Hoja de Ejercicios

Diagramas UML

1. Para el documento de análisis de requisitos de la aplicación que se describe a continuación, realiza un **diagrama de casos de uso**, así como la descripción detallada de uno los casos de uso más relevantes para esta aplicación.

   Se desea tener una aplicación para gestionar las reservas de mesa en una cadena de restaurantes, así como la entrada y salida de los comensales en el restaurante. Los comensales pueden encargarse personalmente de realizar su reserva de mesa o puede que se encargue de ello el maître del restaurante en algunas circunstancias (por ejemplo, si una persona llega al puesto de recepción sin haber hecho reserva previa y pretende solicitar mesa en ese instante−incluso en esa situación es necesario que se haga una reserva en el sistema antes de asignar mesas a los comensales). Sea quien sea quien se encargue de hacer la reserva de mesa, se podrá hacer a través de un sistema interactivo accesible en la Web o a través de la centralita telefónica de reservas de esta cadena de restaurantes (incluso el maître si, por ejemplo, la conexión Web fallase en el momento de una entrada sin reserva previa). Realizar la reserva implica que se debe consultar si hay mesas disponibles para el número de comensales solicitado, aunque la asignación de mesas concretas no se haga hasta el momento de la entrada al restaurante. Cuando los comensales con reserva previa llegan al restaurante, el maître realiza el proceso registro y entrada al restaurante, lo cual incluye, entre otras cosas, localizar la reserva y consultar las mesas disponibles para asignar en ese momento mesa a los comensales. En el momento en que los comensales deseen abandonar el restaurante, el maître debe volver a localizar la reserva, para obtener de ella los datos que le permitan calcular y emitir la factura correspondiente.

2. En una aplicación de ventas on-line especificar el **diagrama de secuencia** de la acción *comprar* definida en una clase *Carrito* asumiendo las siguientes características de la aplicación:

   a. En la clase *Carrito* se almacena un pedido web representado como un listado con todos los productos (clase *Producto*) que el usuario ha añadido a su carrito, así como las cantidades de cada uno de ellos.

   b. Antes de realizar la compra se comprobará que en el almacén (único) que da servicio a la aplicación web (clase *Almacen*) hay stock suficiente de cada uno de los productos incluidos en el carrito.

   c. En caso de no haber stock suficiente, el sistema se conectará a un web service (clase *Proveedor*) que permitirá realizar una petición de productos para el cliente según su código de cliente.

   d. En caso que el pedido al proveedor no se pueda realizar correctamente, la transacción finalizará mostrando un mensaje de error al usuario (no siendo necesario cancelar las peticiones realizadas previamente al proveedor).

   e. Tanto si hay stock suficiente para realizar el pedido, como si la petición indicada en el paso anterior se realiza correctamente, se realizará el cobro del

pedido utilizando un TPV virtual (clase *PasarelaPago*) utilizando los datos almacenados en el objeto *Usuario* que representa al usuario de la web que realiza la compra (almacenado en el *Carrito* correspondiente).

f. Tras el cobro, el pedido se enviará al almacén (clase *Almacen*) para proceder a su preparación.

3. Especificar el diagrama de secuencia del método solucionar:

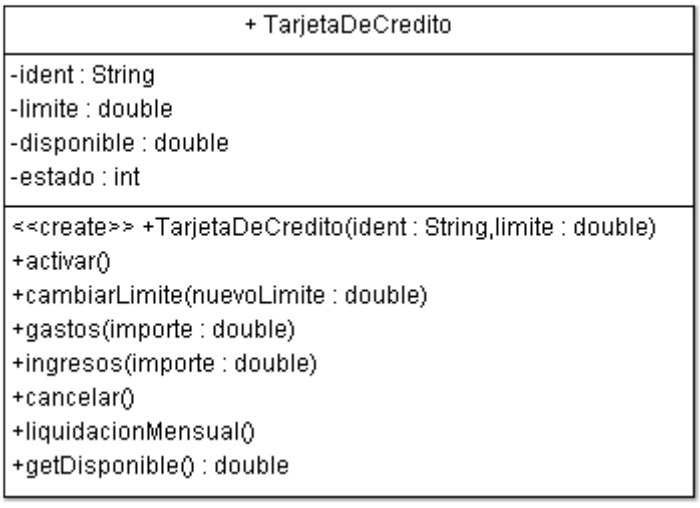
```
public class Puzzle {
   private Fragmentos piezas;

   void solucionar()
   {
          Pieza p1;
          Pieza p2;
          Pieza p3;

          while(piezas.length > 1)
          {
                 p1 = piezas.seleccionar();
                 p2 = piezas.seleccionar();

                 if(p1.match(p2))
                 {
                        p3 = new Pieza(p1, p2);
                        piezas.eliminar(p1);
                        piezas.eliminar(p2);
                        piezas.añadir(p3);
                 }
          }
    }
   }
```

