Introducción al Software - EXAMEN DE PRÁCTICAS SEPTIEMBRE 2016

Se desea crear una aplicación que permita gestionar un sitio de subastas online, en el que los usuarios pueden ofrecer a la venta cualquier producto, indicando el precio de salida y la fecha de finalización de la subasta. A partir de ese instante, cualquier otro usuario puede realizar una puja para adquirir el producto. El usuario que haga la puja más alta al llegar el momento de finalización de la subasta, es quien compra el producto.

La aplicación se organizará en las siguientes clases:

• Puja: contiene la información de cada puja que realiza un usuario para ganar una subasta.

• Subasta: contiene la información de cada subasta que se publica en el sitio, con la lista de las pujas que se han realizado sobre cada una de ellas y métodos para gestionarla.

La clase Puja se da ya hecha, mientras que de la clase Subasta se da un esqueleto que hay que completar

Clase Puja

Es una clase sencilla con el siguiente diagrama de clase:

Puja
- String usuario - double precioOferta - Calendar fecha
+ Puja(String usuario, double precioOferta) + String usuario() + double precioOferta() + Calendar fecha()

• tres atributos: nombre del usuario que realiza la puja, el precio que se oferta en de la puja y el momento en que se hace la puja.

• constructor que recibe como parámetros el usuario y el valor de la puja

• un método observador para cada atributo

Clase Subasta

Clase parcialmente realizada cuyo diagrama de clase es el siguiente:

Subasta
- String usuario - String producto - double precioSalida - Calendar fechaFin - ArrayList<Puja> listaPujas
+ Subasta(int id, String usuario, String producto, double precioSalida, int anyoFin, int mesFin, int diaFin, int horaFin, int minFin) + boolean pujar(Puja puja)

+ String usuarioLider() + Puja[] pujasMayores(double precio) + Puja ultimaPujaUsuario(String usuario)

Tiene 5 atributos: el nombre del usuario que publica la subasta, la descripción del producto a subastar, el precio de salida del mismo, la fecha de finalización de la subasta y un ArrayList, que llamaremos listaPujas, para guardar todas las pujas realizadas por los distintos usuarios.

- Constructor: Crea cada subasta a partir de los parámetros recibidos. Para crear la fecha de finalización, se reciben por separado los parámetros relativos al año, mes, día, hora y minuto. Además, se debe inicializar la lista de pujas, dejándola inicialmente vacía.

- pujar(): Este método recibe una nueva puja como parámetro y la añade a la lista de pujas de la subasta. Para que la puja sea válida y pueda registrarse se deben cumplir los siguientes supuestos:
  - Que el usuario de la subasta no sea el mismo que hace la puja.
  - Que la fecha de la puja sea anterior a la fecha de finalización de la subasta.
  - Que el precio ofertado en la puja sea mayor que el precio de la puja más alta. En caso de que no haya aun pujas, el precio de la oferta debe ser mayor que el precio base de la puja.

  Los casos deben comprobarse en el orden especificado. En caso de que alguno de los casos no se cumpla, el método devuelve falso y la puja no se añade a la lista. Si todos los supuestos se cumplen, la puja se añade a la lista de pujas y el método devuelve verdadero.

- usuarioLider(): método que devuelve el usuario que va ganando la subasta. En caso de que no haya pujas, se devuelve null.

- pujasMayores(): método que devuelve un array de pujas con todas las pujas mayores a un precio dado como parámetro. Si no hubiera pujas, se devuelve null.

- ultimaPujaUsuario(): método que devuelve la última puja realizada por un usuario concreto que se pasa como parámetro. Si no existiese tal puja, se devuelve null.

Además, para probar la clase, se pide realizar un programa principal que realice las siguientes acciones:

- Crear un objeto de la clase Subasta con unos datos cualesquiera. Realizar diversas pujas, que prueben cada uno de los supuestos para que la puja sea válida. Mostrar por pantalla si las pujas son o no válidas.
- Mostrar por pantalla el usuario que es líder después de cada puja que se haga.
- Probar los métodos pujasMayores y ultimaPujaUsuario mostrando los resultados por pantalla.

Una vez realizado debe incluirse en el informe una captura de pantalla donde se vea claramente el funcionamiento de los métodos solicitados.

Nota: Se recomienda ir haciendo el programa principal a medida que se tengan métodos de la clase Subasta hechos, de forma que se pueda ir obteniendo resultados parciales.

Valoración

Para la valoración, al tratarse de un examen de prácticas, es imprescindible que el código entregado compile correctamente y que los métodos realicen la funcionalidad requerida.

- Constructor: 1 punto
- Método pujar: 2 puntos
- Método usuarioLider: 1 punto
- Método pujasMayores: 1,5 puntos
- Método ultimaPujaUsuario: 1,5 puntos
- Programa principal y captura de pantalla: 3 puntos

Anexo. Manejo de fechas con Calendar

Para el manejo de las fechas de la práctica se ha decidido utilizar la clase Calendar, que nos permite representar y manejar fechas, así como compararlas entre sí. La clase dispone de multitud de métodos, de los cuales os indicamos los que pueden ser de mayor interés para la práctica:

- public static Calendar **getInstance**()

Método estático de la clase Calendar que devuelve un objeto concreto de la misma clase. La fecha que contiene el objeto es la fecha actual.

- public final void **set**(int year, int month, int day, int hourOfDay, int minute)

Actualiza la fecha que guarda el objeto, usando los valores que se pasan como parámetros. El método recibe el año, el mes (Enero es el mes 0 y Diciembre el mes 11), el día, la hora y el minuto que deseamos guardar.

- public boolean **after**(Calendar otro)

Método que devuelve verdadero si la fecha almacenada en el objeto es posterior a la fecha almacenada en el parámetro “otro”. En caso contrario devuelve falso.

- public boolean **before**(Calendar otro)

Método que devuelve verdadero si la fecha almacenada en el objeto es anterior a la fecha almacenada en el parámetro “otro”. En caso contrario devuelve falso.

- public int **compareTo**(Calendar otro)

Método que devuelve 0 si la fecha almacenada en el objeto es igual a la fecha almacenada en el parámetro “otro”; un valor menor que 0 si es anterior y un valor mayor que cero si es posterior.