***Informe***

**Primer feedback del profesor:**

1) No cierra que el tp esté en inglés, el enunciado está en castellano, nosotros hablamos en castellano, lo hizo más difícil de entender. También había algunos errores de ortografía (with sin h), que probablemente serían menos frecuentes en español ;-). Finalmente, tomando la idea de lenguaje común (ubiquitous language) de Evans en Domain Driven Design, el lenguage tiene que ser uno único compartido por todos, por lo tanto, arranca con usar el lenguaje hablado.

2) Las pruebas de PromotionTest son correctas.

3) El método:

public double getCost() {

double cost = 0;

cost = calculateCost(this.getAttractions());

cost = cost- ((cost \* percentage)/100);

return cost;

}

podría escribirse más claramente así:

public double getCost() {

double cost = calculateCost(this.getAttractions());

return cost- ((cost \* percentage)/100);

}

4) No cierra que PercentagePromotion y AbsolutePromotionn redefinan applyPromotion y que el código sea el mismo. Debería estar en Promotion. Tampoco me termina de cerrar el código de ese método, no lo entiendo, tal vez falten algunas aclaraciones sobre la fórmula. El de AXB sí lo entiendo.

5) El método calculateCost está repetido con la misma implementación en varios lugares, debería estar en uno sólo, la redundancia innecesaria es una de las peores males del software

6) ¿No sería calculateTravelTime un método de itinerario? No me imagino a la secretaría de turismo haciendo ese cálculo. Idem con el tiempo total.

7) Disponibility no es la palabra, se dice Availability, igual debería estar en castellano.

8) En verifyAttractionIsAppropiateForUser, si la atracción es apropiada, la borrás, no se entiende por qué. En cualquier caso, el nombre no sugiere que haga eso, dice verify, no modify.

**Correcciones aplicadas a la devolución:**

1) Se corrigió todo el código y se lo tradujo al castellano.

3)Se aplicó un refactor al método señalado.

4) Se asignó el método “applyPromotion”, ahora llamado “aplicarCostoDePromocion” a la clase Promocion, de manera que el único que terminaría redefiniendo el mismo sería PromocionPorcentual.

Se aplicó un refactor a ese mismo método tratando de hacerlo más declarativo y entendible.

5) Se asignó el método calcularCosto a la clase Promocion para evitar las repeticiones innecesarias.

6)Respecto a este ítem, consideré que dado que el usuario lo tiene asignado la Secretaría de turismo, me pareció más apropiado que determine él, ell tiempo de viaje total para el recorrido, ya que es el que tiene la información de la velocidad a la que se desplaza el usuario.

Consideré que Una sugerencia debía tener la información necesaria para el usuario, pero no la lógica del negocio.

8)Se aplicó un refactor a la forma en que se generan las sugerencias.

**Cambio importante:**

He realizado un gran cambio a la forma en que se generan las sugerencias, por considerar que la forma en que lo estaba realizando era una complejidad innecesaria.

En el pasado, a medida que se agregaba una atracción a la sugerencia, se aplicaban las promociones de manera que quede un mayor margen de dinero, a causa del descuento, para que entren más atracciones.

Ahora agrego las atracciones hasta que me lo permita el dinero y el tiempo que cuenta el usuario. Y posteriormente aplico los descuentos de las promociones.

**Correcciones aplicadas después de la revisión (Informe de impacto):**

Al depender de una jerarquía de herencia con Promocion y sus implementaciones, se encontró una dificultad de diseño al agregar la promoción PromocionFamiliar, a la hora de poder aplicar el método redefinido aplicarCostoDePromocion(), ya que este mismo, necesitaba pararmetrizar la cantidad de entradas para realizar el cálculo necesario.

Consideré que por la forma en que genero las sugerencias (sin aplicar ninguna promoción a las atracciones hasta que tengo la sugerencia completa), era necesario aplicar primero las 3 promociones que tenía en un principio y por último la nueva Promocion PromocionFamiliar, ya que de ésta manera el descuento iba a ser mayor porque se aplicaría sobre el descuento obtenido por las otras promociones.

Se agregó un atributo más a las sugerencias, cotoXCantidadDeEntradas, de tal manera que además de calcularse el costo total por persona también se cuente con el dato del costo total por la cantidad de entradas solicitadas.

