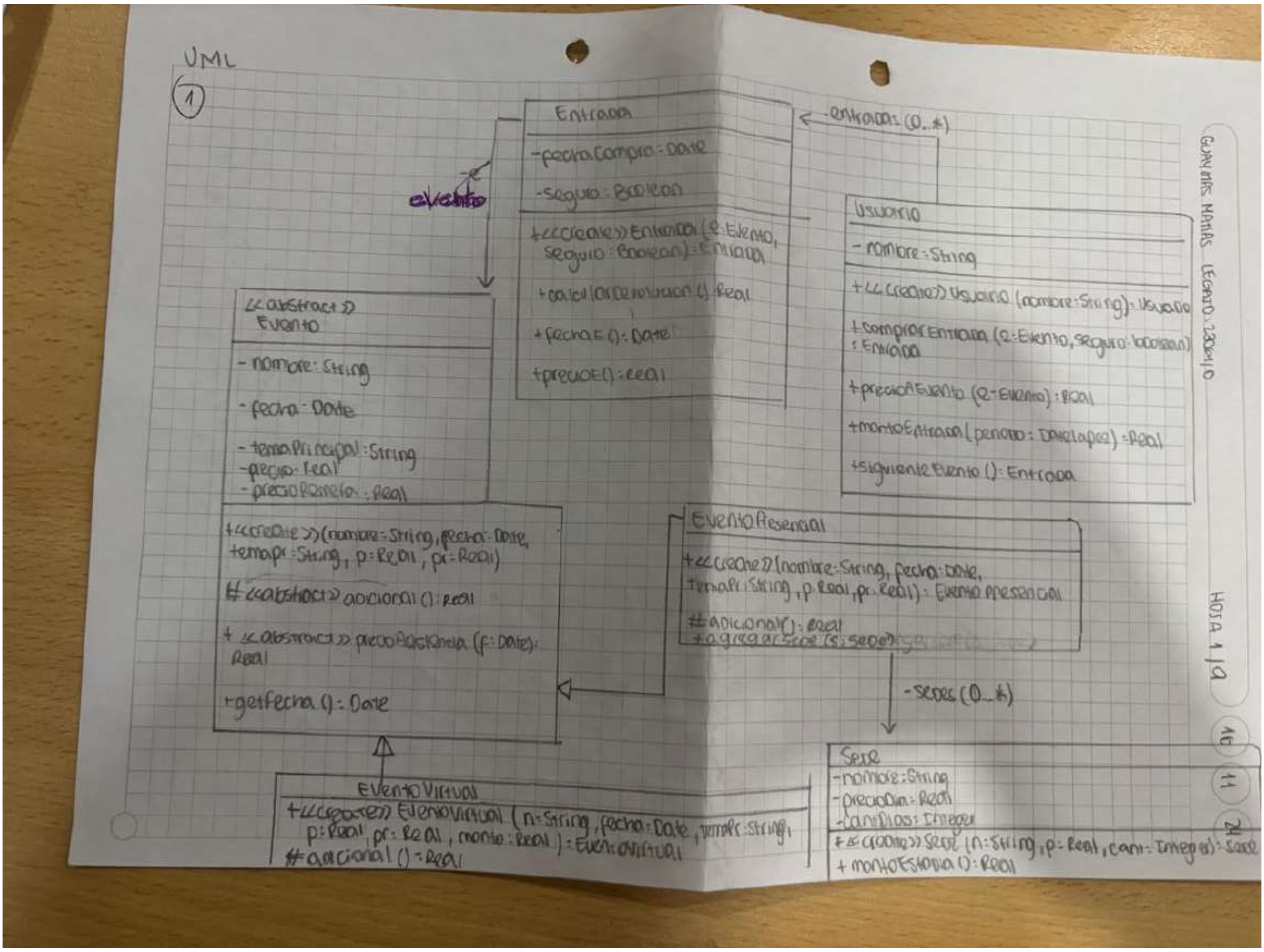

UML
1
Entrada
-fechaCompra: Date
-seguro: Boolean
+<<create>> Entrada (e: Evento, seguro: Boolean): Entrada
+calcularDevolucion (): Real
+fechaE (): Date
+precioE (): Real
-e
evento
-entradas (0..*)
Usuario
-nombre: String
+<<create>> Usuario (nombre: String): Usuario
+comprarEntrada (e: Evento, seguro: boolean): Entrada
+precioAEvento (e: Evento): Real
+montoEntrada (periodo: DateLapse): Real
+siguienteEvento (): Entrada
<<abstract>>
Evento
- nombre: String
- fecha: Date
- temaPrincipal: String
-precio: Real
- precioRemera: Real
+<<create>> (nombre: String, fecha: Date, temaPr: String, p: Real, pr: Real)
#<<abstract>> adicional (): Real
+ <<abstract>> precioAsistencia (f: Date): Real
+getFecha (): Date
EventoPresencial
+<<create>> (nombre: String, fecha: Date, temaPr: String, p: Real, pr: Real): EventoPresencial
#adicional (): Real
+agregarSede (s: Sede)
-sedes (0..*)
EventoVirtual
+<<create>> EventoVirtual (n: String, fecha: Date, temaPr: String, p: Real, pr: Real, monto: Real): EventoVirtual
#adicional (): Real
Sede
-nombre: String
-precioDia: Real
-cantDias: Integer
+<<create>> Sede (n: String, p: Real, cant: Integer): Sede
+ montoEstadia (): Real
GUANERAS MATIAS LEGAJO: 23044/0
HOJA 1/9

Aprobado

GUAYMAS MATIAS LEGAJO: 230611/0 Andrés HOJA 2/9 10 11 24

②

```java
public class Usuario {
    private String nombre;
    private List<Entrada> entradas;
    public Usuario (String nombre){
        this.nombre = nombre;
        this.entradas = new ArrayList<Entrada>();
    }
    public Entrada comprarEntrada ( Evento e, boolean seguro){
        Entrada ent = new (e, seguro);
        this.entradas.add(ent);
        return ent;
    }
    public double precioAEvento (Evento e){
        if (e != null){
            return e.precioAsistencia(LocalDate.now());
        }
        return -1; //En caso de ser null devuelvo un valor imaginario
    }
    public double montoEntrada (DateLapse periodo){
        return this.entradas.stream().filter (e -> periodo.includesDate(e.fechaE()))
        .mapToDouble (e2 -> e2.precioE()).sum();
    }
    public Entrada siguienteEvento(){
        ~~if (this.entradas.isEmpty()) return null;~~
        return this.entradas.stream().sorted ((ex1) -> ex1.fechaE().compareTo(
            LocalDate.now()).collect (Collectors.toList()).~~get(0)~~;
            .findFirst().orElse(null)
```

- Implementa método abstracto
- Envidia de atributo
- entrada SiguienteEvento no cumple lo solicitado

