FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2018

Alumno:	STEFANELLI D'ELIAS, Carlos Martín
Número de padrón:	100488
Email:	martin.stefanelli.96@hotmail.com

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 una aplicación de un calendario en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Los supuestos que determiné debido a algunas ambigüedades del enunciado fueron que el calendario no deja agregar más de cinco personas y cinco recursos, si una persona o recurso se encuentra en el calendario no se permitirá agregar a alguno de ellos nuevamente, en el caso de que se agregue un evento cuyo nombre ya existe lo que se logrará es reprogramar dicho evento y, por último, se pueden agregar toda la cantidad de eventos que se desee.

3. Modelo de dominio

El trabajo consite en modelar un calendario, el cual tiene diversas funciones a realizar como agregar una persona, un recurso, un evento (el cual puede ser simple o semanal), saber si una persona o recurso se encuentra ocupado o no en cierta fecha o si se encuentra en el calendario y remover un evento. Éste delega todas sus responsabilidades en una agenda por lo que la agenda tiene las mismas funciones que el calendario pero no distingue entre un evento semanal o simple, los toma a ambos como semanales.

La agenda antes nombrada conoce a un gestionador de eventos y a un diccionario de invitables. La idea es que el gestinador pueda administrar los eventos para así facilitar las operaciones que realiza la agenda con ellos y que no tenga muchas responsabilidades con el manejo de los eventos. En cuanto al diccionario de invitables, su principal uso es para que la agenda pueda hacer operaciones con los invitables sin tener que recurrir a clases auxiliares.

El gestionador de eventos guarda los eventos que se agregan al calendario y sabe si un determinado evento lo tiene almacenado. Éste gestionador se comunica directamente con los eventos, los cuales tienen información sobre los recursos y personas que participarán, las fechas en las que se desarrollara cada evento y su nombre, por lo que le es fácil accerder a la información y operaciones que tienen los eventos.

4. Diagramas de clase

En la siguiente imágen muestro la relación entre los objetos que tuve que hacer y crear para poder resolver el trabajo práctico de la manera más orienta a objetos posible. En la mayoría de las clases muestro solamente el nombre de algunos de los métodos que tiene cada clase y no su firma ya que el diagrama se hacía muy largo y detallado y consideré que esos pequeños detalles eran suficientes para mostrar en el esqueleto del programa que, como se vio en el paper que escribió Martin Fowler en "What's a model for?", la idea del diagrama es poder visualizar el esqueleto del programa y no su totalidad, donde deben haber pequeños detalles del programa para visualizarlos.

