Trabajo Final

A la caza de las vinchucas

Programación Con Objetos II

Roldan Joaquin
jroldanarin@gmail.com

Wolf Martin Leonardo
wolfmartinleonardo@gmail.com

Patrones

- **state**
  1.
     state: Nivel
     concreteState: NivelBasico y Nivel Experto
     context: Participante
  2.
     state: EstadoMuestra
     concreteState: EstadoMuestraBasico y EstadoMuestraExperto
     context: Muestra

- **strategy**
  context: Organizacion
  strategy: FuncionExterna
  concreteStrategy: FuncionConcreta (solo para test)

- **observer**
  observable: Muestra
  observer: SistemaVinchucas

  observable: SistemaVinchucas
  observer: ZonaCobertura

  observable: ZonaCobertura
  observer: Organizacion

- **singleton**
  singleton: SistemaVinchucas

- **template**
  clase abstracta: Nivel
  clases concretas: NivelBasico y Nivel Experto

Decisiones de diseño

La clase Participante se encarga de recolectar las muestras indicando el TipoDeOpinion y la fecha de recolección.
En la recolección se crea una primera opinión perteneciente al autor de la muestra y además, delega al nivel la actualización del mismo de acuerdo al patrón state. Para actualizarse cada nivel concreto debe cumplir ciertas condiciones sobre la cantidad de envíos y recolecciones hechas por el participante.
El participante puede opinar sobre una muestra indicando el TipoDeOpinion y la fecha de creación de la misma para que nuevamente se delegue a su nivel la interacción con la muestra para que este determine las condiciones necesarias para opinar en base al nivel mismo del participante y a la información que provee la muestra.
En el mensaje opinar delegado a la superclase Nivel, se verifica que se cumplan ciertas condiciones necesarias para que el participante opine tales como, si ya fue opinada por el mismo participante, si puede opinar en base a los niveles de opinión de otros participantes y si la muestra fue verificada. De cumplirse estas condiciones (una de ellas siendo abstracta) se agregaría la opinión al participante y a la muestra. Además, se delegaría al estado que posea el participante la actualización del mismo (siguiendo el patrón State) en base a la actividad anterior del participante tal como se hace en la recolección de muestras.

Para saber cual es el resultado actual de una muestra, se delega al EstadoDeMuestra que posea la muestra y luego determinar cuales son las opiniones validas, este cuente cuantas veces se opinó de una muestra por cada TipoDeOpinion existente y devuelve el TipoDeOpinion que más veces haya aparecido en la lista de opiniones de la muestra, en caso de empate se devuelve “null”. Los estados de muestra existentes implementarán de distinta manera el método de resultado actual basándose en los niveles de opiniones realizados hasta el momento en la muestra.
Además, el EstadoDeMuestra se encarga de verificar o actualizar el mismo (siguiendo el patrón de diseño state) según la nueva opinión recibida.

Para lograr la comunicación entre las muestras y las zonas a las que pertenecen las mismas (para posteriormente comunicar a otras clases) tomamos la decisión de crear la clase SistemaVinchucas que posea las zonas existentes a las que se les avisará de la creación y la verificación de las muestras.
SistemaVinchucas es único en todo el sistema (siguiendo la teoría del patron Singleton), por lo tanto todas las muestras creadas avisaran a una única instancia de esta clase.

Cuando un participante crea una nueva muestra o la misma es verificada, se notifica a las zonas de cobertura existentes a las cuales les corresponda según la ubicación. De esta comunicación se encarga la clase SistemaVinchucas, la cual tiene la responsabilidad de enviar notificaciones diferentes según lo que suceda con las muestras de interés.
Una vez notificada la zona, se avisará a su vez a las organizaciones que estén trabajando con dicha zona sobre la creación o verificación de una determinada muestra.
Luego de recibir la notificación, la Organizacion le delegará a la FuncionalidadExterna correspondiente, en base a la notificación específica que haya recibido de parte de la zona, la ejecución del mensaje nuevoEvento, que implementa la funcionalidad en si mismo.
Esta serie de pasos fue pensada en base al patrón de diseño Observer y Strategy.