



1er fecha 2024

Parcial - Orientación a Objetos I - 16/11/2024

Un sistema tiene el objetivo de gestionar la compra de entradas a eventos para usuarios. En esta plataforma hay dos tipos de eventos: presenciales, donde los asistentes deben acercarse al lugar del evento, o virtuales, donde se conectan de forma remota al mismo.

De cada evento se conoce su nombre, la fecha, el tema principal, el precio de inscripción y el precio de una remera que se entrega como artículo promocional del evento. Los eventos presenciales se llevan a cabo en varias sedes. Cada sede tiene nombre del lugar, precio de estadía por día y cantidad de días de la estadía. En los eventos virtuales además se cobra un monto fijo por el envío de la remera.

De cada usuario se conoce su nombre y las entradas que compró. Además, los usuarios pueden consultar el precio de asistencia a un evento sin necesidad de comprar una entrada.

El precio de asistencia a un evento se calcula de la siguiente manera:

- Para los virtuales es la suma del **precio de inscripción**, el precio de la remera y el monto fijo por el envío.
- Para los presenciales es la suma del **precio de inscripción**, el precio de la remera y la suma de los precios totales de estadía en cada una de las sedes de ese evento. Cada sede tiene un precio total que es el precio de estadía por día multiplicado por la cantidad de días del mismo.
- En ambos casos, si el evento se realiza el mismo día en que se hace la consulta, el **precio de inscripción** tiene un recargo del 20%.

Al momento de comprar una entrada, se puede decidir si pagar o no un seguro de reembolso que permite recuperar un mayor monto en caso de devolución de la entrada. Este seguro tiene un precio de \$500.

Nos piden implementar la siguiente funcionalidad:

Consultar el precio de asistencia a un evento: dado un evento y una fecha retorna el precio de asistencia al evento para la fecha indicada.

Comprar una entrada para un evento: dado un usuario, el evento, y si tiene o no seguro de reembolso, retorna una entrada para dicho evento con fecha de compra igual a la fecha actual.

Calcular el monto a recuperar por una entrada: dada una entrada, retorna el monto que se recupera en caso de devolución. El monto a recuperar se calcula teniendo en cuenta la diferencia de días entre la fecha de compra de la entrada y la fecha del evento:

- Si la diferencia es mayor o igual a 30 días, el monto a recuperar es 50% del precio de asistencia.
- Si la diferencia es menor, el monto a recuperar es 0.

En todos los casos anteriores, si la entrada fue adquirida con seguro de reembolso, al monto a recuperar anterior se suma el 15% del precio de asistencia.

Calcular el monto total por las entradas compradas en un período: dado un usuario, un intervalo de tiempo (fecha inicial - fecha final), retorna el monto total por entradas compradas en ese intervalo.

Retornar la entrada al siguiente evento: dado un usuario, se debe retomar la entrada comprada por el usuario al siguiente evento (el que tenga la fecha posterior más cercana al día de hoy)

Tareas:

1. Describa el diseño de su solución utilizando un diagrama de clases UML.
2. Implemente en Java toda la funcionalidad antes descripta. Puede usar las clases implementadas en los ejercicios de la práctica: DateLapse, FilteredSet, Bag, según se requiera.
3. Diseño de los casos de prueba: Enfocándose en la funcionalidad que permite calcular el monto a recuperar por una entrada (todos los métodos de todas las clases involucradas en conseguir esa funcionalidad), determine y enumere qué métodos testear, indicando clases y casos de prueba (teniendo en cuenta los conceptos de valores de borde y particiones equivalentes). Identifique claramente las particiones que encuentra y los valores de borde para cada caso.
4. Escriba un ejemplo del código Java necesario para un usuario que compró una entrada a un evento presencial con los valores que necesite en cada caso. El evento debe incluir dos sedes asociadas.

Para conocer los días entre dos fechas: `int días = Period.between(fechaInicio, fechaFin).getDays();`