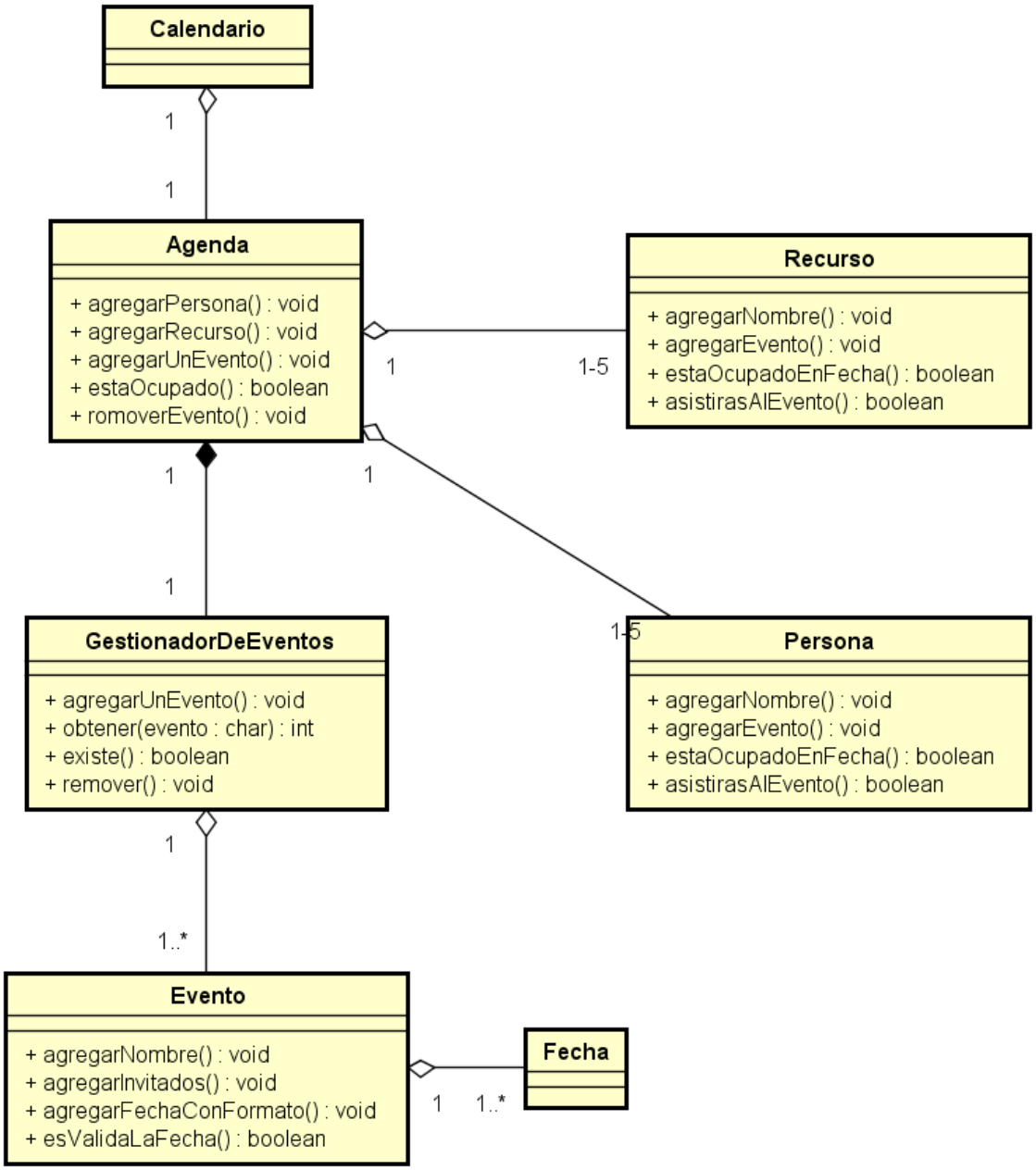

Figura 1: Diagrama del Calendario.

En la siguiente imágen muestro la relación entre la clase invitable, persona y recurso, la cual sirve para ampliar un poco la relación recién señalada que en el anterior diagrama de clases no se podía apreciar y, además, no tenía nada que ver esa relación con lo que muestro en el diagrama anterior.

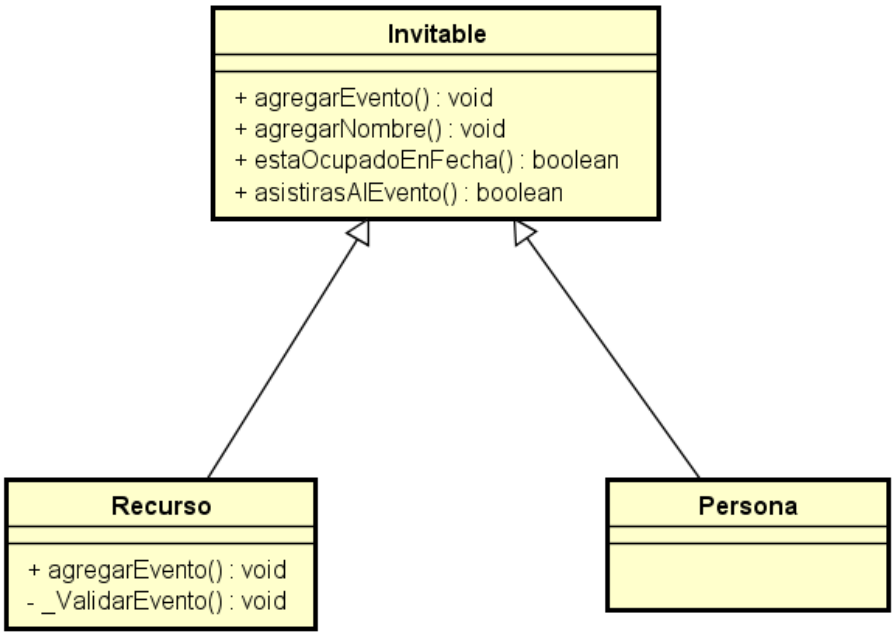

Figura 2: Diagrama de relación.

