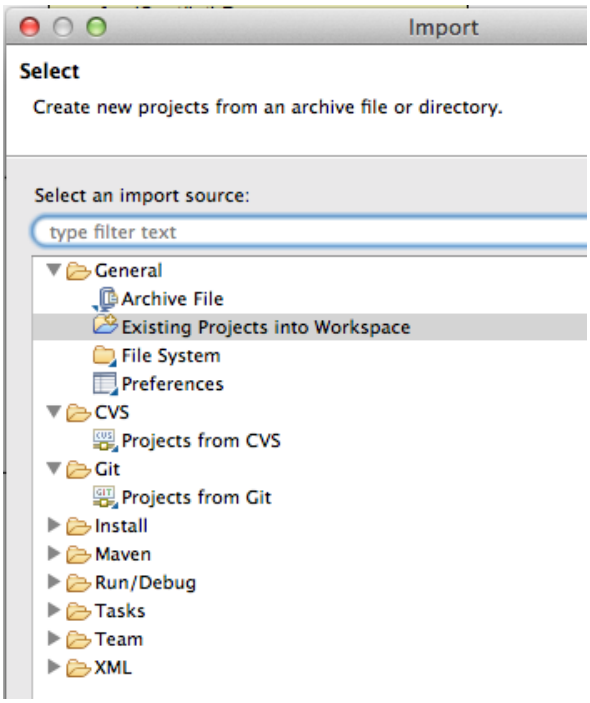
PRÁCTICA A001_2020.- Eventos

Esta práctica va a **gestionar eventos** de cualquier tipo, que tiene un número de butacas, con posibilidad de comprar el ticket con un descuento en el precio por ser venta anticipada.

Las butacas del evento 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). Además indicará si dicha compra se realiza con compra anticipada.

PASOS PARA REALIZAR LA PRÁCTICA:



1. **Descargar el proyecto** de la página de la asignatura en agora.unileon.es:

   - edi-a001-2020.zip

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

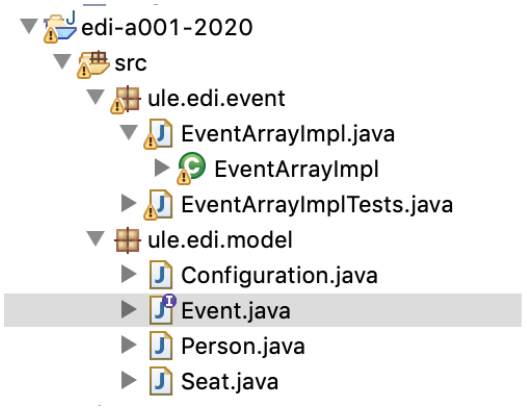
   (seleccionar el archivo ZIP descargado)

   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**.

   <u>En este paquete SOLAMENTE HAY QUE IMPLEMENTAR el método equals de la clase Person.</u>

   Las clases que contiene este paquete son:

