PRÁCTICA A001_2019.- Eventos

Esta práctica va a **gestionar eventos** realizados en una sala que contiene dos tipos de butacas: gold y silver. El precio de la entrada al evento será diferente para cada tipo de butaca.

Las butacas de ambos tipos están numeradas de 1 a n.

Cuando un usuario compra una entrada, elige el número de butaca que quiere de las disponibles (que no estén ocupadas). Esta butaca (Seat) se almacenará en la posición correspondiente del array en la posición pos -1 (ya que en Java los arrays empiezan en 0 y los números de butaca empiezan a numerarse en 1).

PASOS PARA LA REALIZACIÓN DE LA PRÁCTICA:

1. Descargar el proyecto de la página de la asignatura en agora.unileon.es:
   - edi-a001-2019.zip

2. Importar el proyecto en Eclipse : Import… Existing proyects into Workspace… Select archive file…

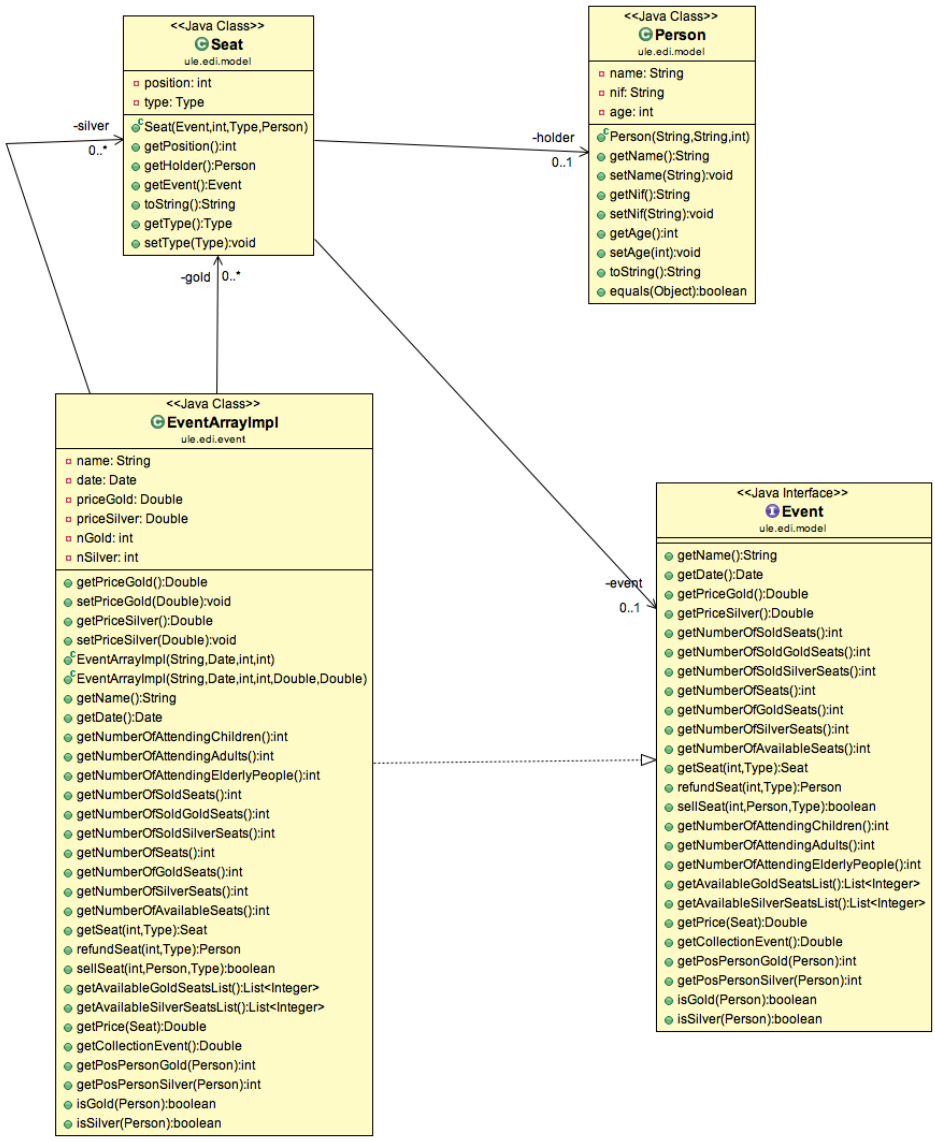
   (indicar el archivo ZIP descargado)

Los proyectos normalmente contendrán la estructura a respetar (e incluso pueden tener errores de compilación ya que todo el trabajo está por hacer).

