Diapositiva 1: del análisis del caso de estudio y el material brindado se identifican 4 clases de entidad, una clase control para gestionar las entidades y una clase límite o interface para interactuar con el usuario, que en este caso es un participante que desea inscribirse a uno o más cursos que dicta el Centro Cultural Municipal.

Se decide realizar la implementación utilizando el patrón modelo vista controlador (MVC) junto al patrón DAO que es un patrón de diseño que permite disponer de un componente que actúe como puente de implementación entre los datos almacenados en los objetos de la lógica del negocio y los datos de persistencia. (este patrón encapsula la lógica necesaria para copiar los valores de los datos desde las clases del dominio del problema hacia la persistencia y viceversa. Suele ofrecer métodos para añadir, actualizar, buscar y borrar elementos)

Diapositiva 2: de acuerdo a esta decisión de diseño surge esta distribución de paquetes donde se van a guardar en el paquete poo.cursosccm.entity las clases entidad y en el paquete resources.hbm los archivos xml de mapeos necesarios para el uso de hibernate. Esto corresponde a la capa modelo del patrón MVC.

Diapositiva 3: siguiendo con el patron MVC tenemos en el paquete poo.cursosccm.ui las clases necesarias para interactuar con el usuario, no sólo el formulario o pantalla de inscripción, sino también los modelos para la tabla y el comboBox y el “oyente” para el comboBox. Estas clases estan puestas en este paquete porque interactúan directamente con la interfaz de usuario para brindar funcionalidad a la interfaz.

Diapositiva 4: aquí tenemos una vista del formulario ya terminado que respeta el prototipo propuesto en la descripción, donde puede apreciarse la caja de texto para ingresar el e-mail del participante, el comboBox para elegir las temáticas y la tabla para mostrar y elegir los cursos en los que desea inscribirse, como así también los botones buscar, inscribirse y cancelar que con su funcionalidad permiten implementar la solución propuesta.

Diapositiva 5: esta es una vista de la clase TematicasComboBoxModel que hereda de AbstractListModel e implementa la interface ComboBoxModel y nos permite adaptar la funcionalidad del comboBox a nuestras necesidades.

Diapositiva 6: lo mismo pasa con la clase TemáticasItemListener que implementa la interface ItemListener y sobre-escribe su método itemStateChanged para cargar la tabla de cursos con los cursos correspondiente según la temática seleccionada. Para esto se asocia con la clase pantallaInscripcion, la clase gestorInscripcion y la clase CursosTableModel para lograr su fucionalidad.

Diapositiva 7: vemos aquí la última clase del paquete de vista del modelo MVC, la clase CursosTableModel que hereda de AbstractTableModel y sobreescribe algunos de sus métodos para darnos una funcionalidad asociada a los cursos. Para ello recibe una Lista de objetos de tipo Cursos y modifica el comportamiento e algunos de los métodos de la clase padre para dar funcionalidad asociada a los cursos. También agrega algunos métodos específicos propios para cumplir con su responsabilidad.

Diapositiva 8: finalmente tenemos el paquete poo.cursosccm.controller que corresponde a las clases controladoras del modelo MVC y encontramos en este paquete a la clase GestiónInscripcion que es la responsable de gestionar todos los aspectos de la inscripción del participante. También se decidió poner en este paquete a la clase Main que es la encargada de dar inicio a la ejecución del programa y que solo contiene la creación y

configuración de la SessionFactory de Hibernate y una llamada al método run() de la clase GestorInscripcion pasándole de esta manera el control. La clase Main podría haberse puesto en otro paquete ya que es una clase un tanto especial que todo desarrollo java debe tener pero para no agregar paquetes innecesariamente se la colocó acá.

Diapositiva 9: vemos que la clase GestorInscripción tiene todos los métodos necesarios para ocuparse de la inscripción de un participante y para lograr su funcionalidad se asocia con clases pertenecientes al patón DAO que explicaremos en un momento. Vemos en las líneas 32 a 35 estos objetos que se usarán en los métodos respectivos para la gestión de cada una de las clases entidad como puede observarse mas abajo.

Diapositiva 10: Como dije antes se eligió el patrón DAO porque provee un nivel más de abstracción en la capa de persistencia y permite modificar fácilmente la forma de persistencia utilizada sin modificar las clases de la lógica del negocio. Para ello utiliza interfaces, que son las que el resto de nuestro sistema conoce, e implementaciones específicas de esas interfaces, en este caso asociadas a hibernate para un mapeo con la base de datos, pero que fácilmente pudieran ser reemplazadas por otra forma de persistencia como archivos xml, archivos de texto plano o lo que sea sólo modificando la implementación de la interface correspondiente sin necesidad de que nuestro sistema se entere de esto ni modifique nada más.

