
calcularGastoFinal
| gastoFinal |
gastoFinal := abonoMensual.
registroLlamadas do: [ :registro | gastoFinal := gastoFinal + registro costoLlamada. ].
^ gastoFinal.
```

En este ejemplo 1 podemos observar:

- Pocos puntos de acoplamiento.
- Se cumplen conceptos de la lectura "The Art of Enbugging' (Código timido que no dice más de lo que se debe, que no toma decisiones en base a un estado, si no que responde ante un mensaje) [Tell Don't Ask].
- Se respeta el ocultamiento de la información (No se exponen atributos de los objetos a los que se les delega una responsabilidad).
- Se respeta el encapsulamiento.
- Se delegan las responsabilidades a los objetos que están encargados o que tienen los comportamientos necesarios para responder de forma esperada ante un mensaje que reciban (TeleAlgo no sabe como calcular el gasto, entonces el objeto registro entiende el mensaje costoLlamada, y la hija que corresponda respondera ante el mismo).
- Correcto uso del polimorfismo con herencia.
- Abstracción para poder reedefinir un método en distintas clases hijas.

Ejemplo 2: Se desea calcular el precio por minuto de una llamada nacional sin incluir el descuento, entonces entran en juego 2 fragmentos de código, el primero hace referencia a la llamada nacional en horario hábil y el otro a la llamada nacional en horario no hábil.

```
precioFinal: unaDuracion
( unaDuracion < 0 ) ifTrue: [ LaDuracionDeLaLlamadaEsNegativaError new signal. ].
^ ( unaDuracion * 0.2 ).
```

```
precioFinal: unaDuracion
( unaDuracion < 0 ) ifTrue: [ LaDuracionDeLaLlamadaEsNegativaError new signal. ].
^ ( unaDuracion * 0.1 ).
```

En este ejemplo 2 podemos observar:

- Ambos métodos son iguales a nivel firma (mismo mensaje), pero se comportan de manera distinta, es decir polimorfismo, que al mismo tiempo sería sin herencia. Además se catalago como polimorfismo, porque: I) los comportamientos se desarrollan dentro de esos objetos. II) Responden ante el mismo mensaje de distinta forma (distinto comportamiento). III) En registro llamada nacional cuando se envía ese mensaje no sabe cual es el objeto que va a responder, lo único que se sabe es que puede ser alguno de esos 2 seguro.
- Pocos puntos de acoplamiento.
- Se cumplen conceptos de la lectura "The Art of Enbugging' (Código timido que no dice más de lo que se debe, que no toma decisiones en base a un estado, si no que responde ante un mensaje) [Tell Don't Ask].
- Se respeta el ocultamiento de la información (No se exponen atributos de los objetos a los que se les delega una responsabilidad).
- Se respeta el encapsulamiento.
- Se hace uso de polimorfismo sin herencia.
- Se cumple la ley de Demeter.

Ejemplo 3: Se desea calcular el precio total de una llamada internacional, entonces se delega la responsabilidad de calcular el precio final con un descuento al objeto descuento.

```
costoLlamada
| costo |
costo := duracion * tarifa.
^ ( descuento calcularDescuentoFinal: costo ).
```

En este ejemplo 3 podemos observar:

- la llamada no sabe como calcular el precio final incluyendo el descuento, entonces lo que hace es delegarle esa responsabilidad a descuento, y descuento es el que se va a encargar de esa tarea.
- Pocos puntos de acoplamiento.
- Se cumplen conceptos de la lectura "The Art of Enbugging' (Código timido que no dice más de lo que se debe, que no toma decisiones en base a un estado, si no que responde ante un mensaje) [Tell Don't Ask].
- Se respeta el ocultamiento de la información (No se exponen atributos de los objetos a los que se les delega una responsabilidad).
- Se respeta el encapsulamiento.
- Se hace uso de polimorfismo sin herencia (descuento no sabe a quien le envia el mensaje, solo sabe que el mensaje ira para algún objeto que entienda ese mensaje).

