EXAMEN IS2 (26 - MAYO - 2010)

1. Utiliza el método que consideres oportuno para diseñar un conjunto de casos de prueba para el método VentasBO.generaTicket que se muestra a continuación. Es imprescindible que indiques los pasos que has seguido. Explica cuál es el objetivo del método elegido. (2p)

```
public class VentasBO{

        //en la lista de artículos puede aparecer el mismo código repetido varias veces

        public TicketTO generaTicket(ClienteTO cliente, List<String> codArticulos){

                //comprueba si el cliente puede realizar la compra

                if(cliente==null || cliente.getNif() == null ||

                (cliente.getEstado()==EstadoCliente.moroso && cliente.getDeuda()>1000)){

                        throw new BOException("El cliente no puede realizar la compra");

                }

                //crea un diccionario indexado por el código de los artículos.
                //a cada código de artículo se le asociara una línea de venta

                Map <String,LineaVentaTO> lineasTicket =

                                                new HashMap<String,lineaVentaTO>();

                float precioTotal = 0.0f;

                for(string cod: codArticulos){

                        //busca el artículo en el diccionario

                        LineaTicketTO linea = lineasTicket.get(cod);

                        if(linea==null){

                                //si todavía no ha sido insertado en el diccionario, crea
                                //una nueva línea de venta

                                linea = new LineaTicketTO();

                                //recupera datos del artículo de la BD

                                IArticuloDAO adao = this.getArticuloDAO();

                                ArticuloTO articulo = null;

                                try{

                                        articulo = adao.getArticulo(cod);

                                }catch(DAOException e){

                                        throw new BOException(

                                        "Error al recuperar datos de los artículos");

                                }

                                if(articulo==null){

                                throw new BOException("El artículo no esta en la BD");

                                }
```

```
                        //inserta los datos de la linea en el diccionario
                        linea.setArticulo(articulo);
                        linea.setUnidades(1);
                        lineasTicket.put(cod,linea);
                        precioTotal += articulo.getPrecioUnitario();
                    }
                    else{
                        //si el articulo ya estaba en el diccionario, incrementa el
                        //número de unidades y el precio de venta
                        linea.setUnidades(linea.getUnidades() + 1);
                        linea.setPrecioLinea(linea.getPrecioLinea() +
                                             linea.getArticulo.getPrecioUnitario());
                        precioTotal += linea.getArticulo().getPrecioUnitario();
                    }//end if
                }//end for
                TicketTO ticket = new TicketTO();
                ticket.setCliente(cliente);
                ticket.setLineas(lineasTicket.allValues());
                ticket.setPrecioTotal(precioTotal);
                return ticket;
            }
            public IArticuloDAO getArticuloDAO(){
                return new JDBCArticuloDAO();
            }
    }
```

A modo de ayuda, a continuación, se muestran las estructuras de datos utilizadas en el método a probar:

```
    public class clienteTO{
            string nif;
            EstadoCliente estado;
            float deuda;
            public ClienteTO(String nif, EstadoCliente estado, float deuda){
                ...
            }
            enumEstadoCliente{normal,moroso};
    }
```

```
    public class ArticuloTO{
              String cod;
              float precioUnitario;
              public ArticuloTO(String cod, float precioUnitario){
              ...
              }
    }
    public class TicketTO{
              ClienteTO cliente;
              List<LineaVentaTO> lineas;
              float precioTotal;
              ...
    }
    public class LineaVentaTO{
              ArticuloTO articulo;
              int unidades;
              float precioLinea;
              ...
    }
    public interface IArticuloDAO{
              public ArticuloTO getArticulo(String cod);
}
```

2. Teniendo en cuenta el código del ejercicio anterior. Implementa, utilizando JUnit lo siguiente: un caso de prueba en el que se produzca la excepción "Error al recuperar datos del artículo", otro con la excepción "El artículo no está en la BD", y otro en el que nos devuelva un objeto TicketTO (con comprobar el precioTotal es suficiente). En el caso de las excepciones no es necesario comprobar el mensaje que contiene. Indica el directorio de fuentes en el que se ubicará cada clase. Implementa todo el código necesario para que las pruebas puedan ejecutarse (2p)

3. Responde a las siguientes preguntas (2p);

(a) ¿Un modelo en cascada puede ser cascada con carácter ágil?_____ Explica por qué. En caso afirmativo pon un ejemplo concreto que ilustre tú respuesta.

(b) Explica una razón concreta por la que podemos necesitar dos ciclos en UP.

(c) Dada una tarea T1 de un proyecto P, si ACWP > BCWP para dicha tarea, ¿podemos afirmar que el proyecto está gastando más de lo planificado? Explica por qué.

(d) Para un mismo proyecto, desarrollado por dos equipos de la gestores, ¿podría existir dos conos de incertidumbre distintos asociados? Explica por qué y pon un ejemplo con el que ilustres tu respuesta.

4. Pon un ejemplo concreto de especificación que contenga 3EIS, 2EO y 1EIF. Calcula el valor de los elementos necesarios para poder consultar en las tablas correspondientes el número de puntos de función no ajustados asociados. Suponiendo que en una aplicación obtenemos finalmente un valor de X puntos de función, explica (enumerando los pasos a seguir) como podríamos, a partir de dicho valor, estimar la productividad de dicho proyecto. (2 p)

5. Supongamos que estamos trabajando en un proyecto con 3 desarrolladores (d1,d2 y d3) y que se va a seguir un modelo de proceso UP. Escribe los comandos CVS para la fase de inicio y la primera iteración de elaboración, pensando que se va a seguir el WBS propuesto por Craig Larman. Detalla dicho WBS (indicando, además las precedencias entre las actividades) y relaciona cada una de las actividades resultantes con los comandos CVS (en caso de que proceda) e indica los EC's afectados, así como las operaciones a realizar con dichos EC's (creación/ modificación). Para responder a esta pregunta debes seguir el formato de la siguiente tabla. (2p)

WBS	**CVS**	**EC**