PROGRAMACIÓN CON OBJETOS II

Trabajo Integrador

Alquileres (Hito2)

GRUPO: TRIPLET

Nicolás Tancredi
tancredi.nikco@gmail.com
Pablo Troche
trochepablodaniel@gmail.com
Agustín Tupilojón
atupilojon@gmail.com

CORRECCIONES HITO 1

Respuesta punto por punto a las observaciones recibidas.

- Propietario debe ser un atributo de Inmueble, y no de Publicación. A menos que interpreten que distintas publicaciones de un mismo inmueble pueden tener propietarios distintos.

  TripleT: interpretamos que un inmueble puede estar publicado por uno o más usuarios.

- El diseño no permite identificar cuáles son las posibles categorías para puntuar cada tipo de Puntuable.

  TripleT: se agrega colaboración (objetoPuntuable) a Sitio en UML.

- El diseño no permite identificar cuáles son los posibles servicios que un Inmueble puede tener, ni los posibles tipos de inmuebles.

  TripleT: se agrega colaboración (servicios) a Sitio en UML y código.

- El diseño no responde a la funcionalidad de obtener el promedio del ranking de categorías para cada entidad.

  TripleT: se agregan métodos a UML:

  - Sitio: obtenerPromedioPuntaje que recibe una Categoria y un IPuntuable (Usuario o Inmueble).
  - IPuntuable: promedioPuntaje que recibe una Categoria y un IPuntuable (Usuario o Inmueble).
  - Mismo método en Usuario e Inmueble que implementan IPuntuable.

- No se entiende el siginificado del atributo "valor" en Servicio. Y el mismo no se encuentra en el código.

  TripleT: se corrige código, eliminando el atributo.

- En UML falta relación sitio - inmuebles, que se encuentra en el código.

  TripleT: se agrega colaboración (inmuebles) a Sitio en UML.

- En el método Publicacion#hayReservasEnFecha está preguntando por la clase:
  rs.getEstado() instanceof EstadoAprobado
  Esto es incorrecto, está preguntando por la clase. Debe delegar en el objeto correspondiente.
  Además, hay un claro Feature Envy (bad smell).

  TripleT: se delegan responsabilidades a EstadoReserva y sus subclases.

- EstadoReserva y sus subclases no tienen comportamiento. Sólo hacen una redirección innecesaria para setear el estado de la reserva.

  TripleT: siguiendo con el punto anterior, delegar responsabilidad aumenta el comportamiento de la clase.

- Exponer métodos más entendibles para manejar el estado de la reserva. #cambiarEstado debería ser un método privado (el haber elegido el Patrón State no debería exponerse hacia afuera), y exponer métodos que sean entendibles:

  TripleT: refactoring de EstadoReserva y subclases.

  - Cómo se acepta una reserva?

    - Publicacion: aprobarReserva(Reserva r)
    - Reserva: aprobarReserva()
      - La Reserva cambia a EstadoAprobado.

  - Cómo se cancela una reserva?

    - Publicacion: cancelarReserva(Reserva r)
    - Reserva: cancelarReserva()
      - La Reserva cambia a EstadoCancelado.

- En FiltroCiudad utilizar el método #equals() y no #== (investigar el motivo)

  TripleT: == es un operador que denota true cuando ambos objetos están referenciados a la misma posición en memoria, mientras que equals() compara (o evalua) los valores que denotan los objetos.

- No se permite que un propietario publique su inmueble sólo en un rango de fechas

  TripleT: se agregan los atributos fechaInicio y fechaFin en Publicacion, donde fechaInicio representa el momento de la creación de la publicación y fechaFin el último día de vigencia. Se agrega un método en Sitio al método getPublicaciones() que se denomina actualizarPubliaciones() para que cada vez que se lo llame se actualice el listado de publicaciones, eliminando aquellas que están vencidas, delegando en Publicacion a través del método estaVigente() determinar la validez de la publiación.

ESTRATEGIA HITO 2

Búsquedas más específicas.

La extensión para este punto consiste en transformar a Buscador en una superclase cuyas subclases (BuscadorConjuntivo y BuscadorDisyuntivo) redefinen el algoritmo de búsqueda a aplicar, configurando de esta forma un **patrón Strategy**: dependiendo del tipo de Buscador que se le envíe como parámetro al método definido en Sitio buscarPublicaciones(Buscador buscador) se ejecutará una **concreteStrategy** u otra.

Promociones.

Determinar si es una Promoción o no es un responsabilidad delegada a Publicacion en el método tienePromocion() en virtud de las fechasInicio y fechaFin.

Para el anuncio a quienes se suscriban a recibir información de una publicacion que es promoción se configura un **patrón Observer** dónde Sitio es el **subject** (administra las suscripciones y anuncia las actualizaciones) y, Usuario y Empresa son los **concreteObservers** que implementan la interfaz ISuscriptor.

AlquileresHito2

TripleT | Objeto 2