5. Detalles de implementación

5.1. Remover un evento

En la clase Agenda, en el método removerEvento, tengo un diccionario donde guardo las instancias de la clase persona y la clase recurso, las cuales entienden los mismos mensajes ya que heredan de la clase Invitable. Para remover los eventos de cada instancia de las clases nombradas tengo que recorrer los valores de los diccionarios y decirle a esas instancias que remuevan el evento ya que entienden el mensaje removerEvento, por lo que puede llegar a fallar en algún momento ese recorrido porque tal vez una persona o recurso no conoce al evento a borrar y se produciría una excepción que se debe manejar. Lo interesante es como manejo la excepción que puede llegar a lanzar en algún momento la porción de código que estoy sospechando que puede romper y lanzar la excepción en cuestion porque por más que surja esa excepción yo quiero seguir recorriendo los valores del diccionario y al estar muy acostumbrado al menejo de excepciones en Python, me encontré en un callejón sin salida hasta que descubrí que algunas excepciones tienen la posibilidad de ser ignoradas sin tener que cortar la ejecución de una porción de código o método. A lo que voy con .ªlgunas excepciones tienen la posibilidad de ser ignoradas"me refiero que se puede continuar la ejecución del método o de una porción de código si la excepción devuelve true en el método isResumable. Por eso, al manejar la excepción, lo que hago es pasarle el mensaje resume y de esa forma no hago nada con la excepción y sigo recorreindo los valores del diccionario.

5.2. Agregar un evento

En la clase Calendario se distingue la diferencia entre agregar un evento simple y otro semanal, en cambio, en la clase Agenda, no se distingue entre el evento semanal y el simple. Ésto se ve cuando a la clase Agenda se le manda el mensaje agregarUnEventoConNombre (la firma es más larga y no me parece nesesario escribirla) desde el método de la clase Calendario agregarEventoSimple, semanas es igual a uno y se lo pasa al método ya nombrado de la clase Agenda, y en el caso del método agregarEventoSemanal de la clase Calendario se le manda el mensaje ya nombrado a la clase Agenda con las semanas igual que el método que lo invoca. Ésto es intersante porque no es

necesario crear dos constructores de clase para la clase Evento y con crear uno que incluya a las semanas es lo mismo, total la lógica de las semanas se va a cumplir siempre y cuando semanas sea mayor a uno.

5.3. Relación entre Persona y Recurso

Dado que la clase Persona y la clase Recurso tienen los mismos métodos, lo que hice para evitar repetir código fue crearme la clase Invitable, la cual es una clase abstracta y con métodos no abstractos. De ésta clase heredan Persona y Recurso ya que utilicé la regla memotecnica para poder saber si se puede aplicar herencia a ciertas clases: si un objeto es un <algo>entonces podemos aplicar herencia para usar el comporatamiento de <algo>, en éste caso las personas y los recursos son invitables así que pueden heredar el comportamiento de la clase Invitable. Al mismo tiempo, como Persona y Recurso tenían el mismo comportamiento prácticamnete, salvo por el agregar un evento que para el caso de un recurso si estaba ocupado en una fecha y se quería asignarle otro evento en esa fecha no se podía (por eso redefiní la implementación de agregarEvento en la clase Recurso), pude aplicar polimorfismo y así evitar repetición de código y no necesitaba distinguir entre Persona y Recurso para su comportamiento. Si no fuera por esto hubiera tenido un gestionador de personas y otro de recursos, los cuales hacían prácticamente lo mismo, como tambíen que Persona y Recurso tenían el mismo comportamiento , y si surgía por algún motivo alguna persona y/o recurso especial iba a necesitar otras clases para manejar ésta cuestión y esa solución no iba a ser muy orientada a objetos y, además, no es muy mantenible. Con la solución de herencia, si surge algo como ésto último, simplemente esas clases heredarían de Invitables y no tendría que distinguir entre ellos para su comportamineto.