3.2. La clase RegistroLlamada y sus hijos

Podía observarse en el diagrama de clases como RegistroLlamadaNacional y RegistroLlamadaInternacional heredaban de la clase abstracta RegistroLlamada, con esto además de buscar la solución polimórfica al problema que se generaba al devolver el costo de la llamada (ya que habían llamadas nacionales y llamadas internacionales) a las cuales calcularles el costo para poder devolver el costo final de dicha llamada, se aplica herencia con la finalidad de reaprovechar el código. Esto es posible gracias a que se cumple la condición 'es un'. Por otro lado, una de las cosas que podemos destacar, es que puede observarse la delegación que hace el hijo RegistroLlamadaNacional, al pasarle los problemas a otra clase.

4. Excepciones

4.1. ElAbonoMensualEsNegativoError

Esta excepción es lanzada, cuando en el proceso de inicialización de un abono mensual en la clase TeleAlgo, se detecta que el abono mensual que se quiere asignar a dicha clase es negativo. De esta manera, la excepción evita que se rompa la lógica de la implementación, ya que no está bien que sea la empresa la que le deba pagar el abono mensual al cliente, cuando la empresa es la que ofrece el servicio, que dicho cliente puede tomar o no, según lo que le convenga.
Ejemplo:

```
| teleAlgo |
teleAlgo := TeleAlgo conAbonoMensual: -50.
```

4.2. LaTarifaInternacionalEsNegativaError

Cuando en el proceso de inicialización de una llamada internacional en las clases TeleAlgo o RegistroLlamada, se detecta que la tarifa internacional que se quiere asignar a dicha llamada es negativa. De esta manera, la excepción evita que se rompa la lógica de la implementación, puesto que no tiene sentido, como lo mencionamos anteriormente, que sea la empresa la que le deba pagar a el cliente.
Ejemplo:

```
| teleAlgo tarifaInternacional |
tarifaInternacional := -3.
teleAlgo := TeleAlgo new.
teleAlgo aplicarTarifaPorMinutoALlamadaInternacional: tarifaInternacional.
```

4.3. ElHorarioNoEstaEntre1Y24Error

Como su nombre lo indica, se lanza está excepción cuando en el proceso de inicialización de una llamada nacional o internacional en las clases TeleAlgo o RegistroLlamada, se detecta que el horario que se quiere asignar a dicha llamada no está dentro del rango comprendido por 1 y 24 inclusive. De esta manera, la excepción evita que se rompa la lógica de la implementación, puesto que no tiene sentido registrar una llamada en un horario que se escapa de la realidad, sabiendo que un día no tiene más de 24 horas, ni puede tener horas negativas.
Ejemplo:

```
| teleAlgo horario |
teleAlgo := TeleAlgo new.
horario := 25.
teleAlgo registrarLlamadaNacionalConDuracion: 100 HechaEnElHorario: horario.
```

4.4. NoHayLlamadasRegistradasError

Cuando se intenta buscar la llamada nacional o internacional registrada más costosa, Se lanza esta excepción si no hay ninguna llamada registrada en la clase TeleAlgo. De esta forma evitamos romper la lógica de la implementación, puesto que si no hay ninguna llamada hecha por el cliente, tiene sentido que la llamada más costosa no exista (que no sea ninguna).
Ejemplo:

```
| teleAlgo |
teleAlgo := TeleAlgo new.
teleAlgo consultarPorLlamadaEfectuadaMasCostosa.
```

4.5. ElPaisEsVacioError

Esta Excepción se arroja cuando en el proceso de inicialización de una llamada internacional en las clases TeleAlgo o RegistroLlamadaInternacional, nos encontramos que el nombre del país que se intenta registrar en la llamada, es vacío. En Consecuencia, nos evitamos tener este problema que rompe la lógica de la implementación, debido a que queremos que la empresa sepa hacia que país se realizo la llamada, o que mínimamente ese campo contenga alguna información y no esté vacío. Sumado a eso también tenemos que usar esa información del país para otros requerimientos del problema, con lo cual si viene vacío, ese error va a afectar a la hora de obtener la solución del problema final.