**Clases que se vieron afectadas con cambios:**

Debido a estos cambios las clases que se vieron impactadas fueron:

SecretariaDeTurismo (cambios referentes al “cambio importante”, la forma en que se manejan las promociones, y la validación para detectar si el usuario es extranjero).

Sugerencia(Se agregaron los campos correspondientes al costoTotalXCantidadEntradas y costoTotalPorPersona).

Promocion(Se agregaron las clases PromocionFamiliar y PromocionExtranjero).

**Segundo feedback del profesor:**

1 ) Me sirvió el repaso de las correcciones anteriores, en general tiene mucha mejor pinta, se nota el esfuerzo.

2) No me cierra para nada el uso de instanceof en SecretariaDeTurismo.obtenerPromocionDeExtranjeros. Estas cosas hacen difícil modificar el sistema porque anulan el polimorfismo. Además, agregas en algún punto la promoción y después el obtener la borra. Este es un error del mismo tipo que otro que ya habíamos discutido, el nombre del método no se condice con lo que hace. Hablemos si no te queda claro qué hacer.

3 ) No me queda claro cierra el “Cambio importante”. Entiendo que quisiste simplificar, pero no me queda claro si no es en detrimento del servicio que le prestas al usuario. ¿Le sirve al usuario como está? Porque si el límite se aplica antes de las promociones, entonces lo que paga al final es menos del límite, por lo tanto puede que el sistema no le sugiera cosas que en realidad sí podría hacer.

4) No me cierra el código de generarSugerencias, que hace un if (distancia > 200) para cada atracción, cuando debería ser una característica del Usuario ser extranjero, no de la relación usuario-atracción (el enunciado dice, es extranjero si está a + de 200 km de distancia de la más cercana a él).

5) Este código es confuso, redundante y parece difícil de modificar  
                if (promocionFamiliar != null) {  
                        sugerencia.setCostoXCantidadDeEntradas(promocionFamiliar  
                                        .aplicarCostoDePromocion(fecha, atracciones, costo,  
                                                        cantidadDeEntradas));  
                } else {  
                        sugerencia.setCostoXCantidadDeEntradas(costo \* cantidadDeEntradas);  
                }

6 ) El informe en general está claro, aunque estaría bueno en la parte de impacto aclarar qué clases se vieron afectadas.

7) No sé si no lo había notado antes, pero el diagrama de clases está un poco anémico, no tiene info de cardinalidades, por ejemplo.

8 ) El daigrama de secuencia está desactualizado, está en inglés.

**Correcciones aplicadas a la devolución:**

2) Se eliminó el uso del instanceof, y la necesidad de buscar la clase PromocionFamiliar para guardar la instancia como constante. Esto se logró, por un lado, identificando las promociones que son acumulables de las no acumulables (como PromocionExtranjero) y manejarlas de forma separada, parametrizandolas de manera independiente en el constructor de la SecretaríaDeTurismo. Por otro lado, ahora se maneja la PromocionFamiliar como una promocionAcumulable más. Posteriormente se explicará mejor este punto como un cambio importante.

3)Se discutió con el profesor y se acordó que la suposición de cargar primero las atracciones para el usuario y luego las promociones es aceptable ya que la definición algorítmica iba a ser un tema de elección personal.

4)Se discutió con el profesor y pude entender cuál era el error que me estaba marcando.

Se corrigió, y ahora se determina si el usuario es extranjero si la distancia a la atracción más cercana de todas las atracciones de la SecretaríaDeTurismo es mayor a 200 km. Anteriormente se hacía esta validación sobre el conjunto de atracciones ya filtradas en cada sugerencia que se generaba.

5)Se eliminó esa porción de código ya que ahora se trata a la PromocionFamiliar como una promoción no acumulable más. Esto está asociado a algunos cambios importantes discutidos posteriormente.

6) Se agregó que clases fueron afectadas en la primera entrega.

7)Se agregaron los cardinales correspondientes en el diagrama de clases.

8) Se actualizó el diagrama de secuencias.

**Cambios Importantes:**

Dado que ahora se trata a la PromocionFamiliar como una promoción acumulable más se presentó la dificultad de manejar el costo de la sugerencia dinámicamente cuando se quiere comprar más de una entrada, a diferencia de antes que se guardaba en la clase Sugerencia en el atributo “costoXCantidadDeEntradas”.

Es decir, ahora Sugerencia sólo tiene un atributo costoTotal, que acumulará el costo total de las atracciones en función de la cantidad de entradas.