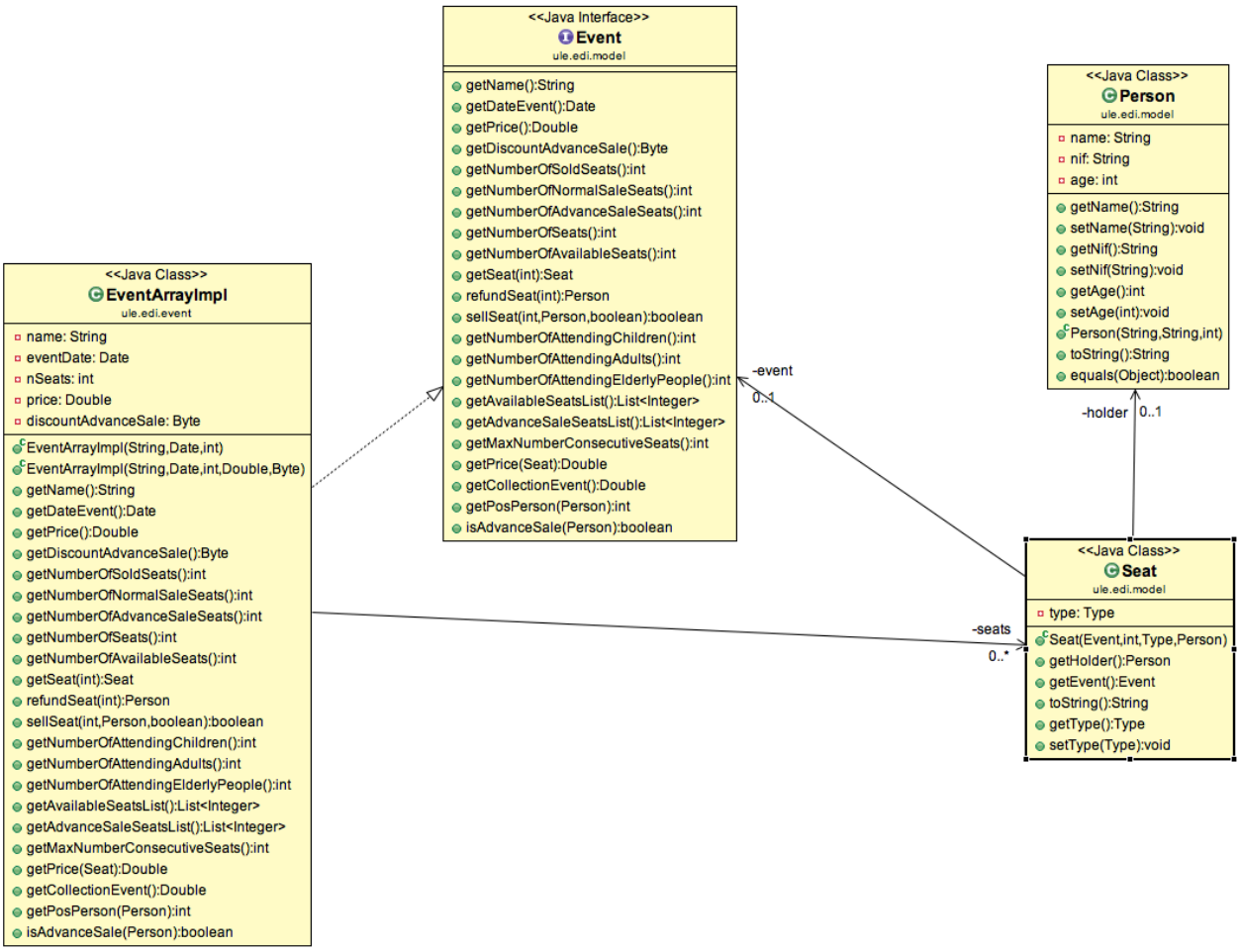
   

   - **Configuration**: Define constantes y un tipo enumerado.
   - **Person** : Define una persona. En esta clase hay que **<u>implementar el método equals</u>**.
   - **Seat**: Define una butaca ocupada, indicando la persona que la ocupa, el tipo de butaca que es (venta normal o anticipada) y el evento 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 un array 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:

```
public interface Event {

	public String getName();

	public Date getDateEvent();

	public Double getPrice();

	public Byte getDiscountAdvanceSale();


	//Calcula el número de asientos totales vendidos del evento.
	public int getNumberOfSoldSeats();
```

```
    // Calcula el número de butacas vendidos en venta normal del evento.
    public int getNumberOfNormalSaleSeats();

    // Calcula el número de butacas vendidas en venta anticipada.
    public int getNumberOfAdvanceSaleSeats();

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

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

    //Devuelve la butaca en la posición dada
    //         o null si no está ocupada o si pos no es válida
    //Las posiciones empiezan en '1'.
    public Seat getSeat(int pos);

    //Libera la butaca de la posición dada.
   // Si la butaca de esa posición ya está libre o la posición no es válida,
    // devuelve null.
    //    sino devuelve la persona que ocupaba la butaca
    // Las posiciones empiezan en '1'.
    public Person refundSeat(int pos);

    // Si la butaca de esa posición ya está ocupada, no hace nada.
    // Las posiciones empiezan en '1'.
    // Parámetros:
    //  p : la persona que ocupará la butaca
    //  isAdvanceSale: true si es venta anticipada; false si es venta normal
    // Devuelve true si se pudo realizar la venta de la butaca,
    //false en caso contrario (ya estaba ocupada)
    public boolean sellSeat(int pos, Person p, boolean isAdvanceSale);

   // Calcula el número de niños asistentes al evento.
   // edad < Configuration.CHILDREN_EXMAX_AGE
    public int getNumberOfAttendingChildren();

    // Calcula el número de adultos asistentes al evento.
    //    [CHILDREN_EXMAX_AGE, ELDERLY_PERSON_INMIN_AGE)
    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
     public int getNumberOfAttendingElderlyPeople();

    // Calcula la lista de números de butacas disponibles
    //  IMPORTANTE: Tener en cuenta que las posiciones empiezan en 1
     public List<Integer> getAvailableSeatsList();

     // Calcula la lista de números de butacas vendidas en venta anticipada
    //   Tener en cuenta que las posiciones empiezan en 1
    public List<Integer> getAdvanceSaleSeatsList();

     // Calcula el número máximo de posiciones disponibles consecutivas
     // Ejemplo: siendo X una posición ocupada, devolverá 3
     //  [null,X,null,null,null,X,X,null,null,X,null]
    public int getMaxNumberConsecutiveSeats();
```

```
      // Calcula el precio de la butaca en función del tipo de venta
      //      y del descuento si es venta anticipada de esa butaca
      //  Si seat es null o si el evento que tiene asignado la butaca no es
      //  el que hace la llamada (this), devolverá 0
      public Double getPrice(Seat seat);


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

      // Devuelve el número de butaca que ocupa la persona pasada
      //   como parámetro
      //  Si la persona no está, devuelve -1
      //  Si la persona ocupase más de una butaca, devuelve el número menor
      public int getPosPerson(Person p);

      // Devuelve True si la persona ocupa una butaca cuyo ticket
      //          fue comprado en venta anticipada.
      //   false en caso contrario
      // Si la persona ocupase más de una butaca, se considerará la butaca de menor n°
      //  Si no ocupa ninguna butaca, devuelve false
      public boolean isAdvanceSale(Person p);
}
```

