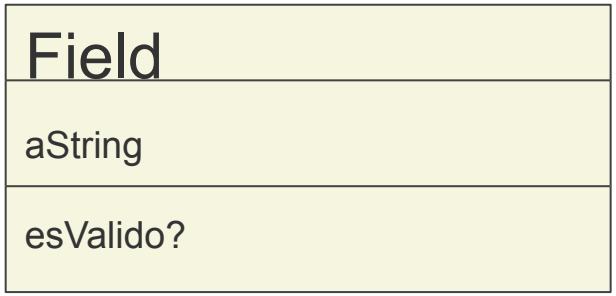
Problema: Validación de Strings

- Supongamos que estamos diseñando un formulario para ingresar datos.
- Los input fields deben ser validados, pero la validación depende del dominio del texto ingresado
  - Teléfono.
  - Número entero.
  - E-mail.
  - Dirección.
  - ...
- No podemos preveer de antemano todas las posibles formas de validación.

Solucion

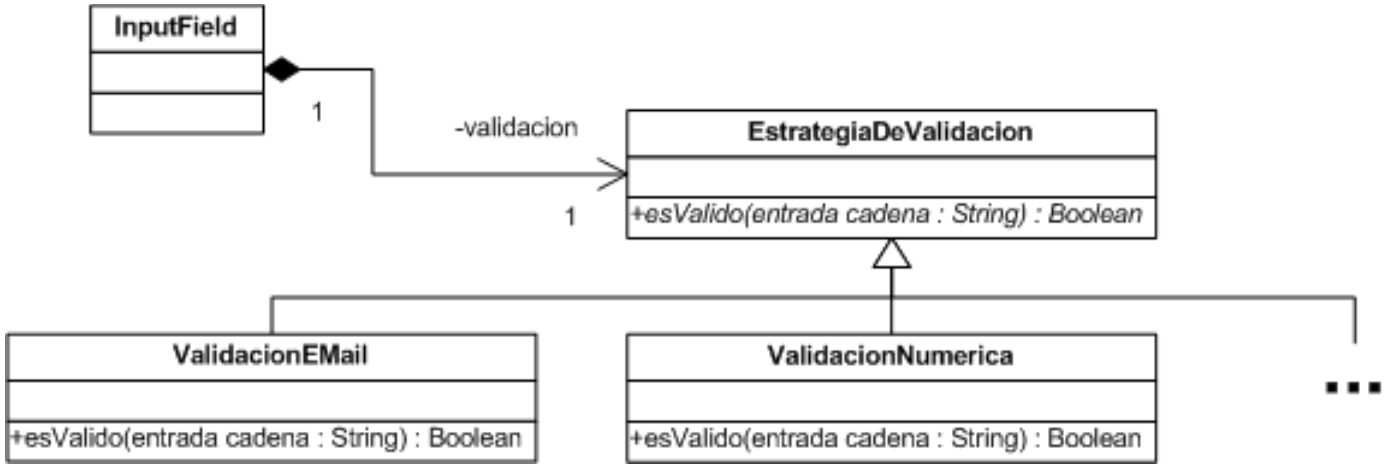
- ✓ En la clase donde esta el String, por ejemplo Field, chequear que se esta ingresando….



Problemas?

Validación de Strings

- ✓ Solución: Encapsular el algoritmo de validación en un objeto.



Patrón Strategy

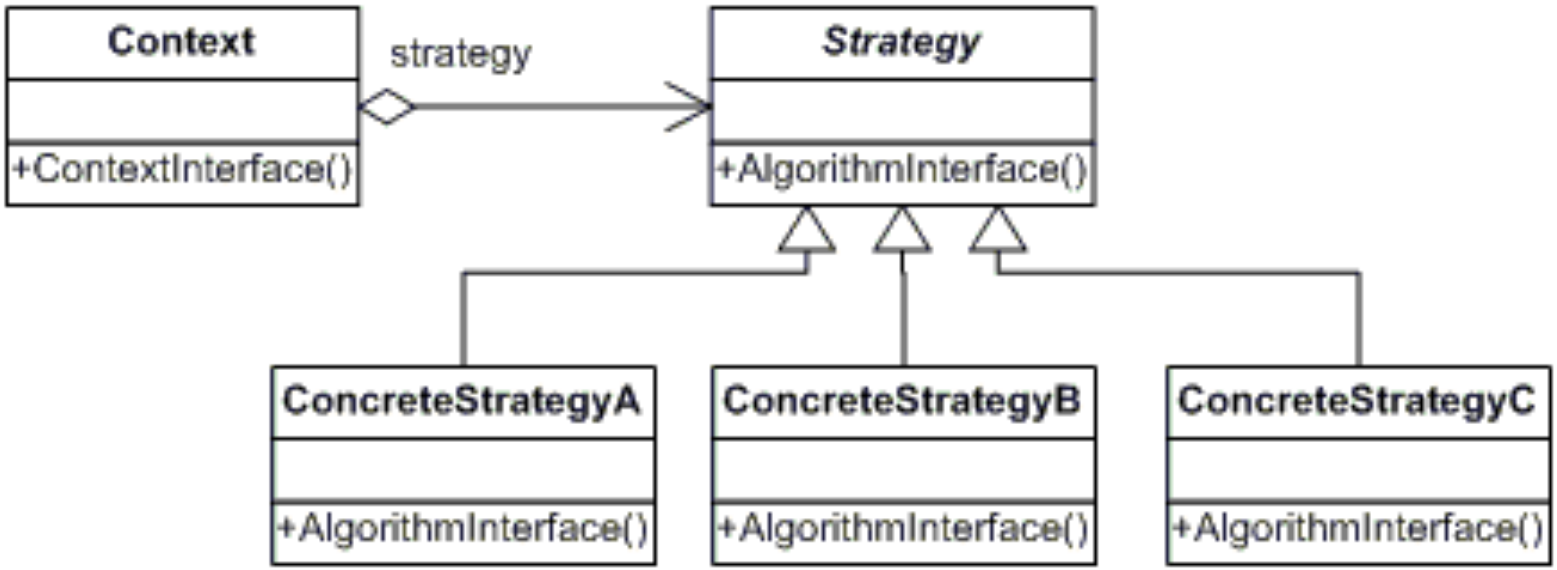
- **Intent:**
  - Desacoplar un algoritmo del objeto que lo utiliza.
  - Permitir cambiar el algoritmo que un objeto utiliza en forma dinámica.
  - Brindar flexibilidad para agregar nuevos algoritmos que lleven a cabo una función determinada.

Patrón Strategy

- **Applicability:**
  - Existen muchos algoritmos para llevar a cabo una tarea.
  - No es deseable codificarlos todos en una clase y seleccionar cual utilizar por medio de sentencias condicionales.
  - Cada algoritmo utiliza información propia. Colocar esto en los clientes lleva a tener clases complejas y difíciles de mantener.
  - Es necesario cambiar el algoritmo en forma dinámica.

Patrón Strategy

- **Solución:**
  - Definir una familia de algoritmos, encapsular cada uno en un objeto y hacerlos intercambiables.



Patrón Strategy

- **Consecuencias:**
  - \+ Alternativa a subclasificar el contexto, para permitir que se pueda cambiar dinámicamente.
  - \+ Desacopla al contexto de los detalles de implementación de las estrategias.
  - \+ Se eliminan los condicionales.
  - – Overhead en la comunicación entre contexto y estrategias.
  - – Los clientes deben conocer las diferentes estrategias para poder elegir.

- **Implementación**:
  - El contexto debe tener en su protocolo métodos que permitan cambiar la estrategia
  - Parámetros entre el contexto y la estrategia