Universidad de Buenos Aires

Facultad de Ingeniería

75.10 Técnicas de Diseño

Trabajo Práctico 2

GRUPO N° 20

Integrantes:

CHELOTTI, ADRIANA GRETEL	83513	adrichelo84@yahoo.com.ar
PEREZ STALTARI, DARIO MARTIN	83514	dmpstaltari@yahoo.com.ar

Índice

Enunciado 3
Explicación 4
Diagrama de Clases 6
Diagrama de Secuencia 7
Documentación 8
Interface Accion 8
Clase Cancelador 8
Clase CanceladorPorDefecto 9
Clase Configuracion 10
Clase Evaluador 11
Clase EvaluadorContinuo 11
Clase EvaluadorDiscontinuo 12
Clase EvaluadorSecuenciaContinua 12
Clase EvaluadorSecuenciaDiscontinua 12
Clase Implicacion 13
Clase ManejadorDeSucesos 14
Clase Suceso 15

Enunciado

Mejoras a la API para manejo de sucesos

Objetivo

Dado que la API construida tuvo buena aceptación y dado el feedback recibido por las primeras pruebas en varios dominios en los cuales se les esta empezando a dar uso, nuestro cliente nos solicita algunas mejoras que son detalladas a continuación.
Se quiere que al definir el conjunto de sucesos, que notificarán o terminarán ejecutando alguna acción en algún objeto del dominio, se le pueda especificar alguna configuración adicional como la que se detalla a continuación:

- Solo si los mismos ocurren sin importar el orden (continuo y discontinuo).
- Solo si los mismos ocurren solo en orden (continuo y discontinuo)

Continuo significa que no puede ocurrir ningún otro suceso del dominio entre la secuencia de los sucesos deseados.
Al haber sucesos que cancelan a otros sucesos:

- Solo si no se canceló ningún suceso de los deseados.
- Solo sin importar si se canceló o no algún suceso, solo importa si ocurrieron.
- Si la API administra alguna lista de sucesos o estructura similar que pueda crecer ilimitadamente, configurar un máximo general en la API, que a partir de dicho valor vaya descartando el suceso más viejo.

Explicación

El Manejador de Suceso es una API que permite configurar sucesos ocurridos en un dominio cualquiera. Esta API permite que objetos puedan suscribir implicaciones a las mismas. Con implicaciones nos referimos a un conjunto de sucesos que tienen que ocurrir para que se produzca una acción de interés. Los objetos que quieran utilizar la API, deben crear dichos sucesos diferenciados por un identificador que los caracterice y pueden tener un identificador de un suceso que lo cancele.
Además deben crear una clase que implemente la interfaz Acción. Dentro de esta clase se debe implementar la acción a efectuar por el objeto cliente.

El cliente tendrá la posibilidad de configurar la API, según el comportamiento que desee.
Existen 4 configuraciones posibles:

- Sucesos Continuos.
- Sucesos Discontinuos.
- Secuencia Continua
- Secuencia Discontinua

Sucesos Continuos
En esta configuración, la comparación se da si el conjunto de sucesos esperados por la implicación está incluido en el conjunto de sucesos ocurridos sin importar el orden.

Sucesos Discontinuos
En esta configuración, la comparación se da si el conjunto de sucesos esperados por la implicación ocurrieron en su totalidad sin importar el orden pero sí que los mismos sean continuos.

Secuencia Continua
En la configuración Secuencia Continua, se comparan los sucesos de las implicaciones suscriptas a la API con los sucesos ocurridos. Para que se ejecute la acción, la secuencia de sucesos de la implicación debe ocurrir en forma continua y en orden.

Secuencia Discontinua
En la configuración Secuencia Discontinua, se comparan los sucesos de las implicaciones suscriptas a la API con los sucesos ocurridos. Para que se ejecute la acción, la secuencia de sucesos de la implicación debe ocurrir en forma discontinua y en orden.

Otra posibilidad que ofrece la API, es permitir habilitar o deshabilitar las cancelaciones entre sucesos. La cancelación de sucesos cancelables entre si se producen uno a uno. Se considera que dos sucesos son cancelables cuando el identificador del suceso cancelador del último de los sucesos ocurridos concuerda con el identificador del primero de los sucesos ocurridos.