3. Las clases del paquete **ule.edi.model NO DEBEN MODIFICARSE**.
   En este paquete SOLAMENTE HAY QUE IMPLEMENTAR el método equals de la clase Person.
   Las clases que contiene este paquete son:
   - **Configuration**: Define constantes y un tipo enumerado.
   - **Person** : Define una persona
   - **Seat**: Define una butaca ocupada, indicando la posición en el evento, la persona que la ocupa, el tipo de butaca que es (gold o silver) y el evento al que pertenece
   - **Event** : interface que define las operaciones a implementar por la clase EventArrayImpl

4. El paquete **ule.edi.event** contiene dos ficheros java **QUE DEBEN SER COMPLETADOS (NO PUEDE MODIFICARSE LA ESTRUCTURA DE DATOS)**
   - **EventArrayImpl**: Clase que **deberá implementar** un evento con arrays. Dispone de dos arrays diferentes, uno para cada tipo de butacas.
   - **EventArrayImplTests**: Clase que **deberá contener** todos los tests de prueba necesarios para probar todos los métodos de la clase EventArrayImpl. **SE UTILIZARÁ JUNIT4**.

ESTRUCTURAS DE DATOS UTILIZADAS:



OPERACIONES DEL INTERFACE Event:

```
package ule.edi.model;

import java.util.Date;
import java.util.List;

import ule.edi.model.Configuration.Type;

public interface Event {


	public String getName();

	public Date getDate();

	public Double getPriceGold();

	public Double getPriceSilver();


	/**
	 * Calcula el número de butacas totales vendidas del evento.
	 */
	public int getNumberOfSoldSeats();
```

```java
/**
 * Calcula el número de butacas Gold vendidas del evento.
 */
public int getNumberOfSoldGoldSeats();

/**
 * Calcula el número de butacas Silver vendidas del evento.
 */
public int getNumberOfSoldSilverSeats();

/**
 * Número de butacas totales del evento (ocupadas y disponibles).
 */
public int getNumberOfSeats();

/**
 * Número de butacas GOLD totales del evento (ocupadas y disponibles).
 */
public int getNumberOfGoldSeats();

/**
 * Número de butacas SILVER totales del evento (ocupadas y disponibles).
 */
public int getNumberOfSilverSeats();


/**
 * Calcula el número de butacas disponibles (no vendidas).
 */
public int getNumberOfAvailableSeats();

/**
 * Devuelve la butaca en la posición dada y del tipo dado o null si no está ocupada
 *
 * Las posiciones empiezan en '1'.
 *
 * @param pos
 * @return
 */
public Seat getSeat(int pos, Type type);


/**
 * Libera la butaca de la posición dada y del tipo dado.
 *
 * Si la butaca de esa posición ya está libre, devuelve null.
 *
 * @param pos
 * @return p :la persona que ocupaba la butaca o null si la butaca no estaba ocupada.
 *
 */
public Person refundSeat(int pos, Type type);

/**
 *
 * Si la butaca de esa posición ya está ocupada, no hace nada.
 *
 * Las posiciones empiezan en '1'.
 *
 * @param pos
 * @param p : la persona que ocupará la butaca
 * @return true indica si se pudo realizar la venta de la butaca, false en caso contrario
 */
public boolean sellSeat(int pos, Person p, Type type);


/**
 * Calcula el número de niños asistentes al evento.
 *
 *        [0, Configuration.CHILDREN_EXMAX_AGE)
 *
 * CHILDREN_EXMAX_AGE no contabiliza como niño (menor que)
 *
 * @return
 */
public int getNumberOfAttendingChildren();

/**
 * Calcula el número de adultos asistentes al evento.
 *
 *        [Configuration.CHILDREN_EXMAX_AGE, Configuration.ELDERLY_PERSON_INMIN_AGE)
 *
 * ELDERLY_PERSON_INMIN_AGE no incluido como adulto (menor que)
 *
 * @return
```

```
         */
        public int getNumberOfAttendingAdults();

        /**
         * Calcula el número de ancianos asistentes al evento.
         *
         *       [Configuration.ELDERLY_PERSON_INMIN_AGE, Integer.MAX_VALUE)
         *
         * ELDERLY_PERSON_INMIN_AGE incluido como anciano
         *
         * @return
         */
        public int getNumberOfAttendingElderlyPeople();


        /**
         * Calcula la lista de números de butacas gold disponibles
         * Tener en cuenta que las posiciones empiezan en 1
         *
         * @return lista de posiciones disponibles
         */
        public List<Integer> getAvailableGoldSeatsList();


        /**
         * Calcula la lista de números de butacas silver disponibles
         * Tener en cuenta que las posiciones empiezan en 1
         *
         *        @return lista de posiciones disponibles
         */
         public List<Integer> getAvailableSilverSeatsList();

         /**
         * Calcula el precio de la butaca en función del tipo de butaca y del precio de ese tipo de butaca para el
         * evento al que pertenece
         *
         *
         *        @return lista de posiciones disponibles
         */
        public Double getPrice(Seat seat);


        /**
         * Calcula el importe total recaudado por las butacas ocupadas
         *
         *        @return importe total recaudado
         */
        public Double getCollectionEvent();

        /**
         *
         * @param p persona a buscar en la parte GOLD
         * @return la butaca que ocupa la persona o -1 si no está
         */
        public int getPosPersonGold(Person p);

        /**
         *
         * @param p persona a buscar en la parte SILVER
         * @return la butaca que ocupa la persona o -1 si no está
         */
        public int getPosPersonSilver(Person p);

        /**
         *
         * @param p persona a buscar en la parte GOLD
         * @return true si la persona ocupa una butaca, false en caso contrario
         */
        public boolean isGold(Person p);

        /**
         *
         * @param p persona a buscar en la parte SILVER
         * @return true si la persona ocupa una butaca, false en caso contrario
         */
        public boolean isSilver(Person p);
}
```