Ejemplo:

```
| teleAlgo |
teleAlgo :=  TeleAlgo new.
teleAlgo registrarLlamadaInternacionalCon: 31 HechaEnElHorario: 20 Hacia: ''.
```

4.6. LaDuracionDeLaLlamadaEsNegativaError

Esta excepción es lanzada, cuando en el proceso de inicialización de una llamada nacional o internacional en las clases TeleAlgo o RegistroLlamada, se detecta que la duración que se quiere asignar a dicha llamada es negativa. También puede ser lanzada cuando se intenta calcular el precio por minuto en las clases PrecioPorMinutoHabil ó PrecioPorMinutoNoHabil de una llamada nacional, y dicha llamada tiene una duración negativa. De esta manera, la excepción evita que se rompa la lógica de la implementación, puesto que no tiene sentido registrar una llamada con una duración negativa.
Ejemplo:

```
| teleAlgo |
teleAlgo := TeleAlgo new.
teleAlgo registrarLlamadaInternacionalCon: -60 HechaEnElHorario: 20 Hacia: 'Chile'.
```

4.7. CostoNegativoError

Se lanza esta excepción cuando se intenta calcular el descuento para una llamada nacional o internacional, y el valor del costo de dicha llamada sin descuento es negativo. De esta manera lo que hace esta excepción es evitar que se rompa la lógica de implementación, esto es debido a que no tiene lógica que el cliente tenga que pagar una llamada con el descuento aplicado, y que la misma tenga un costo negativo, ya que de esta manera sería la empresa la que le debe pagar a el cliente, y eso no es lo que queremos que pase.
Ejemplo:

```
| precioSinDescuento descuento |
precioSinDescuento := -21.7.
descuento := DescuentoNulo new.
descuento calcularDescuentoFinal: precioSinDescuento.
```

5. Diagramas de clases

5.1. Vista General

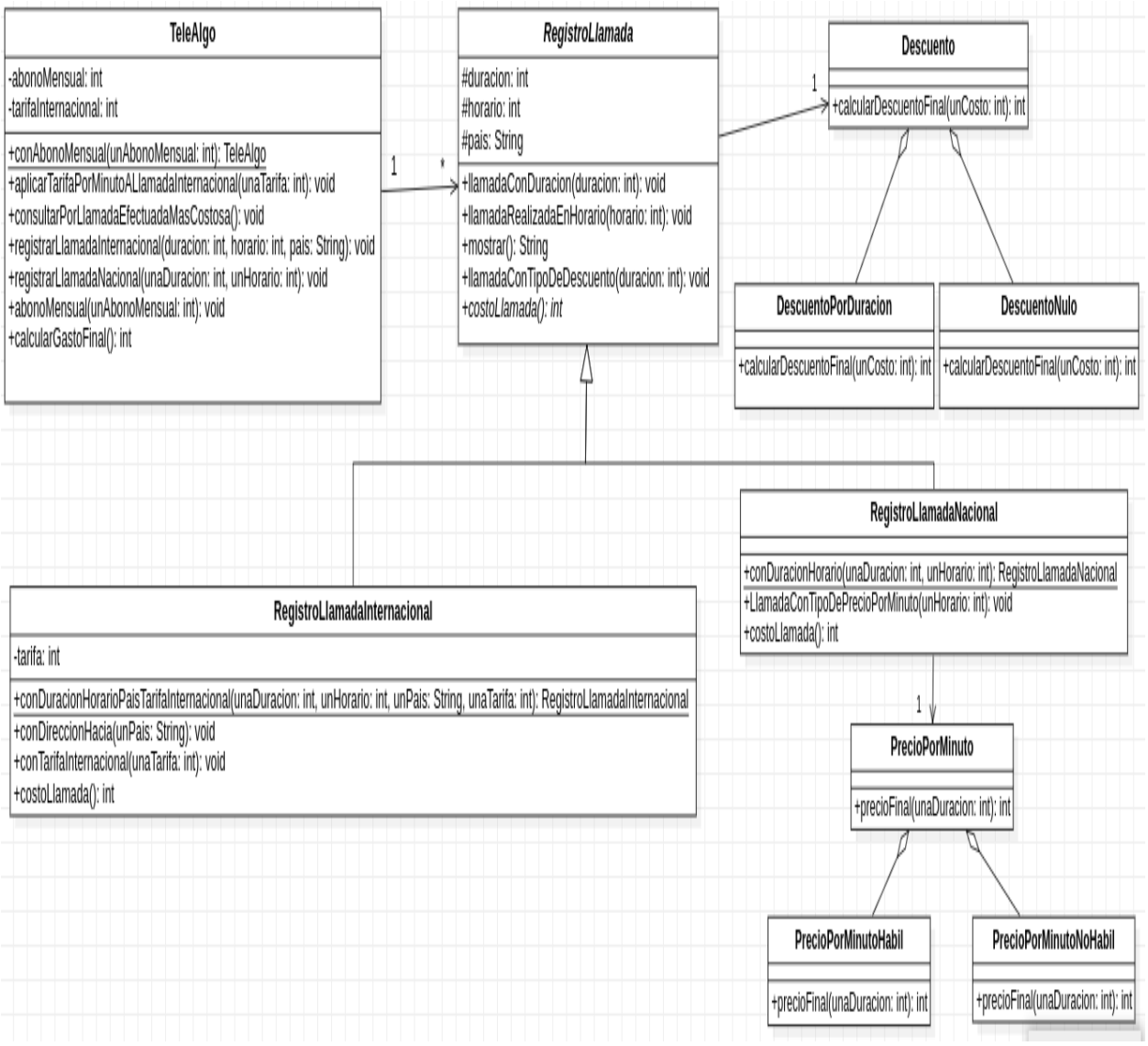

Figura 1: Diagrama de clases (Interacción entre las clases para obtener la solución del problema).

Esta figura tiene finalidad de mostrar la relación que existe entre las clases principales y como interactúan para llegar a la solución del problema.

5.2. Vista Con Las Excepciones

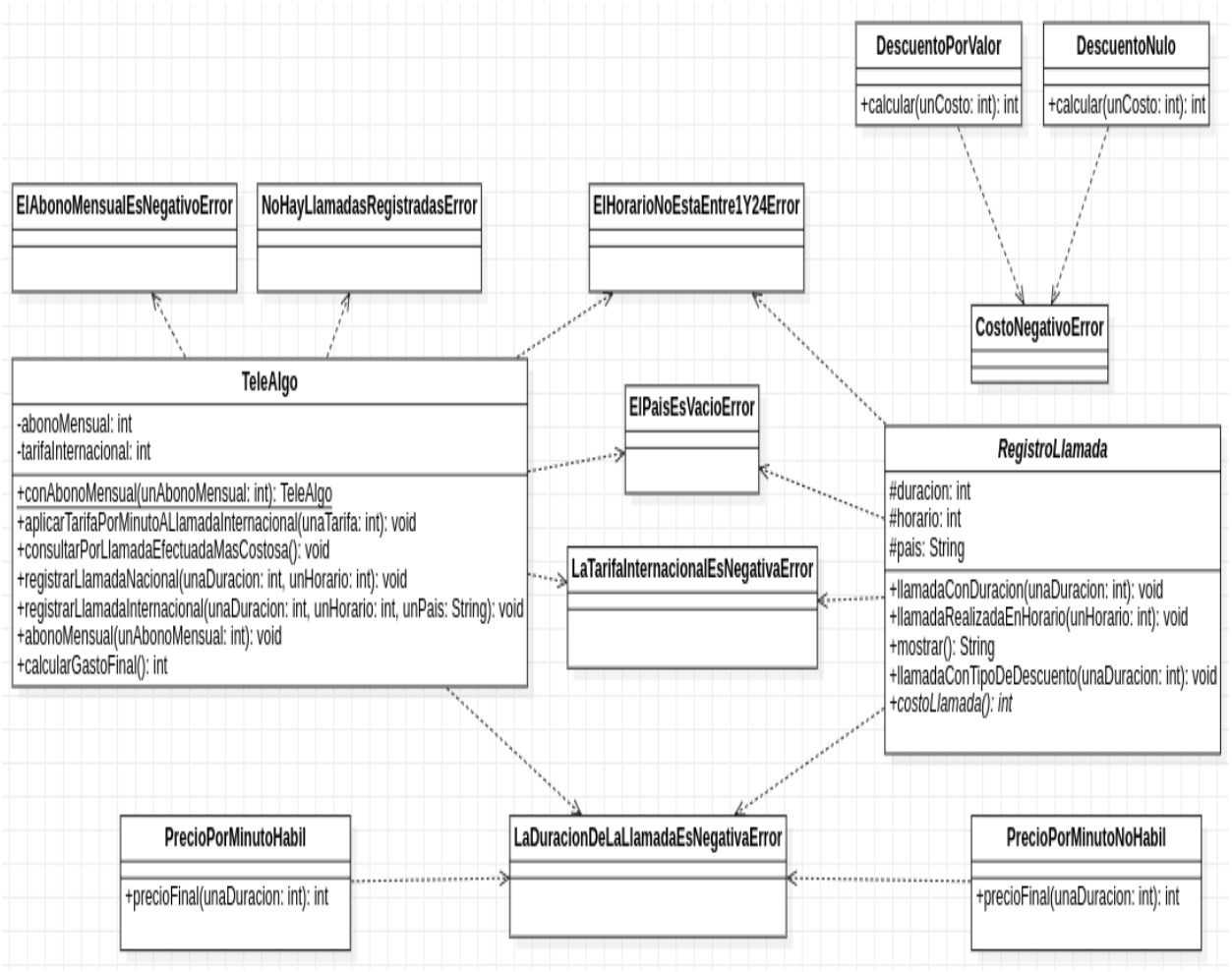

Figura 2: Diagrama de clases (Excepciones que dependen de cada clase).

Esta figura tiene como finalidad mostrar las relaciones de dependencia que tiene cada excepción con las clases mostradas en el diagrama anterior. En adición con lo dicho anteriormente, nos permite tener más detalles de como actúa nuestro código ante distintas situaciones que pueden romper la lógica de la implementación del diseño.

Para destacar, se puede observar que en está misma figura no aparece el atributo descuento de la clase RegistroLlamada, puesto que el mismo tenia que ser representado junto con 2 clases más (DescuentoNulo y DescuentoPorDuracion), las cuales no aportaban información para el proposito que tiene este modelado, el cual es mostrar la relación de dependencia que tienen las excepciones con sus respectivas clases. Para obtener más información acerca del atributo mencionado anteriormente, se recomienda ver la figura 1 del diagrama de clases, el cual se encuentra en la página anterior.