[redacted]

- No suma el costo del seguro
- MontoEntrada no cumple lo solicitado

Usado ✓

Código defensivo innecesario

Pasamanos.

~~Mala solución~~ No aporta a la solución

✗

No calcula el monto de las entradas.

Calcula el monto de los eventos

Debería decidirlo la entrada.

[redacted]

Envidia de atributo

✱

```
public Entrada[] siguienteEvento() {
    List<Entrada> ls = new ArrayList<Entrada>(this.entradas);
    ls = ls.stream().sorted((e1, [illegible]) -> e1.fecha e[1].compareTo(LocalDate.now())).
            collect(Collectors.toList());

    if (ls.size
```

incompleto

* No filtra las entradas con fecha pasada. No cumple lo solicitado

GUAYMAS MAGAS [illegible] HOJA 3/4

```
public class Entrada {
    private LocalDate fechaCompra;
    private Evento evento;
    private boolean seguro;
    public Entrada(Evento e, boolean seguro){
        this.fechaCompra = LocalDate.now();
        this.e = e;
        this.seguro = seguro;
    }
    public double calcularDevolucion() {
        int dias = Period.between(this.fechaCompra, e.getFecha()).
        Double recuperar = 0;
        if((dias >= 30) || (this.seguro)){
        Double precioA = this.e.precioAsistencia(LocalDate.now());
            if (Dias >= 30){
                recuperar = precioA * 0,5;
            }

            if (this.seguro{
                recuperar += precioA * 0,15;
            }
        return recuperar;
    }
    public LocalDate fechaE(){
        return this.e.getFecha();
    }
    public double precioE(){
        return this.e.precio;
    }
```

Usar nombres inteligibles.

No se considera el precio del seguro en el costo de la Entrada

Tiene que ser de la fecha de compra

Falta una llave. No es claro dónde termina el primer if

GUAYMAS MATIAS LEGAJO: 220611/0 HOJA 4/9 16 11 24

```
public abstract class Evento{
    private String nombre;
    private LocalDate fecha;
    private String temaPrincipal;
    private double precio;
    private double precioRemera;
    public Evento(String nombre, LocalDate fecha, String temaPr, double p, double pr){
        this.nombre=nombre;
        this.fecha=fecha;
        this.temaPrincipal=temaPr
        this.precio=p
        this.precioRemera=pr;
    }
```

Un método abstracto no puede tener implementación

```
public abstract double precioAsistencia(LocalDate f){
    Double precioA=this.precio+this.precioRemera+this.adicional();
    if( f).equals(this.fecha)){
        precioA+= precio*0,2;
    }
    return precioA;
}
```