El objeto cliente puede agregar un suceso o un conjunto de sucesos a la API, los cuales serán notificados a las implicaciones que fueron suscriptas. Los sucesos que ocurrieron antes de la inscripción de una implicación no serán visibles por esta. Teniendo en cuenta esto último, la API mirará las implicaciones inscriptas y mediante un evaluador corroborara si los sucesos que espera ocurrieron o no. En caso de cumplir con los sucesos de la implicación según la configuración establecida procederá a ejecutar la acción de interés para el objeto cliente.

Diagrama de Clases

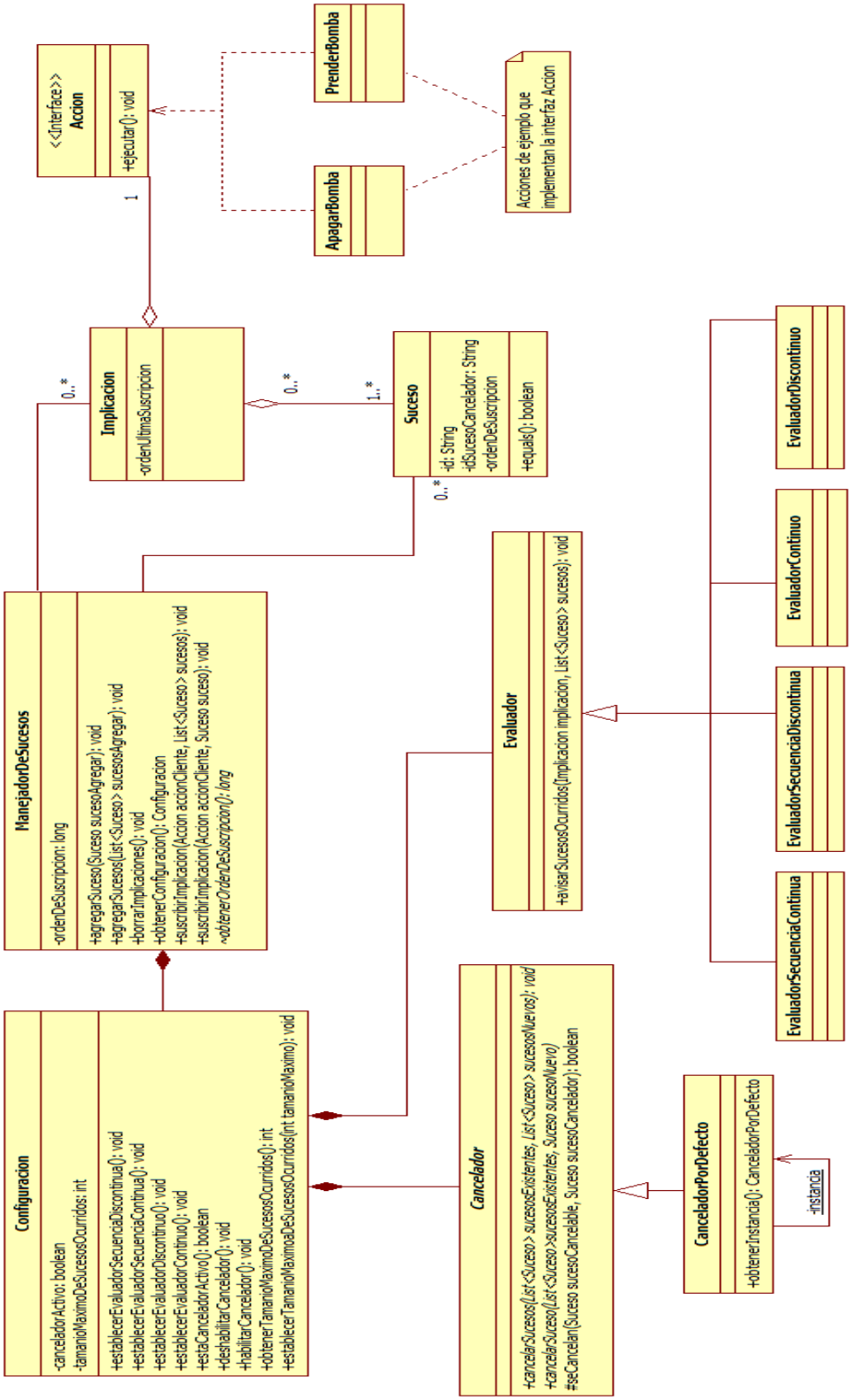
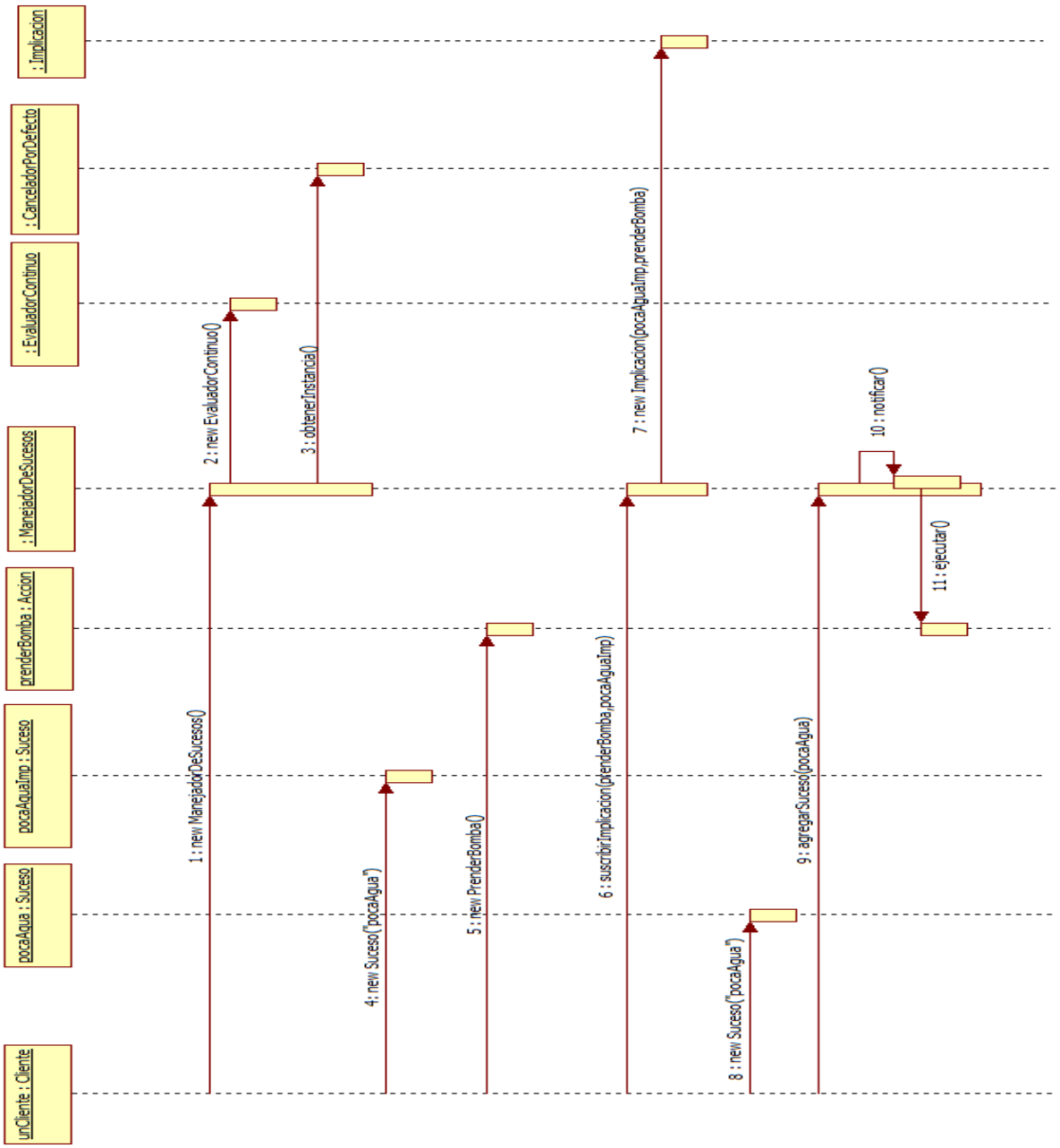


Diagrama de Secuencia

El contexto del diagrama esta definido en la creación del manejador de sucesos. Por lo cual no existen implicaciones ni sucesos suscriptos, no se encuentra habilitado el cancelador y se utiliza el evaluador continuo. En primer lugar se crean el suceso “pocaAgua” y la accion “prenderBomba”, quienes seran el antecedente y consecuente de la implicación suscripta a continuación. Seguido de esto se crea un nuevo suceso “pocaAgua” y se agrega al manejador, como consecuencia de esto se notifica a las implicaciones suscriptas, ejecutandose la acción asociada a las que tengan este ultimo suceso como antecedente.