6. Diagramas de secuencia

6.1. Diagrama de secuencia 01 - Se Registra Una Llamada Nacional Y Se Calcula El Gasto Final

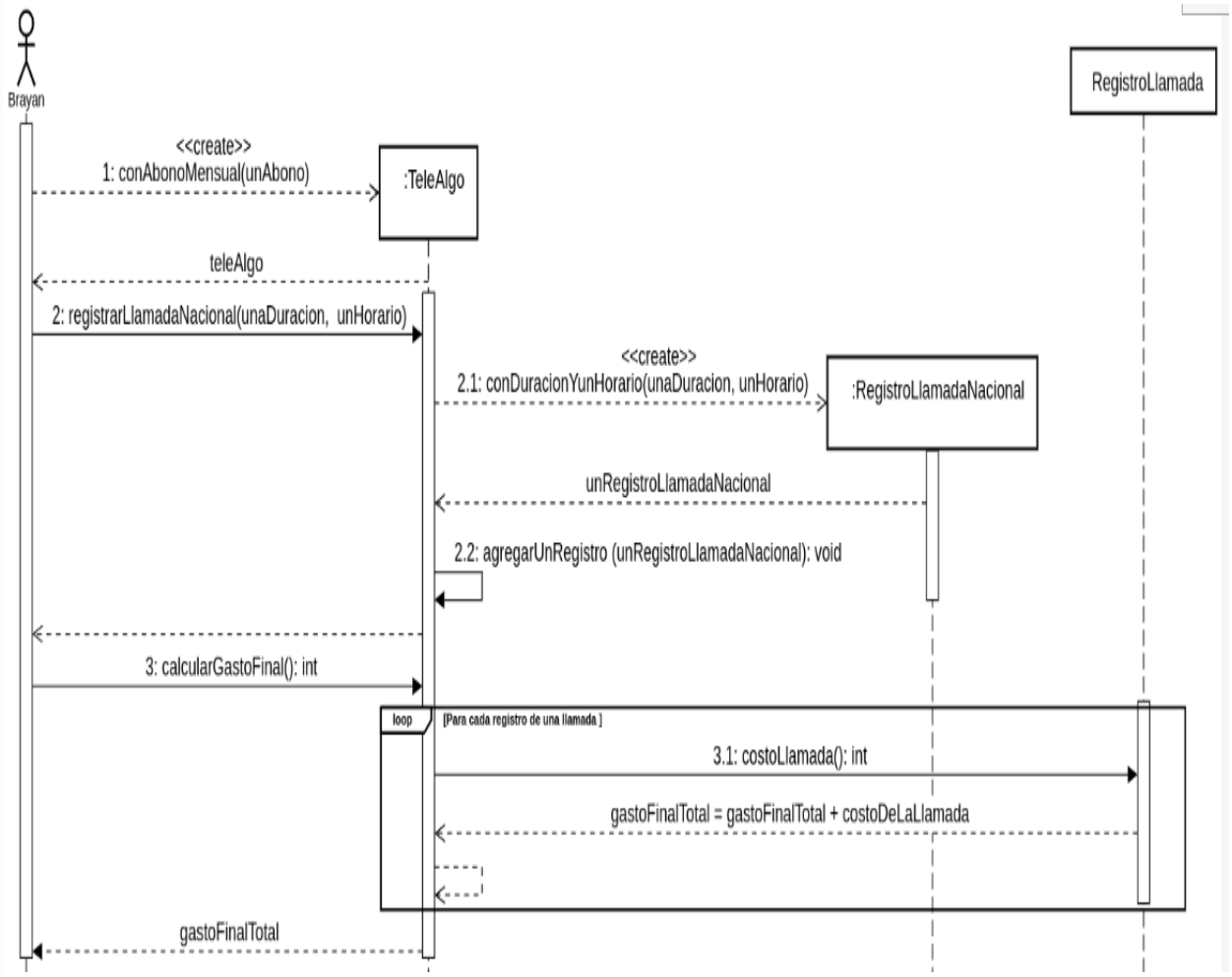

Figura 3: Se crea una instancia de la clase TeleAlgo con una abono mensual válido, luego el usuario registra una llamada nacional válida en dicha instancia de TeleAlgo, Seguidamente después de eso el usuario manda un mensaje para calcular el gasto final, y el obtiene el gasto final que solicito.

