Proyecto Integrador

Fecha de publicación: 16 de marzo de 2015
Fecha de entrega obligatoria de la primera etapa: 6 de abril 2015
La URL para realizar la entrega online será enviada por el grupo de google, así como el horario en el que dicha posibilidad caduca.
Carácter: grupal (3 integrantes)

Completar el siguiente formulario para registrar el grupo de trabajo:

https://docs.google.com/forms/d/1S1yBlgnh12kyYM9AqzjAem-Hfg9Sgenc1NyvT3vO_t4/viewform?usp=send_form

Características generales

El proyecto integrador, consta de las siguientes etapas:

- implementación usando Java de la especificación brindada
  - Para poder validar la implementación realizada, se provee una serie de tests que deberán pasar.
- mapeo usando Hibernate
- implementación de consultas usando HQL

Software y versiones

Para configurar sus equipos, pueden seguir las instrucciones detalladas en este sitio:

http://catedras.s3.amazonaws.com/bbdd2-2015/instructions.html

Etapa de implementación en Java

El proyecto consiste en un administrador de tareas (ver por ejemplo el sitio trello.com para entender cómo funciona).

Hay usuarios y proyectos, y cada proyecto a su vez tiene tareas organizadas en pizarras. Las tareas se pueden ir moviendo de una pizarra a la otra (por ejemplo “Hacer” a “Hecho”), y en ese caso la tarea se guarda un pequeño historial (Paso) que indica que se movió a una pizarra en una fecha dada.

Los proyectos tienen usuarios comunes y administradores, y en particular un administrador es el creador del proyecto, que no puede ser eliminado del mismo. Los otros usuarios sí pueden ser eliminados del proyecto.

En un proyecto se pueden archivar pizarras, no se pueden borrar.

Se proporciona una especificación a partir de un diagrama de clases UML. La misma se encuentra en el archivo "*Modelo.pdf*" de la carpeta Proyecto Integrador/ETAPA 1.
A continuación se brinda una especificación complementaria de ciertos métodos con lógica particular. La lógica del resto de los métodos debería ser evidente.

Importante: todo el diagrama de clases debe ser implementado respetando el protocolo y los tipos dados.

Clase	Método	Comportamiento
Proyecto	public Proyecto(Usuario creador)	Constructor. El usuario enviado como parámetro tiene que agregarse al proyecto con perfil de administrador, y marcarse como creador.
Proyecto	public void agregarAdministrador(Usuario usuario)	Genera un perfil de usuario administrador (PerfilDeAdministrador) con el usuario enviado como parámetro, y lo agrega a la colección de perfiles.
Proyecto	public void agregarColaborador(Usuario usuario)	Idem agregarAdministrador, pero con un PerfilDeUsuario.
Proyecto	public Collection<Usuario> getIntegrantes()	Retorna los usuarios dentro de los perfiles. Notar que se espera que retorne objetos de la clase Usuario, no de la clase PerfilDeUsuario.
Proyecto	public void eliminarUsuario(Usuario candidato) throws Exception	Elimina al usuario enviado como parámetro. Si el usuario que se desea eliminar es el creador, lanzar una excepción con el mensaje: "**No se puede eliminar al creador del proyecto**"
Proyecto	public void archivarPizarra(Pizarra pizarra)	Archiva la pizarra enviada como parámetro. Esto significa que deja de estar en la colección de pizarras para pasar a la colección de pizarras archivadas.
PerfilDeUsuario	public PerfilDeUsuario(Date fechaDeCreacion, Usuario usuario)	Constructor, toma la fecha de creación y al usuario como parámetro.
PerfilDeUsuario	public void eliminarDe(Proyecto proyecto) throws Exception	Elimina el perfil del proyecto enviado como parámetro.
PerfilDeAdministrador	public void eliminarDe(Proyecto proyecto) throws Exception	Al igual que su comportamiento heredado, elimina el perfil, pero en caso de que el perfil sea creador, lanza una excepción con el mensaje: "**No se puede eliminar al creador del proyecto**"
Tarea	public Collection<Paso> getPasos()	Retorna el historial de pasos, donde cada objeto Paso representa la incorporación de la tarea en una pizarra, en una fecha dada.
Tarea	public void agregarAPizarra(Pizarra pizarra)	Se agrega a la pizarra enviada como parámetro. Debe registrar este movimiendo generando un nuevo Paso y agregándolo a su colección de pasos, con la fecha actual y la pizarra en cuestión.
Tarea	public Boolean vencida()	Retorna true si la fecha límite de la tarea ya pasó, false en caso contrario.
Pizarra	public void agregarTarea(Tarea tarea)	Agrega **tarea** a la pizarra receptora. Notar además que la tarea tiene que