6. Excepciones

CantidadDeSemanasInvalidasError Ésta excepción surgió a partir de pensar casos borde para testear en las pruebas y sirve para cortar la ejecución del programa si al agregar un evento semanal, la cantidad de semanas pasadas al programa es menor a uno.

CapacidadMaximaDePersonasError Surgió a partir de unos de los supuestos que detallé en la sección de supuestos y la creé para cortar la ejecución del programa si se agrega una persona más de la cantidad máxima de personas definida en los supuestos.

CapacidadMaximaDeRecursosError Ídeam la excepción anterior pero para los recursos.

FechasNoDefinidasError Éste evento surgió con el fin de cortar la ejecución del programa si alguien quería obtener las fechas en las que se iba a desarrollar el evento y éste no tenía información acerca de ellas. Nació a partir de testear casos borde, y al hacer un getter al evento de las fechas, si no tiene información acerca de ellas es lógico que se tiene que lanzar esa excepción.

InvitadoNoEncontradoError Ésta excepción se lanza cuando se le quiere asignar a un invitable un evento y el invitable no existe. Surgió con el fin de evitar hacer un if para saber si un invitable está en la agenda y en función a eso le asigno el evento, por eso directamente trato de acceder a la instancia del invitable y si no lo encuentro, pido perdón y lanzo la excepción; y también surgió a partir de pensar los casos borde, de qué pasaría si agrego un evento a un invitable que no está en el calendario.

NoExisteEventoError Ésta excepción se lanza cuando se trata de remover un evento y éste no existe y cuando se quiere obtener un evento y éste no existe. Surgió con el fin de poder transmitirle al código que intentaba obtener un evento o remover un evento que el evento no existía.

NombreNoDefinidoError Ésta excepción surgió con el fin de cortar la ejecución del programa si alguien quería obtener el nombre de un evento que no tenía el nombre definido. Nació a partir de testear casos borde, y al hacer un getter al evento del nombre, si no tiene información acerca de él es lógico que se tiene que lanzar esa excepción.

PersonaYaExistenteError Ésta excepción se lanza cuando se intenta agregar una persona que ya está en el calendario. Surgió a partir de la ambiguedad del enunciado y decidí que el calendario no admite personas repetidas.

RecursoOcupadoError Surgió a partir del enunciado ya que un recurso no puede estar ocupado dos veces al mismo tiempo y porque tampoco tiene la posibilidad de elegir como la persona a qué eventos puede asistir. Los recursos no aceptan superposición de eventos y por eso, si se llega a intentar agregar un evento en una fecha que ya tiene asignada, lanza la excepción señalada.,

RecursoYaExistenteError Ésta excepción se lanza cuando se intenta agregar un recurso que ya está en el calendario. Surgió a partir de la ambiguedad del enunciado y decidí que el calendario no admite recursos repetidos.

7. Diagramas de secuencia

En la siguiente imágen muestro un diagrama de secuencia del método de Calendario agregarEvento(nombre, invitados, enAnio, mes, dia, hora) y las relaciones que surgen a partir de ese mensaje que le llega a la clase Calendario con el resto de las clases que son parte de la solución del trabajo práctico presente. Algunas cuestiones a aclarar son las de los métodos que están presentes en el diagrama, los cuales no poseen su firma completa, sino una forma abreviada de sus firmas para que quede un poco mas legible el diagrama.

Por otro lado, según lo que devuelva el método que se llama en 2.3.1 se le enviará el mensaje agregarEvento a Persona o a Recurso ya que Agenda conoce a invitables. En el diagrama parece que se llaman en distintos momentos y la Agenda diferencia las personas de los recursos pero no pasa ésto, solo que no sé como representar en el diagrama ésto que estoy aclarando.