ATRIBUTOS DE LA CLASE EventArrayImplINTERFACE Event:

```
        private String name;
        private Date date;

        private Double priceGold;    // precio de entradas tipo GOLD
```

```
        private Double priceSilver;  // precio de entradas tipo SILVER

        private int nGold;          // Nº de butacas de tipo GOLD
        private int nSilver;        // Nº de butacas de tipo GOLD

        private Seat[] gold;    // Array de butacas de tipo GOLD
        private Seat[] silver;  // Array de butacas de tipo SILVER
```

CONSIDERACIONES PARA LA IMPLEMENTACIÓN DE LA PRÁCTICA:

- El constructor de la clase EventArrayImpl **debe crear los arrays de Seat gold** y **silver**, pero **no debe crear los Seats de cada posición del array**. Estos objetos Seat se crearán cuando se venda una entrada para esa butaca con el método **sellSeat(int pos, Person p, Type type).**
- Es importante recordar que los arrays empiezan en 0 pero la numeración de las butacas empiezan en 1, por lo que el Seat correspondiente a la butaca n estará almacenada en el array en la posición n-1.
- En el método **sellSeat(int pos, Person p, Type type)**, el primer parámetro es la posición o número de butaca, que indicará por tanto la posición en el array donde debe insertarse (en la pos-1).
- Cualquier método que reciba como parámetro la posición o nº de butaca debe comprobar que es un valor válido para que **no provoque errores** en el programa.
- **IMPORTANTE**: Utilizar JUNIT 4.
- Se valorará la cobertura total de los test implementados, por lo que debe usarse un plugin de Eclipse para comprobar la cobertura de los test generados.

FECHA LIMITE de entrega de la práctica A001-2019: 3 de Marzo de 2019 23:55