ATRIBUTOS DE LA CLASE EventArrayImplEvent: Clase que implementa el interface Event

```
private String name;
private Date eventDate;
private int nSeats;

private Double price;    // precio de entradas
private Byte discountAdvanceSale;   // descuento en venta anticipada (0..100)

private Seat[] seats;
```

ATRIBUTOS DE LA CLASE Seat:

```
private Person holder;
private Event event;
private Type type;
```

ATRIBUTOS DE LA CLASE Person:

```
private String name;
private String nif;
private int age;
```

CONSTANTES INCLUIDAS EN LA CLASE Configuration:

```
public static final Integer CHILDREN_EXMAX_AGE = 14;
public static final Integer ELDERLY_PERSON_INMIN_AGE = 65;
public static final Double DEFAULT_PRICE=100.0;
public static final Byte DEFAULT_DISCOUNT=25;
```

```
public enum Type {
            ADVANCE_SALE,
            NORMAL
      }
```

FICHERO PARA INCLUIR LOS TESTS DE LA PRÁCTICA

```
public class EventArrayImplTests {

      private DateFormat dformat = null;
      private EventArrayImpl e;

      private Date parseLocalDate(String spec) throws ParseException {
        return dformat.parse(spec);
      }

      public EventArrayImplTests() {
            dformat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
      }

      @Before
      public void testBefore() throws Exception{
          e = new EventArrayImpl("The Fabulous Five",
                          parseLocalDate("24/02/2018 17:00:00"), 110);
      }

      @Test
      public void testEventoVacio() throws Exception {
          Assert.assertTrue(e.getNumberOfAvailableSeats()==110);
          Assert.assertEquals(e.getNumberOfAvailableSeats(), 110);
          Assert.assertEquals(e.getNumberOfAttendingAdults(), 0);
      }

      @Test
      public void testSellSeat1Adult() throws Exception{
          Assert.assertEquals(e.getNumberOfAttendingAdults(), 0);
          Assert.assertTrue(e.sellSeat(1,
            new Person("10203040A","Alice", 34),false)); //venta normal
          Assert.assertEquals(e.getNumberOfAttendingAdults(), 1);
          Assert.assertEquals(e.getNumberOfNormalSaleSeats(), 1);
      }


      @Test
      public void testgetCollection() throws Exception{
            Event  ep = new EventArrayImpl("The Fabulous Five",
parseLocalDate("24/02/2018 17:00:00"), 4);
            Assert.assertEquals(ep.sellSeat(1, new Person("1010", "AA", 10),
true),true);
            Assert.assertTrue(ep.getCollectionEvent()==75);
      }
}
```

CONSIDERACIONES PARA LA IMPLEMENTACIÓN DE LA PRÁCTICA:

- El constructor de la clase EventArrayImpl **debe crear el array de Seat**, 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, boolean isAnticipedSale).**
- 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.
- 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. No hay que disparar excepción, pero si controlar que es una posición válida para que no de errores por acceder a una posición inválida del array.
- En el método **sellSeat(int pos, Person p, boolean isAnticipedSale)**, 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).
- En el método **getPriceSeat(Seat s),** si el evento que tiene asignado la butaca no es el que hace la llamada (this), **devolverá 0.**
- **IMPORTANTE**: Utilizar JUNIT 4.
- Se valorará la cobertura total de los test implementados, por lo que debe usarse un plugin de Eclipse (por ejemplo EclEmma) para comprobar la cobertura de los test generados.

FECHA LIMITE de entrega de la práctica A001-2020:
6 de Marzo de 2020 23:59