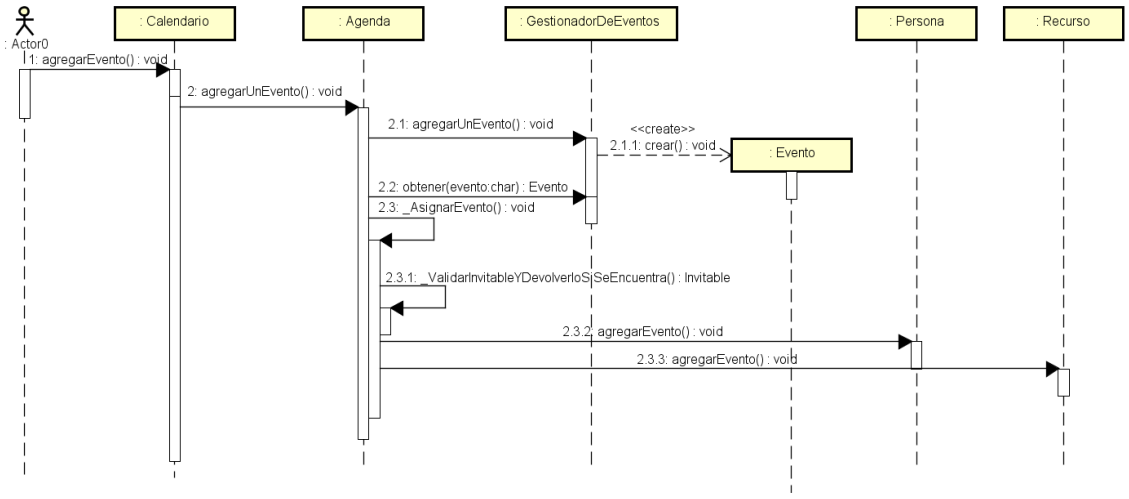

Figura 3: Diagrama del Calendario, agregar un evento simple.

En la siguiente imágen muestro un diagrama de secuencia del método de Calendario removerEvento(evento) y las relaciones que surgen a partir de ese mensaje que le llega a la clase Calendario

con el resto de las clases que son parte de la solución del trabajo práctico presente. Algunas cuestiones a aclarar son las de los métodos que están presentes en el diagrama, los cuales no poseen su firma completa, sino una forma abreviada de sus firmas para que quede un poco mas legible el diagrama.

Por otro lado, en el diagrama parece que se llaman en distintos momentos a removerEvento tanto para la Persona como para el Recurso y la Agenda diferencia las personas de los recursos, pero no sucede ésto, solo que no sé como representar en el diagrama ésto que estoy aclarando, es decir que a partir del invitable que toque cuando se recorre la lista de invitables la Agenda le manda el mensaje a una persona o a un recurso y responden al mismo como saben ellos mismos.

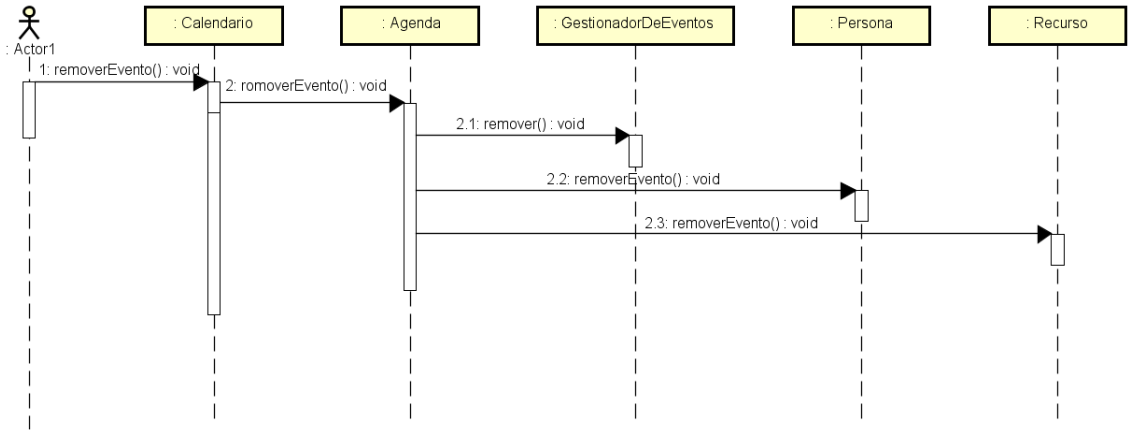

Figura 4: Diagrama del Calendario, remover un evento.