Trabajo Practico Integrador:
A la caza de las vinchucas

Materia: Programación con Objetos II.

Integrantes:

- Ariel Giménez
- Alexander Ferragut
- Nicolás Bossi.

e-mails:

- alexanderferragut@gmail.com
- ariel_5195@hotmail.com
- nbossi@hotmail.com

Patrones de Diseño

Patrón State: Se utilizo el Patrón State para modelar la jerarquía del Estado de la Muestra. Tiene los siguientes participantes:

Estado abstracto: interface IEstadoDeMuestra.
Estados concretos: Clases EstadoNoVerificado, EstadoStandBy, EstadoVerificado.
Contexto: Clase Muestra.
Transiciones:

1. De EstadoNoVerificado a EstadoStandBy
2. De EstadoStandBy a EstadoVerificado.

Patrón Template Method: Se utilizo en un refactor dentro del patrón State para modelar los estados de la muestra.

Abstract Class: EstadoDeMuestra.
Concrete Class: EstadoNoVerificado, EstadoStandBy, EstadoVerificado.
Primitive Operation: doFiltrarOpiniones(Muestra): List<String>
Template Method: resultadoActual(Muestra): List<String>

Patrón State: Se utilizo el Patrón State para modelar la jerarquía del Estado del Participante, tiene los siguientes participantes:

Estado abstracto: Clase EstadoDeCateogira.
Estados concretos: Clases EstadoParaBasico, EstadoParaExperto, EstadoParaEspecialista.
Contexto: Clase Participante.
Transiciones:

1. De EstadoParaBasico a EstadoParaExperto
2. De EstadoParaExperto a EstadoParaBasico.

Aclaración: El EstadoParaEspecialista no tiene transiciones, es un estado constante del contexto.

Patrón Adapter: Se utilizo el patrón Adapter para adaptar el protocolo de la Especie de Vinchuca como un tipo de opinión, tiene los siguientes participantes.

Objetivo: Interface ITipoDeOpinon.
Adapter: Clase Vinchuca.
Adaptable: Clase abstracta EspecieDeVinchuca.
Cliente: Opinión.

Patrón Composite: Se utilizo el Patrón Composite para modelar los filtros de búsqueda, tiene los siguientes participantes:

Component: Interface Filtro.
Composite: Clase Compuesto.
Leaf: Clases TipoDeInsecto, ÚltimaVotacion, NivelDeVerificación, FechaDeCreación.

Patrón Strategy: Se utilizo el Patrón Strategy para modelar los conectores lógicos para los filtros de búsqueda, tiene los siguientes participantes:

Contexto: Clase Compuesto.
Estrategia Abstracta: Interface Operador.
Estrategias Concretas: Clases Or y And.

Aclaración: La estrategia se establece al instanciar la clase, y no cambia la estrategia en tiempo de ejecución.

Patrón Observer: Se utilizo el Patrón Observer para notificar a las zonas de cobertura cuando se agrega una nueva muestra, tiene los siguientes participantes:

Sujeto concreto: Clase Sistema.
Observador abstracto: Interface IObserverNuevaMuestra.
Observador concreto: Clase ZonaDeCobertura

Aclaración: Se utiliza la clase EventManager para delegar la gestión del sujeto.

Patrón Observer: Se utilizo el Patrón Observer para notificar a las organizaciones cuando se agrega una nueva muestra, tiene los siguientes participantes:

Sujeto concreto: Clase ZonaDeCobertura.
Observador abstracto: Interface IObserverOrganizacion.

Observador concreto: Clase OrganizacionNoGubernamental.

Aclaración: Se utiliza la clase EventManagerZona para delegar la gestión del sujeto.

Patrón Observer: Se utilizo el Patrón Observer para notificar a las zonas de cobertura cuando se verifica una nueva muestra, tiene los siguientes participantes:

Sujeto concreto: Clase Muestra.
Observador abstracto: IObserverMuestraVerificada
Observador concreto: Clase ZonaDeCobertura.

Patrón Observer: Se utilizo el Patrón Observer para notificar a las organizaciones cuando se verifica una nueva muestra, tiene los siguientes participantes:

Sujeto concreto: Clase ZonaDeCobertura.
Observador abstracto: IObserverOrganizacion
Observador concreto: Clase OrganizacionNoGubernamental.

Aclaración: La interface IObserverMuestraVerificada implementada por la clase ZonaDeCobertura es sujeto de este observer ya que tiene la responsabilidad de notificar a las organizaciones.

Decisiones importantes de Diseño

Consideraciones para la clase Muestra: Al momento de instanciar una muestra se considera que la primera opinión (que es la del creador) es la especie de vinchuca, sin tener en cuenta el estado del participante, por lo tanto, el estado inicial de la muestra a la hora de instanciarse siempre es EstadoNoVerificado.

Detalles importantes de Implementación

Consideraciones para la clase Muestra: La clase EstadoStandBy perteneciente a la clase Muestra tiene una responsabilidad adicional, la

cual es, al momento de realizar la transición a la clase EstadoVerificado, este notificara a los observers de la Muestra (del cambio de estado).

Consideraciones para el Voto del Participante: Se utilizó un enum para modelar el Voto de un participante Básico y el Voto de un participante Experto para que al momento de crear una instancia de Opinión, está reciba en su constructor el voto de un Participante Básico o bien uno Experto, por lo tanto, sí el participante cambia de categoría, la opinión mantiene el voto original.