**Taller patrones creacionales**



Laboratorio de ingeniería de software 2

Presentado por:

Isabela Mosquera Fernández

Alejandro Cardenas

Profesor:

Julio Ariel Hurtado

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de Sistemas

Popayán, noviembre

**1.Patrón prototype**

**¿Cómo lograr que los objetos clonados no compartan referencias no deseadas?**

Este problema se resuelve implementando un clonado profundo (deep clone) en lugar de un clonado superficial (shallow clone) ya que el clonado profundo  , no solo se copia el objeto principal, sino también los objetos internos que están referenciados.Esto asegura que el objeto clonado tenga sus propias copias de los objetos internos, en lugar de compartirlos con el prototipo original.

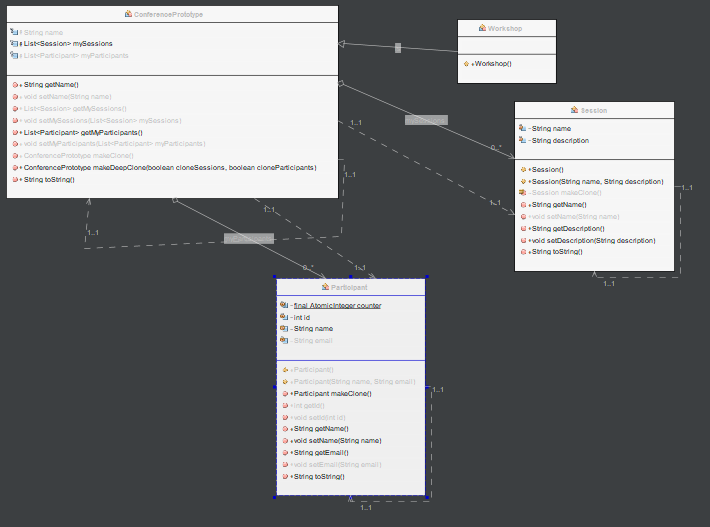
**¿En qué escenarios podría resultar útil clonar una conferencia?**En los siguientes escenarios es útil el patrón prototype:

 -Una conferencia se organiza anualmente (o periódicamente) y gran parte de la estructura, como el nombre, los temas principales, el lugar, y los organizadores, permanece constante.

-Una conferencia internacional tiene versiones regionales (por ejemplo, en América, Europa, y Asia) que comparten temas principales pero difieren en ubicación y detalles específicos.

-Clonar la conferencia como respaldo para poder recuperar la versión original en caso de que algo salga mal.

-Se desea adaptar una conferencia existente para audiencias específicas (por ejemplo, académicos, profesionales de la industria, o estudiantes) con pequeños cambios en los temas o actividades.



**2.Patrón Builder**

**¿Qué ventaja tiene construir una evaluación paso a paso en lugar de con un constructor tradicional?**

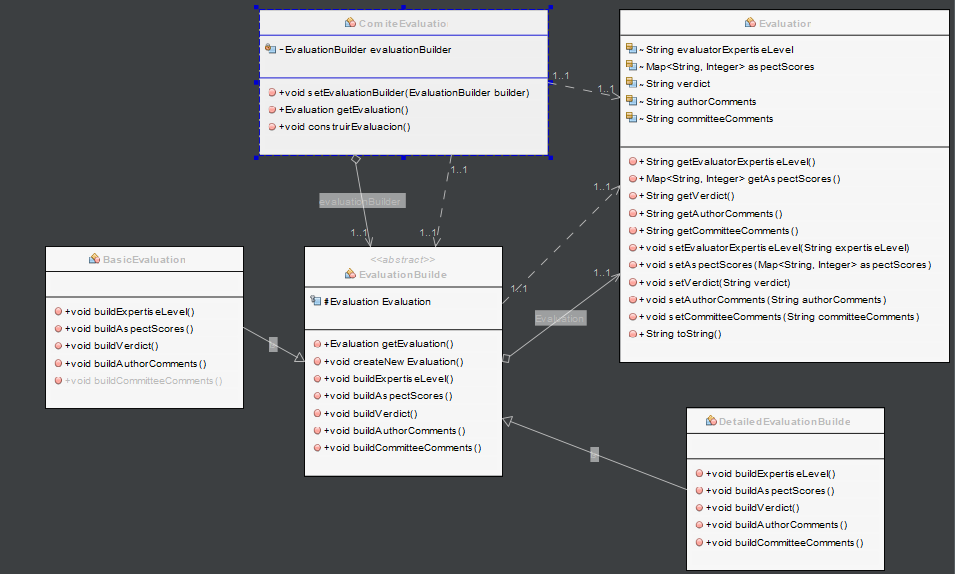
Legibilidad: Facilita entender qué atributos se están configurando, ya que cada paso del proceso de construcción utiliza nombres explícitos para los parámetros.

