

Nombre: _____

Duración: 2,5 horas (+ media hora para los que asisten a la parte III)

Puntuación: 8 puntos (+ 1 punto para la parte III, a los que se aplique)

Parte II: problemas

Duración: 1,5 horas

Puntuación: 4 puntos

Conteste a **dos de los tres siguientes problemas**, cada problema en una hoja separada. No se permite el uso de libros ni apuntes.

Problema 1 (2 puntos)

(a) **(1,25 puntos)** La base de datos de una tienda contiene las siguientes tablas:

Productos

Codigo_producto	Ref_proveedor	Unidades_por_paquete	Info_producto

Detalle_pedidos

Fecha_pedido	Ref_pedido	Codigo_producto	Cantidad

Pedidos

Ref_pedido	Ref_proveedor	Importe	Metodo_de_pago

Existencias

Ref_existencia	Codigo_producto	Cantidad	Localizacion

Proveedores

Ref_proveedor	Nombre	Direccion	Tel	Fax	Web	E-mail

Ventas

Codigo_producto	Precio	Anho	Semana	Unidades_vendidas

donde cada producto se obtiene de un (y sólo un) proveedor y el `Importe` de cada pedido es la suma de los precios de todos los productos de ese pedido. Puede asumir la existencia de las funciones `currentDate()`, `currentWeek()` y `currentYear()`, que retornan la fecha actual, la semana en curso y el año actual, respectivamente, en el formato requerido. Se le pide que proporcione sentencias SQL que implementen las siguientes funcionalidades:

Nombre: _____

- (i) **(1 punto)** Se desea crear un pedido de un paquete de cada producto que está agotado de existencias (`STOCK.Quantity = 0`) en el almacén (`Localizacion = 'almacen'`); proporcione sentencias que implementen las correspondientes modificaciones de la base de datos. Puede suponer que
1. no se puede hacer más de un pedido al día a un proveedor dado, de modo que cada nueva `Ref_pedido` se puede construir concatenando la fecha actual al final de la correspondiente `Ref_proveedor` (el operador SQL de concatenación es `||`),
 2. el valor del `Metodo_de_pago` para todos los Pedidos es la cadena 'CASH',
 3. todos los productos que se quieren pedir tienen un precio en Euros (diferente de cero) que puede cambiar semanalmente (se puede obtener el precio actual de la tabla `Ventas`)
 4. el valor por defecto de la columna `Fecha_pedido` es el resultado de ejecutar la función `currentDate()`.

Pistas:

1. Necesitará crear una nueva fila en ambas tablas `Detalles_pedidos` y `Pedidos`; no se preocupe de los aspectos transaccionales de los comandos empleados para hacer tales operaciones.
 2. Existe una variante del comando que crea una nueva fila que requiere, en lugar de simplemente proporcionar los valores de la nueva fila, proporcionar una petición SQL (entre paréntesis) que retorne los valores de una o varias nuevas filas.
 3. En el comando que crea una o varias nuevas filas, si no se proporcionan valores para todas las columnas, cuando el comando se ejecuta se asignan valores por defecto a las columnas sin valor asignado (`null`, si el esquema no contiene la especificación de un valor por defecto para la columna en cuestión). Se puede utilizar una "especificación de columnas" – una lista de nombres de columnas, separados por comas, entre corchetes, situada justo antes de los valores de la fila (o justo antes de la petición que retorna los valores de una o varias filas, véase la pista previa) – para evitar la ambigüedad respecto a cuales de las columnas se quiere dar valores explícitamente.
 4. Los elementos separados por comas que siguen a la palabra clave `SELECT` pueden ser *value expressions* de SQL (es decir, no es obligatorio que sean nombres de columnas).
- (ii) **(0,15 puntos)** Se vende un artículo cuyo `Codigo_producto` es 351224; proporcione un comando que implemente la correspondiente modificación a la tabla `Ventas` de la base de datos.
- (iii) **(0,1 puntos)** Elimine de la tabla `Ventas` todas las entradas de 2005 cuyo precio sea menor de 10 Euros.