6.2. Diagrama de secuencia 02 - Se Lanza Una Excepción Cuando Se Intenta Buscar La Llamada Más Costosa

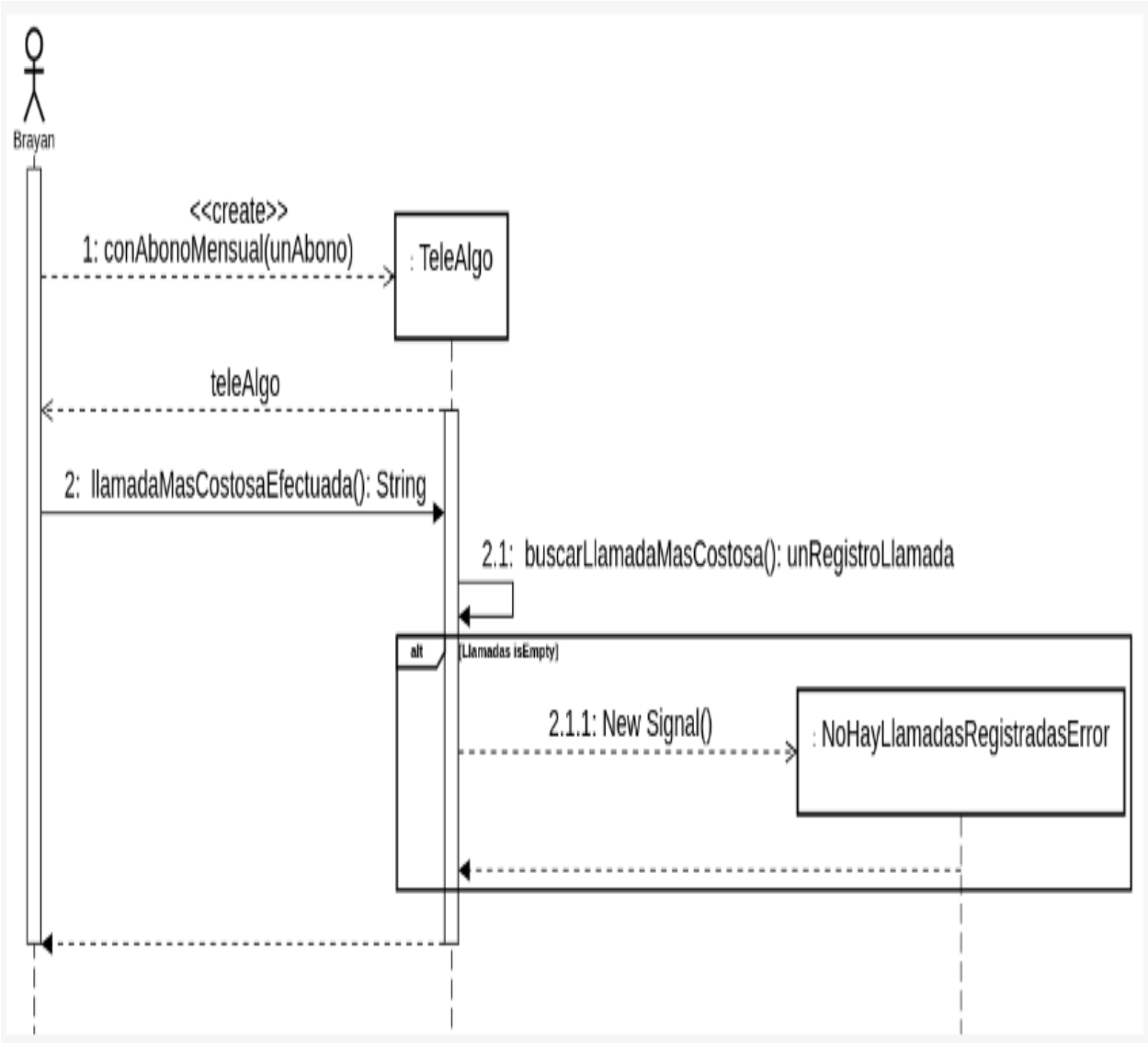

Figura 4: Se crea una instancia de TeleAlgo con un abono mensual válido, y cuando se intenta buscar la llamada más costosa efectuada, se arroja una excepción indicando que no hay llamadas registradas, por lo tanto, nunca va a existir ninguna llamada más costosa que otra.