Flexibilidad: Permite crear objetos con configuraciones opcionales o por etapas, sin necesidad de usar múltiples sobrecargas de constructores.

Mantenibilidad: Si se agregan nuevos atributos o cambian los requerimientos, el patrón Builder facilita estas modificaciones sin afectar el código existente que utiliza la clase.

Validación: Es posible incorporar verificaciones durante el proceso de construcción para asegurar que el objeto final cumple con las reglas esperadas

**¿Cómo podrías implementarse el constructor para que sea más intuitivo para otros desarrolladores?**Implementado como en el proyecto donde hay una entidad especial encargada de crear la conferencia con el build.



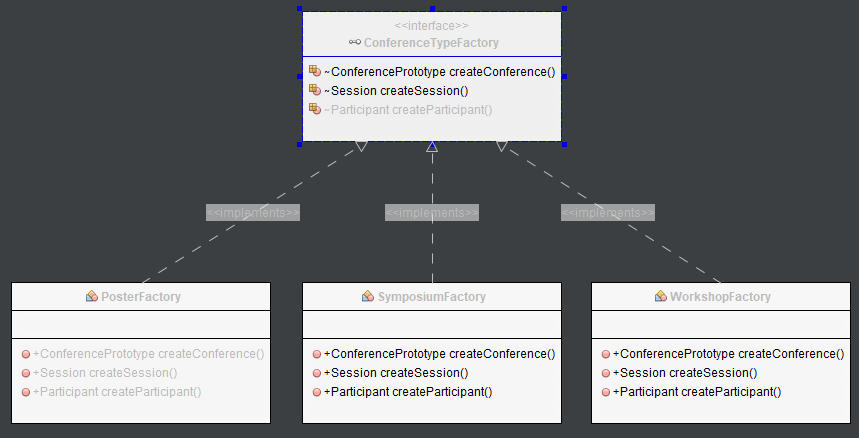
**3.Patrón abstractFactory**

**¿Cómo se manejaría el cambio si hay que agregar un nuevo tipo de conferencias en el sistema (Por ejemplo, una MultiConference)?**

Se define una clase MultiConferenceFactory que implemente la interfaz ConferenceTypeFactory. Esta clase proporcionará las implementaciones específicas de:

* createConference() para crear una MultiConference.
* createSession() para las sesiones únicas de MultiConference.
* createParticipant() para los participantes especializados en este tipo de conferencia.

**¿Cuáles serían los beneficios de utilizar una fábrica abstracta en lugar de condicionales para manejar los tipos de conferencia?**Los beneficios de utilizar una fábrica abstracta en lugar de condicionales para manejar los tipos de conferencia son que al eliminar grandes estructuras de control condicionales (como if-else o switch), el sistema se vuelve más escalable y fácil de extender. Cada nuevo tipo de conferencia se incorpora mediante una nueva fábrica concreta, sin necesidad de modificar el código existente. Esto sigue el Principio de Abierto/Cerrado (OCP) del diseño de software, que establece que el código debe estar abierto a extensiones pero cerrado a modificaciones. Como resultado, el diseño se vuelve más robusto, flexible y menos propenso a errores, ya que permite integrar nuevos tipos de conferencias sin afectar las partes ya implementadas del sistema.

****

**Enlace repositorio de github:** **https://github.com/isa0813/PatronesCreacionales.git**