- (b) **(0,75 puntos)** La Figura 1 contiene un diagrama UML que representa el modelo del dominio de un departamento de una universidad española. Estudie dicho modelo y conteste a las siguientes preguntas:

- (i) **(0,25 puntos)** ¿Qué tipo de diagrama UML tiene delante y a qué categoría de diagramas pertenece? ¿Qué representa la caja etiquetada como `Empleado`? ¿Qué representa el texto contenido en esa caja? ¿Qué otra información podría haber sido proporcionada en las cajas?
- (ii) **(0,25 puntos)** ¿Cuál es el significado de la línea etiquetada `Es autor` que conecta las cajas `Publicación` y `Empleado`? ¿Cuál es el significado del texto `autores Locales` localizado en el extremo `Empleado` de esta línea (esto es, donde esta línea se encuentra con la caja etiquetada `Empleado`)? ¿Cuál es el significado de la anotación `*` que aparece

Examen de Software de Comunicaciones

5º Ingeniero de Telecomunicación

26 de enero de 2010

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

Nombre: _____

en el extremo `Publicación` y de la anotación `1..*` que aparece en el extremo `Empleado` de esta línea? ¿Qué significa la ausencia de una tal anotación en un extremo de una línea que conecta dos cajas del diagrama (p.e. el extremo `Empleado` de la línea etiquetada `Es responsable` que conecta `Publicación` y `Empleado`)?

- (iii) **(0,25 puntos)** ¿Cuál es el significado de la línea que conecta las cajas etiquetas `Grupo_Investigación` y `Línea_Investigación` (no olvide tomar en consideración la forma de diamante y explicar lo que significa en términos prácticos)? ¿Cuál es el significado de las líneas que conectan la caja `Empleado_Universidad` y las cajas etiquetadas `PDI` y `PAS`?

Nombre: _____

Problema 2 (2 puntos)

Estudie el siguiente código conforme a la especificación EJB3 (de Bill Burke y Richard Monson-Haefel) y conteste a las preguntas que aparecen a continuación.

En el fichero `TravelAgentRemote.java`:

```
package com.titan.travelagent;
import com.titan.processpayment.CreditCardDO;
import javax.ejb.Remote;
import com.titan.domain.Customer;

@Remote
public interface TravelAgentRemote {
    public Customer findOrCreateCustomer(String first, String last);
    public void updateAddress(Address addr);
    public void setCruiseID(int cruise);
    public void setCabinID(int cabin);
    public TicketDO bookPassage(CreditCardDO card, double price)
        throws IncompleteConversationalState;
}
```

En el fichero `TravelAgentBean.java`:

```
package com.titan.travelagent;
import com.titan.processpayment.*;
import com.titan.domain.*;
import javax.ejb.*;
import javax.persistence.*;
import javax.annotation.EJB;
import java.util.Date;

@Stateful
public class TravelAgentBean implements TravelAgentRemote {

    @PersistenceContext(unitName="titan")
    private EntityManager entityManager;

    @EJB private ProcessPaymentLocal processPayment;
    private Customer customer;
    private Cruise cruise;
    private Cabin cabin;

    public Customer findOrCreateCustomer(String first, String last) {
        try {
            Query q = entityManager.createQuery(
                "from Customer c where c.firstName = :first"
                + " and c.lastName = :last");
            q.setParameter("first", first);
            q.setParameter("last", last);
            this.customer = (Customer) q.getSingleResult();
        } catch (NoResultException notFound) {
            this.customer = new Customer();
            this.customer.setFirstName(first);
            this.customer.setLastName(last);
            entityManager.persist(this.customer);
        }
        return this.customer;
    }
}
```

Nombre: _____

```
public void updateAddress(Address addr) {
    this.customer.setAddress(addr);
    this.customer = entityManager.merge(customer);
}

public void setCabinID(int cabinID) {
    this.cabin = entityManager.find(Cabin.class, cabinID);
    if (cabin == null) throw new NoResultException("Cabin not found");
}

public void setCruiseID(int cruiseID) {
    this.cruise = entityManager.find(Cruise.class, cruiseID);

    if (cruise == null) throw new NoResultException("Cruise not found");
}

@Remove
public TicketDO bookPassage(CreditCardDO card, double price)
    throws IncompleteConversationalState {
    if (customer == null || cruise == null || cabin == null)
    {
        throw new IncompleteConversationalState();
    }
    try {
        Reservation reservation = new Reservation(
            customer, cruise, cabin, price, new Date());
        entityManager.persist(reservation);
        processPayment.byCredit(customer, card, price);
        TicketDO ticket = new TicketDO(customer, cruise, cabin, price);
        return ticket;
    } catch (Exception e) {
        throw new EJBException(e);
    }
}
}
```

En el fichero Customer.java:

```
package com.titan.domain;
import javax.persistence.*;

@Entity
@Table(name="CUSTOMER_TABLE")
public class Customer implements java.io.Serializable {

    private long id;
    private String firstName;
    private String lastName;

    @Id
    @Column(name="CUST_ID", nullable=false, columnDefinition="integer")
    public long getId() { return id; }
    public void setId(long id) { this.id = id; }

    @Column(name="FIRST_NAME", length=20, nullable=false)
    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }
    public void setLastName(String lastName) { this.lastName = lastName; }
}
```