6.3. Diagrama de secuencia 03 - Se Lanza Una Excepción Al Intentar Registrar Una Llamada Internacional Con El Pais Vacío

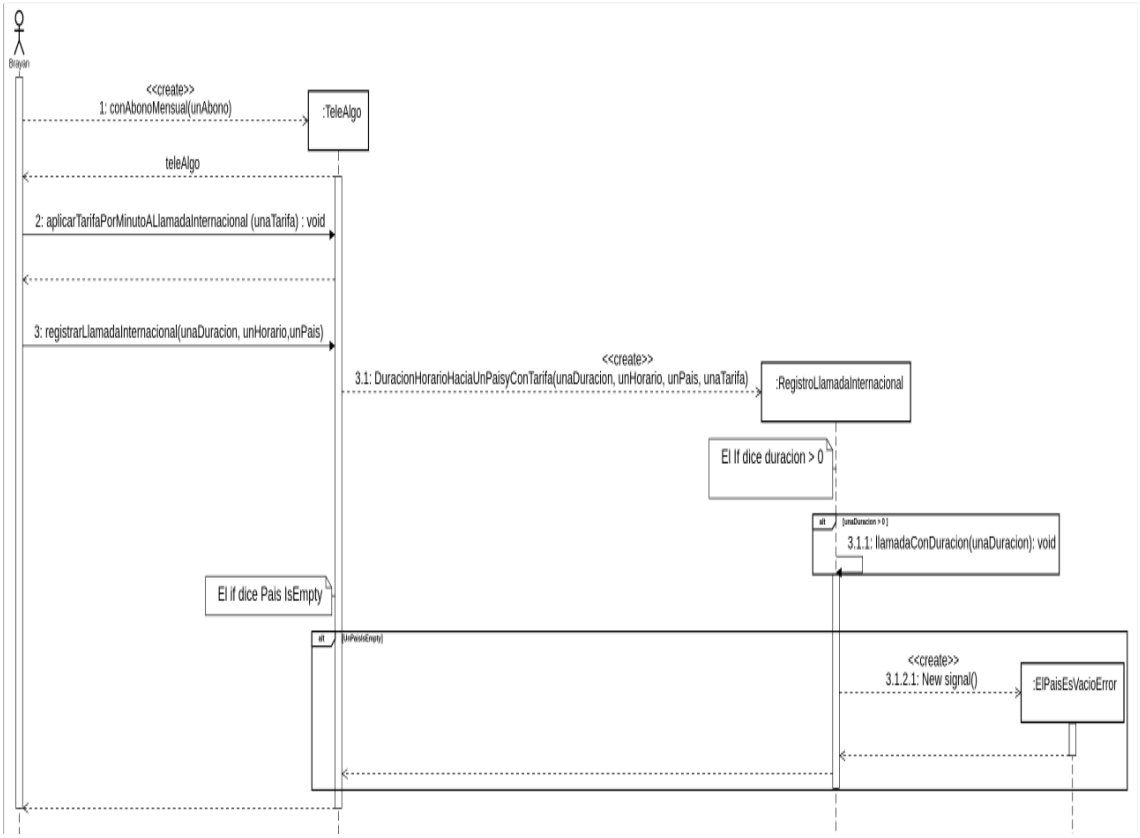

Figura 5: Se crea una instancia de la clase TeleAlgo con un abono mensual válido, luego el usuario estable una tarifa internacional para la llamada internacional que va a querer registrar, TeleAlgo le manda un mensaje a una instancia de RegistrarLlamadaInternacional (una duración válida, un horario válido, una tarifa válida, pero le mandan un país vacío), pero se arroja una excepción, puesto que el contenido del país con el que se queria registrar dicha llamada internacional está vacío.