Solo el precio de ~~asistencia~~ inscripción

```
protected abstract double adicional();
public LocalDate getFecha(){
    return this.fecha;
}
```

```
public class EventoVirtual{
    private double montoEnvio;
    public EventoVirtual(String n, LocalDate fecha, String temaPr, double p, double pr, double
                                                                                    monto){
        super(n,fecha,temaPr,p,pr);
        this.monto=monto;
    }

protected double aplicarDual(){
    return this.monto;
}
```

GUAYMAS MATIAS LEGAJO: 230610

HOJA 6/9

16 11 24

```
public class EventoPresencial {
    private List<Sede> sedes;
    public EventoPresencial(String nombre, LocalDate fecha, String temaAR, double p, double pr){
        super(nombre, fecha, temaAR, p, pr);
        this.sedes = new ArrayList<Sede>();
    }
}
protected double adicional() {
    return this.sedes.stream().mapToDouble(s -> s.montoEstadia()).sum();
}
public void agregarSede(Sede s){
    this.sedes.add(s);
}
```

4)).

GUAYMAS MATÍAS LEGAJO: 23061/0 HOJA 7/9 16 11 24

```
public class Sede {
    private String nombre;
    private double precioDia;
    private int cantDias;
    public Sede(String n, double p, int cant){
        this.nombre = n;
        this.precioDia = p;
        this.cantDias = cant;
    }
    public double montoEstadia(){
        return this.precioDia * this.cantDias;
    }
```

③. Clase: Entrada

Método: calcularDevolucion(): real

Casos: 1. - La diferencia de días entre la fecha de compra de entrada y fecha de evento es < 30 → (A)

2. - " es = 30 → (B)

3. - " es > 30 → (B)

4. - La entrada fue adquirida sin seguro de reembolso → (C)

5. - " con seguro de reembolso → (D)

~~Hallar diferencia de días entre~~

Habiendo explicado los casos además entraríamos en juego las combinaciones entre 1. y con 4. y 5., 2. con 4. y 5., y 3. con 4. y 5. Como interesa la funcionalidad, se aplicarían o no ciertas sumas ~~descuentos~~ al valor a recuperar dependiendo de la combinación.

Hay dos casos interesantes:

- Cuando la diferencia es mayor o igual a 30 días y adquirió la entrada con reembolso. En este caso retornaría el 50% del precio de asistencia más el 15% del precio de asistencia.

- Cuando la diferencia es menor a 30 días y NO adquirió la entrada con reembolso. En este caso retorna 0.

(A) Retorna 0. + ~~(15% o 5.~~ resultado de si tiene seguro.

(B) Retorna 50% precio de asistencia + resultado de si tiene seguro

(C) Retorna resultado de chequeo por dif. de días entre las fechas * (valor cambio) + 0

(D) Retorna resultado de chequeo por dif. de días entre las fechas (valor devolver + (15% de precio asistencia))

~~(a) Un evento presencial siempre tiene una sede, ¿es necesario que la sede tenga mínimo o al menos de una sede, porque no todos los [illegible]?~~

- Para clase: Evento

Método: precioInscripcion(fechaConsulta): Real

Casos: - es un evento virtual y la fecha de consulta coincide con la del evento → retorna (E)
- " y " no coincide con la del evento → (F)
- es un evento presencial y la fecha de inscripción coincide con la del evento → (G)
- " y " no coincide con la del evento → (H)

(E) Retorna precio de inscripción + precioBase + montoFijo + Recargo del 20% de precio inscripción

(F) " + " + " // SIN RECARGO DEL 20% DE precio inscripción

(G) Retorna precio de inscripción + precioBase + suma total de precios estadía con sede + Recargo del 20% de precio inscripción

(H) " + " + " // SIN RECARGO DEL 20% DE precio inscripción

No testeo el método adicional por ser protected además, en la clase EventoPresencial solo suma monto, resultado de una multiplicación entre dos valores. En caso de no tener sedes, devolvería 0, caso contrario suma montos.

GUAYMAS MATIAS LEGAJO: 230061/0 HOJA 9 | 9

16 11 24

4:

```
public static void main(String[] args) {

    Usuario u = new Usuario("Matias");

    Sede s1 = new Sede("LP", 400, 10);

    Sede s2 = new Sede("HUDSON", 200, 5);

    ~~EventoPresencial e = null;~~
    ~~// Acá iría haciendo Evento e = new EventoPresencial(... datos);~~
    EventoPresencial e = new EventoPresencial("Evento", LocalDate.of(2024, 01, 31), "Bitcoin", 400, 100);

    e.agregarSede(s1);
    e.agregarSede(s2);

    u.comprarEntrada(e, true);

}
```