Documentación

Interface Accion
public interface **Accion**

Interfaz que deben implementar las acciones a efectuar.

Métodos	
Void	**ejecutar**() Metodo en donde se deben implementar las acciones a realizar.

Clase Cancelador
public abstract class **Cancelador**

Clase abstracta que estable la estructura de un cancelador.

Metodos	
abstract void	**cancelarSuceso**(List<Suceso> sucesosExistentes, Suceso sucesoNuevo) Cancela los sucesos cancelables entre el conjunto y el suceso pasados como parámetro. **Parametros:** sucesosExistentes - conjunto de sucesos existentes. sucesosNuevos - conjunto de sucesos nuevos.
abstract void	**cancelarSucesos**(<Suceso> sucesosExistentes, List<Suceso> sucesosNuevos) Cancela los sucesos cancelables entre los conjuntos pasados como parametro. **Parametros:** sucesosExistentes - conjunto de sucesos existentes. sucesosNuevos - conjunto de sucesos nuevos.

Clase CanceladorPorDefecto
public class **CanceladorPorDefecto** extends Cancelador

Clase que modela un cancelador, las cancelaciones de sucesos cancelables entre si se producen uno a uno.

Métodos	
void	**cancelarSuceso**(List<Suceso> sucesosExistentes, Suceso sucesoNuevo) Cancela los sucesos cancelables entre el conjunto y el suceso pasados como parametro, **Parámetros:** sucesosExistentes - conjunto de sucesos existentes. sucesoNuevo - suceso nuevo.
void	**cancelarSucesos**(List<Suceso> sucesosExistentes, List<Suceso> sucesosNuevos) Cancela los sucesos cancelables entre los conjuntos pasados como parametro. **Parámetros:** sucesosExistentes - conjunto de sucesos existentes. sucesosNuevos - conjunto de sucesos nuevos.
static CanceladorPorDefecto	**obtenerInstancia**() Obtiene la instancia de la clase **Retorna:** Instancia del cancelador por defecto.

Clase Configuracion
public class **Configuracion**

Clase encargada de representar la configuración del manejador de sucesos.

Métodos	
void	**deshabilitarCancelador**() Deshabilita la cancelación de sucesos.
void	**establecerEvaluadorContinuo**() Cambia el evaluador a conjunto de sucesos continuos.
void	**establecerEvaluadorDiscontinuo**() Cambia el evaluador a conjunto de sucesos discontinuos.
void	**establecerEvaluadorSecuenciaContinua**() Cambia el evaluador a secuencia continua de sucesos.
void	**establecerEvaluadorSecuenciaDiscontinua**() Cambia el evaluador a secuencia discontinua de sucesos.
void	**establecerTamanioMaximoDeSucesosOcurridos**(int tamanioMaximoDeSucesosOcurridos) Carga el tamaño máximo de sucesos ocurridos. **Parámetros:** tamanioMaximoDeSucesosOcurridos - tamaño máximo de sucesos a almacenar.
boolean	**estaCanceladorActivo**() Obtiene el estado del cancelador. **Retorna**: true si el cancelador está activo. False en caso contrario.
Cancelador	**getCancelador**() Obtiene el cancelador de la configuración. **Retorna**: Cancelador actual de la configuración.
Evaluador	**getEvaluador**() Obtiene el evaluador de la configuración. **Retorna**: Evaluador actual de la configuración.

void	**habilitarCancelador**() Habilita la cancelación de sucesos.
int	**obtenerTamanioMaximoaDeSucesosOcurridos**() Obtiene el tamaño máximo de sucesos ocurridos. **Retorna:** Tamaño de la lista de sucesos ocurridos.

Clase Evaluador

public abstract class **Evaluador**

Clase abstracta que estable la estructura de un evaluador. Evalua un conjunto de sucesos con el antecedente de una implicacion en caso de cumplirse se ejecuta el consecuente.