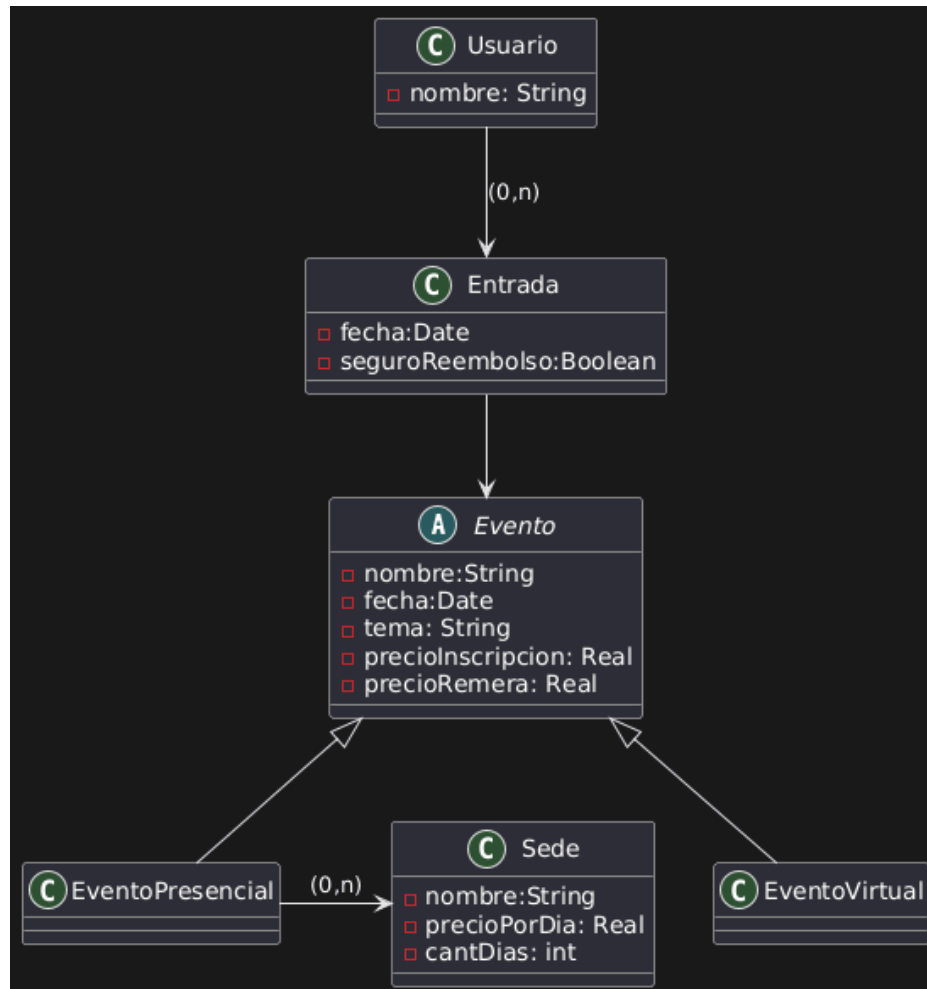
Tenga en cuenta que las 4 tareas anteriores son **requeridas para aprobar**. No deje ninguna en blanco.

1) Conceptos:

Evento, entradas, usuarios, evento presencial, evento virtual, nombre, fecha, temaPrincipal, precioInscripcion, precioRemera, sedes, nombreSede, precioPorDia, cantidadDiasEstadia, nombreUsuario, entradasQueCompro, consultarPrecioAsistencia, calcularPrecioAsistencia, comprarEntrada, seguro

de reembolso, calcularMontoARecuperar,
calcularMontoTotalPorEntradasEnPeriodo, retornarEntradaAlSiguienteEvento

Modelo de dominio:



Modelo UML:

▼ PLANTUML

@startuml

```
class Usuario{
  -nombre: String
  +<<Create>> Usuario (nombre:String): Usuario
  +consultarPrecio():Real
  +comprarEntrad(Evento e, boolean seguro): Entrada
}
```

```

    +montoEntrada(periodo Date):Real
    +siguienteEvento():Entrada
}

class Entrada{
    -fecha:Date
    -seguroReembolso:Boolean
    -evento: Evento
    +<<Create>> Entrada (evento Evento, seguroReembolso boolean): Entr
ada
    +getFecha(): Date
    +precioE(): Real
    +calcularDevolucion():Real
}

abstract class Evento{
    -nombre:String
    -fecha:Date
    -tema: String
    -precioInscripcion: Real
    -precioRemera: Real
    +<<Create>> Evento(nombre String, fecha Date, tema String, pi Real, pr
Real): Evento
    +getPrecio(): Real
    +getPrecioRemera():Real
    +getFecha():Date
    +<<abstract>> precioAsistencia(fecha Date): Real
}

class EventoPresencial{
    +<<Create>> EventoPresencial(nombre String, fecha Date, tema String,
pi Real, pr Real): EventoPresencial
    +precioAsistencia(fecha Date): Real
    +agregarSede(Sede s)
}

```

```

class EventoVirtual{
    +<<Create>> EventoVirtual(nombre String, fecha Date, tema String, pi Real, pr Real): EventoVirtual
    +precioAsistencia(fecha Date): Real
}

```

```

class Sede{
    -nombre:String
    -precioPorDia: Real
    -cantDias: int
    +precioTotal():Real
}

```

Usuario → Entrada: --entradas
 Entrada → Evento: --evento
 Evento <|-- EventoPresencial
 Evento <|-- EventoVirtual
 EventoPresencial → Sede: --sedes
 @enduml