Nombre: _____

- (a) **(0.7 puntos)** Explique brevemente el uso de cada una de las siguientes anotaciones:
- (i) `@Remote` (véase la interfaz `TravelAgentRemote`)
 - (ii) `@Stateful`, `@PersistenceContext`, `@EJB` and `@Remove` (véase la clase `TravelAgentBean`)
 - (iii) `@Entity`, `@Id` (véase la clase `Customer`)
- (b) **(0.4 puntos)** Explique brevemente el papel que desempeña la clase `EntityManager` y el uso de cada uno de los siguientes métodos de esa clase: `createQuery()`, `persist()`, `merge()` and `find()`.
- (c) **(0.4 puntos)** Suponga que no vamos a crear las tablas de la base de datos a partir del código Java (ni vice versa). Proporcione el código SQL para crear la tabla de la base de datos correspondiente a la clase `Customer`.
- (d) **(0.4 puntos)** Si utilizáramos el código siguiente para la clase `Customer` en vez de el que aparece anteriormente, ¿cómo sería ahora el código SQL para crear la tabla de la base de datos correspondiente a esa clase?

```
@Entity
public class Customer implements java.io.Serializable {

    @Id
    private long id;
    private String firstName;
    private String lastName;

    public long getId() { return id; }
    public void setId(long id) { this.id = id; }

    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) {this.firstName = firstName;}

    public String getLastName() { return lastName; }
    public void setLastName(String lastName) {this.lastName = lastName;}
}
```

- (e) **(0,1 point)** ¿Qué anotación utilizaría si quisiera que fueran generados automáticamente los valores de la clave primaria de la tabla de la base de datos que corresponde a la clase `Customer`?

Nombre: _____

Problema 3 (2 puntos)

Estudie el siguiente código para el método `service` de un servlet Java y conteste a las preguntas que aparecen a continuación. Dese cuenta de que al método `getProductItem` se le pasa una lista de los productos y un identificador de producto y que retorna el producto de la lista que tiene ese identificador o `null` si dicho producto no está en la lista.

```
public void service(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    List<Product> products = null;
    try {
        HttpSession session = object1.method1(false);
        if (session == null) {
            object2.method3(object2.encodeRedirectURL("/shop/login"));
            return;
        }
        Cart cart = (Cart) object3.method2("cart");
        if (cart == null) {
            object2.method3(object2.encodeRedirectURL("/shop/login"));
            return;
        }
        String productIDToAdd = object1.method4("Buy");
        String productId = (String) object1.method4("productId");
        products = (Vector) object3.method2("products");

        if (productIDToAdd != null) {
            Product item = getProductItem(products, productIDToAdd);
            cart.add(new LineItem(item, 1, 0));
            object1.method5("ProductPurchased", item.getName());
            this.getServletContext().getRequestDispatcher(
                "/catalog.jsp").method6(request, response);
            return;
        }

        else if (productId != null) {
            object1.method5("productItem", getProductItem(products, productId));
            this.getServletContext().getRequestDispatcher(
                "/productInfo.jsp").method6(request, response);
            return;
        }

        else {
            if (products == null) {
                products = catalog.getProductList();
                object3.method7("products", products);
            }
            this.getServletContext().getRequestDispatcher(
                "/catalog.jsp").method6(request, response);
            return;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Nombre: _____

- (a) **(0,3 puntos)** ¿Cuáles son los roles de los objetos `request` y `response` (esto es, los objetos de la clase `HttpServletRequest` y `HttpServletResponse`) que aparecen como argumentos del método `service`?
- (b) **(0,1 puntos)** La manera en que está implementado este servlet – sobrescribiendo el método `service` – no es la manera habitual de implementar un servlet. ¿Cuál es la manera más usual?
- (c) **(1,0 puntos)** Proporcione los nombres de `object1`, `object2` y `object3`, y de los métodos `method1`, `method2`, `method3`, `method4`, `method5`, `method6`, y `method7`. Si no puede recordar alguno, explique brevemente lo que hace.
- (d) **(0,4 puntos)** Explique brevemente la diferencia entre `method3` y `method6`. Para cada una de las tres llamadas a `method6`, exponga cómo se vería afectada la funcionalidad si la llamada a este método del objeto `requestDispatcher` fuese reemplazada por una llamada a `object2.method3`.
- (e) **(0,2 puntos)** Hasta donde se puede saber, explique si la aplicación de que este código forma parte sigue el patrón MVC. Debe justificar su respuesta (de forma muy breve).