Diapositiva 11: este es un ejemplo concreto de esta característica, en este caso el método buscarParticimanteXemail, que como vemos en el javaDoc correspondiente devuelve una única instancia de la clase Participantes que coincide con el campo email que recibe como parámetro.

Diapositiva 12: otro ejemplo de implementación de este patrón: la interface InscripcionDAO con los métodos necesarios para realizar la funcionalidad requerida...

Diapositiva 13: y la correspondiente implementación en la clase InscripcionDaoHibernateImpl para realizar las acciones. Ésta sería la única clase que deberíamos modificar si cambiáramos a otra forma de persistencia.

Diapositiva 14: a partir de aquí comienza el desarrollo del diagrama de secuencias y la correspondiente implementación en java. En primer lugar tenemos este grupo de acciones que se describen en los puntos 1 al 6 del flujo básico en la descripción del resumen esencial del caso de uso y que se reflejan en estas secuencias y pasos de mensajes entre las clases...

Diapositiva 15: que son implementadas mediante esta porción del formulario y estos métodos correspondientes de las clases PantallaInscripción, GestorInscripción y su ayudante ParticipantesDaoHibernateImpl, con el correspondiente paso de mensajes entre ellas que se pueden apreciar en los respectivos métodos. Para esto se hace uso también de la API Criteria.

Diapositiva 16: luego tenemos los puntos 7 al 9 en el flujo básico de la descripción del resumen esencial del caso de uso y que se reflejan en estas secuencias y pasos de mensajes entre las clases. Aclaro que he puesto la cabecera del diagrama como para poder seguir a qué objetos pertenecen.

Diapositiva 17: esta secuencia se ve implementada en el método habilitarInscripción de la clase PantallaInscripcion, método al que se llama si, como vimos en la diapositiva 15 (dos para atrás), el método buscarParticipante de la clase GestorInscripcion nos devuelve un participante válido, en cuyo caso se rellenarán los label con el nombre y el apellido

(línea 329 y 330) y, como vemos en el desarrollo del método:

- se carga el modelo del comboBox con las temáticas, (línea 324)
- se le asigna un oyente para sus eventos de cambios de item, (línea 325)
- se carga el modelo de la tabla con los cursos correspondientes al tema elegido, (línea 326),
- se le asigna el modelo a la tabla. (línea 327)
- se coloca en true el radioButton de Estudiante Registrado (línea 328)
- y se hacen disponibles y visibles tanto el comboBox como la tabla, que hasta ese momento no se podían manipular (líneas 331 a 333)

Diapositiva 18: aquí se pueden ver la funcionalidad de los métodos correspondientes en las clases afectadas para la funcionalidad: el método sobre-escrito de TemáticasItemListener que llama al método listarCursos de la clase GestorInscripción cuando se produce una selección en un item del comboBox Temáticas de la clase PantallaInscripcion, que a sus vez le pide a su clase ayudante CursosDaoHibernateImpl que que le devuelva una lista de Cursos correspondientes a la temática seleccionada a través de su método listarCursos.

Diapositiva 19: si todo hasta aquí salió conforme a lo estipulado en el flujo de éxito de la descripción esencial del caso de uso, entonces da lugar a las últimas secuencias del diagrama de secuencias y la implementación correspondiente. (vuelvo a aclarar que coloqué la cabecera del diagrama como para poder entender a que pertenece cada linea de vida.

Diapositiva 20: esta sería la pantalla que implementa estas acciones, cuando el participante selecciona una temática y un curso y pulsa el botón inscribirse, vemos que este evento primeramente verifica que se haya seleccionado un curso y luego, si esto es así, llama al GestorInscripción invocando su método generarInscripción pasándole como parámetro un participante y un curso.

Diapositiva 21: finalmente, es el gestorInscripcion el encargado de generar la inscripción correspondiente ayudado por toda la funcionalidad que la clase InscripcionDaoHibernateImpl provee. El gestor es el encargado de hacer las validaciones necesarias de acuerdo a las especificaciones y para ello usa continuamente los métodos de la clase InscripcionDaoHibernateImpl. Mostrando mensajes al usuario tanto con la salida de éxito como si alguna validación no se cumple. Y con esto damos fin al caso de uso y a su implementación.