		registrar su incorporación a la pizarra.
Pizarra	public void moverTareaAPizarra(Tarea tarea, Pizarra destino)	Mueve una tarea propia a la pizarra **destino**. Luego de esto, naturalmente, la tarea **tarea** no debe estar más en la pizarra receptora sino en **destino**. Notar además que la tarea tiene que registrar su paso a la pizarra **destino**.

Tener en cuenta para la implementación:

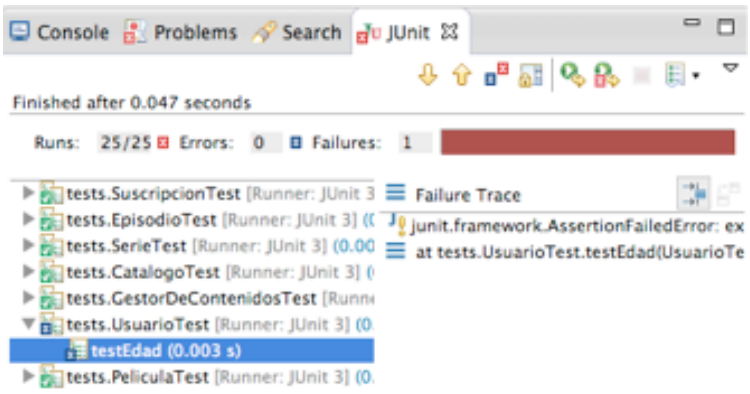
- Usar Collection al momento de *tipar* colecciones, luego se instanciará con una clase concreta
  Por ejemplo:
  ```
  private Collection<Usuario> usuarios = new HashSet<Usuario>()
  ```
- En particular, en la clase **Tarea** instanciar la colección de pasos con la clase **ArrayList** mientras que el resto de las clases hacerlo con la clase **HashSet**.

2- Una vez implementado el modelo, ejecutar los tests provistos en el proyecto. Para correr los tests basta con presionar el botón derecho sobre un método, una clase o incluso un paquete, y seleccionar:

Run As -> JUnit Test

En la ventana de JUnit se mostrarán los resultados (en caso de no ver la ventana, se puede abrir desde **Window -> Show View -> Other... -> JUnit**).

Si algún test falla aparecerá el indicador en rojo y, en el reporte, se destacarán los que fallaron con un ícono azul. En estos casos, se muestra también el Failure Trace, un detalle de las pruebas específicas (assertions) que fallaron.



En el ejemplo que se ve en la figura, el test del método edad() de la clase Usuario falló (puede verse en el ícono azul), y en el Failure Trace se indica el resultado esperado en contraste con el obtenido de la siguiente manera:

junit.framework.AssertionFailedError: expected:<20> but was:<21>

Para ir directamente al código que produjo el fallo se puede hacer doble click en el método dentro del listado del reporte. Una vez corregidos los errores, al correr nuevamente los tests el indicador debería quedar en verde.

Consideraciones contempladas para la aprobación de la primera etapa

1) Respetar nombre del proyecto y paquetes
2) Las clases y métodos deberán estar documentados
3) El comportamiento debe ser el solicitado
4) La implementación debe respetar la especificación dada
5) Los test deberán ejecutar y pasar.
   Importante: Los test no deben ser alterados.
6) La entrega se debe realizar a término

Etapa de mapeo en Hibernate

El documento de esta etapa se publicará antes del 6 de abril

Etapa de Consultas HQL, cumplimentación del Proyecto Integrador

El documento de esta etapa se publicará antes del 4 de mayo