4. Realiza y comenta el **diagrama de transición de estados** para modelar el comportamiento de los objetos de la clase TarjetaDeCredito descrita a continuación:



Una tarjeta de crédito se crea con un identificador y un límite máximo de gasto, y se envía a su titular desactivada de forma que no se puede realizar ninguna operación con ella excepto activarla (o cancelarla). Cuando el titular activa la tarjeta, se puede empezar a utilizar con una cantidad disponible igual al límite máximo. Al realizar gastos (compras o extracciones en cajero) se disminuye el disponible, y al realizar ingresos (devoluciones de compras o trasferencias de efectivo) se aumenta el disponible. Si un gasto deja el disponible en negativo, la tarjeta queda pendiente de un o varios ingresos que vuelvan a generar un disponible positivo que permita continuar con el uso normal de la tarjeta. En cambio, si una tarjeta está pendiente de un ingreso y se genera otro gasto, la tarjeta quedará bloqueada. Una tarjeta bloqueada no podrá ser utiliza hasta después de la siguiente liquidación mensual. La liquidación mensual de la tarjeta restaura su disponible al límite máximo actual de la tarjeta. El límite máximo inicial de la tarjeta puede cambiar (ampliarse o reducirse) inmediatamente después de una liquidación mensual pero antes de realizarse el primer gasto del siguiente mes. La tarjeta sólo podrá sea cancelada cuando el disponible coincida con el límite máximo y una vez cancelada no podrá ser reactivada.

5. Realiza y comenta el **diagrama de transición de estados** para modelar el comportamiento de los objetos de la clase Despertador descrita a continuación:

Cuando un despertador se conecta empieza a funcionar con la alarma desactivada y sin hora de alarma establecida. Cuando el despertador está conectado, el usuario puede establecer o cambiar la hora de alarma, a menos que la señal de alarma esté disparada. Desde que se fije la hora de alarma el despertador tendrá la alarma activa y comprobará constantemente si llega la hora de disparar la alarma. El sistema invoca automática y periódicamente (al menos, cada segundo) al método que actualiza la hora para que el despertador lleve control del paso del tiempo. A menos que se haya desactivado la alarma antes, cuando llegue la hora de la alarma el despertador producirá la señal (acústica, luminosa, o musical) correspondiente según el modo en que se encuentre, y esperará a que el usuario detenga la alarma. Inicialmente estará en modo alarma acústica, aunque tiene otros dos modos de trabajo (señal luminosa y musical), que indican el tipo de actividad que se generará cuando llegue la hora de disparar la alarma. El usuario puede cambiar el modo del despertador en cualquier momento excepto cuando la alarma está disparada y produciendo la señal

correspondiente. Cuando el usuario detenga la alarma, el despertador seguirá en el mismo modo de trabajo y con la misma hora de alarma fijada y activada hasta que se vuelva a disparar la alarma al día siguiente si nada cambia. Si el despertador se desconecta, se desactiva automáticamente su alarma (si estaba activada) y pierde su memoria de hora de alarma (si es que la tenía). Cuando se vuelva a conectar el despertador, vuele al modo alarma acústica sin hora de alarma.

6. En una aplicación de ventas on-line se utiliza una clase CarritoCompra, descrita a continuación. Realiza y comenta el **diagrama de transición de estados** para modelar el comportamiento de los objetos de dicha clase.

   El carrito se crea vacio y sin dirección envío asignada. Cuando se añade algún producto, empieza a llevar control de cuantos productos contiene. Mientras contenga productos podrá modificar la cantidad comprada de cualquiera de ellos, o se podrá sacar el producto del carrito. Independientemente de si el carrito está vacío o no, se podrá asignar una dirección de envío. Un carrito con dirección asignada y productos podrá cerrarse, momento en el cual se calcula el importe total con impuesto y gastos de envío. Un carrito cerrado puede reabrirse para seguir modificando los productos que contiene o su dirección envío. Un carrito cerrado puede pagarse y si el pago ha sido completado satisfactoriamente pasará constar como enviado. Hasta que una vez recibido, el pedido se da por terminado definitivamente.