Metodos	
abstract void	**avisarSucesosOcurridos**(Implicacion implicacion, List<Suceso> sucesos) Evalua si los sucesos cumplen con el antecedente de la implicacion. En caso afirmativo se ejecuta la accion de la implicacion. **Parámetros:** implicacion - implicacion a evaluar. sucesos - sucesos a evaluar.

Clase EvaluadorContinuo

public class **EvaluadorContinuo** extends Evaluador

Clase que modela un evaluador. Evalua un conjunto de sucesos con el antecedente de una implicacion,en el caso de que los sucesos a evaluar esten incluidos de manera continua en el antecedente se ejecuta la accion del consecuente.

Clase EvaluadorDiscontinuo

public class **EvaluadorDiscontinuo** extends Evaluador

Clase que modela un evaluador. Evalua un conjunto de sucesos con el antecedente de una implicacion,en el caso de que los sucesos a evaluar esten incluidos en el antecedente se ejecuta la accion del consecuente.

Clase EvaluadorSecuenciaContinua

public class **EvaluadorSecuenciaContinua** extends Evaluador

Clase que modela un evaluador. Evalua un conjunto de sucesos con el antecedente de una implicacion,en el caso de que los sucesos a evaluar esten incluidos en el antecedente formando una secuencia continua se ejecuta la accion del consecuente.

Clase EvaluadorSecuenciaDiscontinua

public class **EvaluadorSecuenciaDiscontinua** extends Evaluador

Clase que modela un evaluador. Evalua un conjunto de sucesos con el antecedente de una implicacion,en el caso de que los sucesos a evaluar esten incluidos en el antecedente formando una secuencia se ejecuta la accion del consecuente.

Clase Implicacion

public class **Implicacion**

Clase que modela un implicación. Una implicación está formada por un antecedente compuesto por un conjunto de sucesos y un consecuente compuesto por una acción. En caso de que se cumpla con el antecedente se ejecutara el consecuente.

Métodos	
Accion	**getAccion**() Obtiene la accion del consecuente. **Retorna:** accion que forma el consecuente.
List<Suceso>	**getSucesos**() Obtiene el conjunto de sucesos del antecedente. **Retorna:** conjuntos de sucesos que forman el antecedente.
void	**setAccion**(Accion accion) Carga la accion del consecuente. **Parámetros:** accion - accion a cargarse.
void	**setSucesos**(List<Suceso> sucesos) Carga la lista de sucesos del antecedente. **Parámetros:** sucesos - conjuntos de sucesos a cargarse

Clase ManejadorDeSucesos

public class **ManejadorDeSucesos**

Clase encargada del manejo de sucesos.

Constructor
Implicacion(List<Suceso> antecedente, Accion consecuente) Constructor de la implicación. **Parámetros:** antecedente - conjunto de sucesos que forman el antecedente. consecuente - acción que forman el consecuente.

Metodo	
void	**agregarSuceso**(Suceso sucesoAgregar) Agrega un suceso al conjunto de sucesos pendientes de notificacion. **Parámetros:** sucesoAgregar - suceso a agregar.
void	**agregarSucesos**(List<Suceso> sucesosAgregar) Agrega un conjunto de sucesos al conjunto de sucesos pendientes notificacion. **Parámetros:** sucesosAgregar - conjunto de sucesos a agregar.
void	**borrarImplicaciones**() Elimina las implicaciones almacenadas.
void	**obtenerConfiguracion**() Obtiene la configuración del manejador de sucesos. **Retorna:** Configuración del manejador de sucesos.
void	**suscribirImplicacion**(Accion accionCliente, List<Suceso> sucesos) Suscribe una implicacion a evaluar.
void	**suscribirImplicacion**(Accion accionCliente, Suceso suceso) Suscribe una implicacion a evaluar.

Clase Suceso
public class **Suceso**

Clase que modela un suceso.

Constructores
Suceso(String idSuceso) Constructor. **Parámetros:** idSuceso - identificador del suceso.
Suceso(String idSuceso, String idSucesoCancelador) Constructor. **Parámetros:** idSuceso - identificador del suceso. idSucesoCancelador - identificador del suceso con el que se cancela.

Métodos	
String	**getIdSuceso**() Obtiene el identificador del suceso. **Retorna:** identificador del suceso.
String	**getIdSucesoCancelador**() Obtiene el identificador del suceso que lo cancela. **Retorna:** identificador del suceso con el que se cancela.