Práctico de Programación 4 – clase 8

Práctico 4, Ejercicio 12[1]:

La solución a este ejercicio se encuentra en el documento *Solución laboratorio 2008* correspondiente al Práctico 4 (*Diseño, Diagramas de Comunicación*). Aquí comentaremos aspectos salientes de dicha solución, particularmente de los casos de uso (CU) Ingreso de Reclamo, Cancelar Reclamo y Liquidación de haberes.

Las operaciones del sistema a diseñar son las que surgen de los Diagramas de Secuencia del Sistema de la *Solución laboratorio 2008* correspondiente al Práctico 3 (*Análisis, Modelo de Comportamiento del Sistema*), que a su vez se basan en el modelo de dominio (segunda alternativa, que utiliza el concepto EspTarea) de la *Solución laboratorio 2008* correspondiente al Práctico 2 (*Análisis, Modelo de Dominio*).

Comentarios generales sobre la estructura de la solución:

- Por el tamaño del sistema se ha descartado la opción de tener un solo controlador fachada para todos los CU (la clase Sistema). En su lugar se han definido tres controladores (ControladorReclamos, ControladorTareas y ControladorTecnicos), aplicando el criterio Controller (teórico, clase *11 - Diseño: GRASP*) y agrupando casos de uso con cierta afinidad temática (teórico, clase *13 - Diseño: Guias*).
- Cada controlador tiene la responsabilidad (además de implementar uno o más CU relacionados con su temática) de almacenar las colecciones de todas las instancias fuertes del sistema, relacionadas con su temática. Las instancias fuertes son aquellas cuya existencia no depende de otras. De esta forma:
    - ControladorReclamos contiene todas las instancias de Reclamo.
    - ControladorTareas contiene todas las instancias de EspTarea y de TipoArticulo.
    - ControladorTecnico contiene todas las instancias de Tecnico (que serán instancias de las clases concretas Mensual o Jornalero).
- Notar además que:
    - Ningún controlador tiene la responsabilidad de almacenar todas las instancias de Tarea ni de Registro, ya que estas instancias son débiles (existen debido a la existencia de otras clases).
    - Los controladores se diseñan utilizando el patrón Singleton (ver teórico, clase *15 - Diseño: Patrones de Diseño*), dado que existe solamente una instancia de cada uno. Por esa razón, antes de utilizar un controlador se debe pedir a su clase la única instancia existente, mediante la operación getInstance. Observar que, si un controlador es el responsable de almacenar todas las instancias del sistema de una determinada clase, no pueden existir varias colecciones de dichas instancias.
    - Los controladores pueden comunicarse entre ellos, generalmente para obtener información que no está directamente a su alcance. Notar que para esto es necesario crear operaciones adicionales que NO son operaciones del sistema (no están en ningún DSS).

[1] En esta discusión se hacen algunas precisiones y correcciones sobre las soluciones publicadas del Laboratorio 2008.

Ingreso de Reclamo

Es probablemente el CU más complejo, requiere ingresar información variada e involucra varias interacciones con el sistema. Además, las acciones se confirman recién al final del CU (tal como lo establece su descripción), por lo tanto, todos los cambios a realizarse deben almacenarse en la memoria y en caso de que el usuario confirme, se impactan en el sistema (de lo contrario no se modifica nada). A continuación, se comentan los diseños de cada una de las operaciones del CU:

ingresarDatosReclamo:

- Se obtiene la instancia del ControladorTareas.
- Se le pide que devuelva la instancia de TipoArticulo a la que hace referencia el reclamo a ingresar. Notar que se hace un chequeo de existencia que no es estrictamente necesario si se modela el curso típico de eventos; una precondición de esta operación debería ser que exista en el sistema una instancia de TipoArticulo con el código ingresado. Por otra parte, la búsqueda de la instancia de TipoArticulo en esta operación, solo tendría sentido si se va a recordar para no tener que buscarla nuevamente en las siguientes operaciones.
- Se almacena en la memoria los datos del reclamo a ingresar, incluyendo el identificador de la instancia de TipoArticulo.

obtenerTareasAsocTipoArticulo:

- Se le pide a ControladorTareas que devuelva una colección de datos de las instancias de EspTarea que apliquen al TipoArticulo identificado por el código (alternativamente, se podría haber utilizado directamente el identificador de TipoArticulo recordado). Para eso, se recorre la colección de todas las instancias de EspTarea y a cada una se le pregunta si está asociada al TipoArticulo de interés. Notar que aquí se tomó una decisión de diseño importante: cada instancia de EspTarea tiene una colección de los TipoArticulo a los que aplica (que es diferente de la colección de todos los TipoArticulo del sistema).

obtenerTecnicosCapacitadosTipoArticulo:

- Análogo al anterior, pero interactuando con ControladorTecnicos. Notar que aquí también se tomó una decisión de diseño importante: cada instancia de Tecnico tiene una colección de los TipoArticulo sobre los que está capacitado (que es diferente de la colección de todos los TipoArticulo del sistema).

agregarTareaReclamo:

- Cada una de las operaciones anteriores devuelve una lista: una de EspTarea y otra de Tecnico.
- A partir de esas dos listas, el usuario selecciona pares de identificadores de EspTarea y Tecnico (varias veces, tantas como tareas desee planificar).
- Esta operación simplemente recuerda a nivel del sistema, una lista de pares de datos que identifican EspTarea y Tecnico. Aún no se realiza ninguna modificación en el sistema.

obtenerDatosIngresadoReclamo:

- Pide al sistema todos los datos recordados referentes al reclamo que se va a ingresar, para que se pueda mostrar al usuario y luego pedirle que confirme o cancele.

confirmarAltaReclamo:

- Esta operación debería estar dentro de un bloque Opt en el DSS, o alternativamente debería recibir un parámetro booleano que indique si se desea confirmar o cancelar. Notar que es la operación más compleja del CU, ya que es la responsable de crear todos los objetos y links necesarios para ingresar un reclamo.
- Lo primero que se hace es acceder a la información recordada del reclamo: sus propios datos, el identificador del TipoArticulo y la lista de pares de datos de EspTarea y Tecnico.
- Se crea un Reclamo con sus datos y se le delega la responsabilidad de crear todos los demás objetos y links. Esta solución da prioridad a los criterios GRASP alta cohesión y bajo acoplamiento para no sobrecargar al ControladorReclamos. Una solución alternativa y válida, sería que ControladorReclamos asuma algunas de esas responsabilidades. Notar que de esa forma se evitaría que una instancia de Reclamo se comunique con ControladorTareas.
- Las acciones que realiza la instancia de Reclamo son las siguientes:
    - Pide al ControladorTareas que devuelva la instancia de TipoArticulo, a la que el reclamo luego tendrá una referencia.
    - Para cada par de datos de [EspTarea,Tecnico] recordado pide al ControladorTareas que obtenga la instancia de EspTarea, a la cual se transfiere la responsabilidad de crear la Tarea (la cual deberá a su vez crear su link al Tecnico, a través del ControladorTecnico).
- Finalmente se agrega el Reclamo recién creado a la colección de todos los reclamos del sistema, almacenada en ControladorReclamos.
- Notar que esta operación toma decisiones de diseño importantes relativas a las navegabilidades. En particular, algunas se diseñan de modo bidireccional para facilitar operaciones de otros CU (esto se verá con más detalle en el práctico 5, en particular en la discusión de la solución del Laboratorio 2008).

Cancelar Reclamo

Haremos una reformulación de la descripción de este caso de uso, y por lo tanto, de su correspondiente operación y su contrato, con respecto a lo especificado en la solución publicada del Laboratorio 2008.

Nombre	Cancelar Reclamo
Actores	Administrador
Sinopsis	El caso de uso comienza cuando el usuario Administrador desea cancelar un reclamo. Para ello indica el número del reclamo. Podrán eliminarse reclamos que tienen tareas planificadas pero que no tienen registro de horas trabajadas.

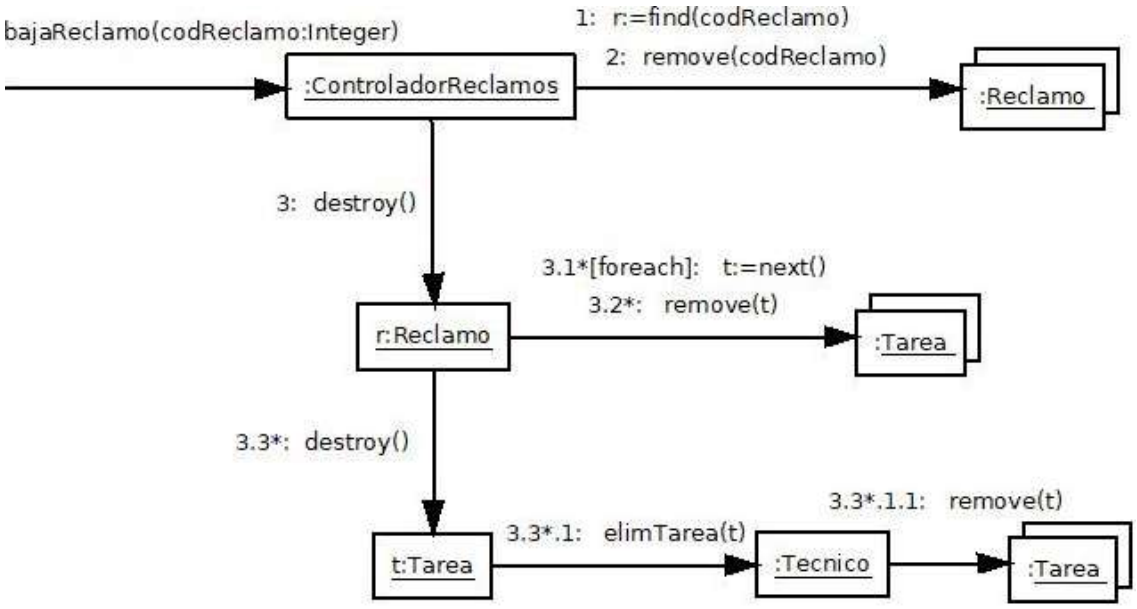
El DSS es el mismo que el propuesto en la solución publicada, sin embargo, el contrato de su única operación sería el siguiente:

Firma	`bajaReclamo(codReclamo: Integer)`
Parámetros	`codReclamo` es el identificador de la instancia de Reclamo a eliminar
Responsabilidades	Eliminar un reclamo del sistema, junto con sus tareas planificadas.
Referencias cruzadas	Cancelar Reclamo
Salida	No tiene

Precondiciones	1. Existe en el sistema una instancia de `Reclamo` identificada por `codReclamo`. **2.** Las instancias de `Tarea` asociadas al `Reclamo` no tienen instancias de `Registro` asociado.
Postcondiciones	1. Se eliminan todas las instancias de `Tarea` asociadas a la instancia del `Reclamo` identificado por `codReclamo` y se eliminan todos los links desde y hacia las instancias eliminadas. **2.** Se elimina la instancia de `Reclamo` identificada por `codReclamo` y todos los links desde y hacia ella.

Notar que, en la implementación esta operación requerirá de dos operaciones complementarias: una para verificar la precondición 1 y otra para verificar la precondición 2.

El diagrama siguiente sustituye al de la solución publicada.



bajaReclamo:

- Se obtiene la instancia del Reclamo a eliminar, a partir de su código. Se mantiene una referencia a la misma.
- Se elimina de la colección de todos los reclamos del sistema.
- Se delega la responsabilidad al Reclamo de destruir las instancias de sus Tareas (acciones que en la implementación serán ejecutadas por el destructor).
- Para la destrucción de links:
  - Los links salientes de Tarea (hacia EspTarea y Tecnico, creados en el CU Ingreso de Reclamo) no es necesario eliminarlos explícitamente, dado que se asume que se destruyen junto con la Tarea. En términos de implementación, serán referencias o punteros en la clase Tarea.
  - Los links entrantes (desde Tecnico, creados en el CU Ingreso de Reclamo) deben destruirse explícitamente. La instancia de Tecnico debe ser notificada de que no existe más la Tarea, de lo contrario va a estar haciendo referencia a objetos que no existen más en el sistema. Dado que la Tarea tiene una referencia a su (único) Tecnico, le envía un mensaje para que esté elimine su referencia a ella.

Liquidación de Haberes

El DSS tiene una sola operación.

obtenerLiquidacionMensual:

- Observar que devuelve un conjunto de datavalues, cada uno conteniendo los datos del técnico y el monto a pagarle. Este monto se calcula de forma diferente, según el técnico sea mensual o jornalero.
- Se recorren todas las instancias de Tecnico del sistema.
- A cada una se le pide que devuelva los datos de la liquidación. Como la clase Tecnico es abstracta, la primera parte del diagrama finaliza ahí.
- Luego, para cada clase concreta derivada de Tecnico (que debe tener un método para la operación getDataLiquidacionTecnico) debe hacerse un diagrama distinto. De esta forma:
    - En la clase Mensual, el método puede resolverse con información que está en la propia clase, por lo tanto, el diagrama finaliza en Mensual (no se requiere interacción con otras clases).
    - En la clase Jornalero, el método debe recorrer todas las instancias de Tarea del Jornalero (accesibles a través de la asociación heredada desde Tecnico). A su vez, cada Tarea debe recorrer su Registro (omitido en la solución) y contabilizar solamente las horas correspondientes al mes indicado por el parámetro de la operación obtenerLiquidacionMensual.

Observación final:

- Todos los mensajes enviados a multiobjetos (que representan colecciones) son exclusivamente operaciones de colecciones (next, add, find, remove, etc), ver teórico de clase 10 (Diseño: Diagramas de Comunicación).
- Debido al software utilizado para realizar los diagramas en este documento, hay una desviación con respecto a la sintaxis de UML. Cuando un objeto envía un mensaje a otro, la línea que los une no lleva flecha. La flecha corresponde al mensaje y se dibuja debajo de su nombre.