

Patrons de Conception

-

Strategy

Simon Urli
urli@i3s.unice.fr

Master I MIAGE
2014-2015

Objectifs

“ Définir une hiérarchie de classes pour une famille d’algorithmes, encapsuler chacun d’eux et les rendre interchangeables.

Les algorithmes varient indépendamment des clients qui les utilisent.”

Classification : patron de comportement

Synonyme : policy

Exemple

The image shows a web form with several fields, each with a red error message and a warning icon. The fields and their errors are:

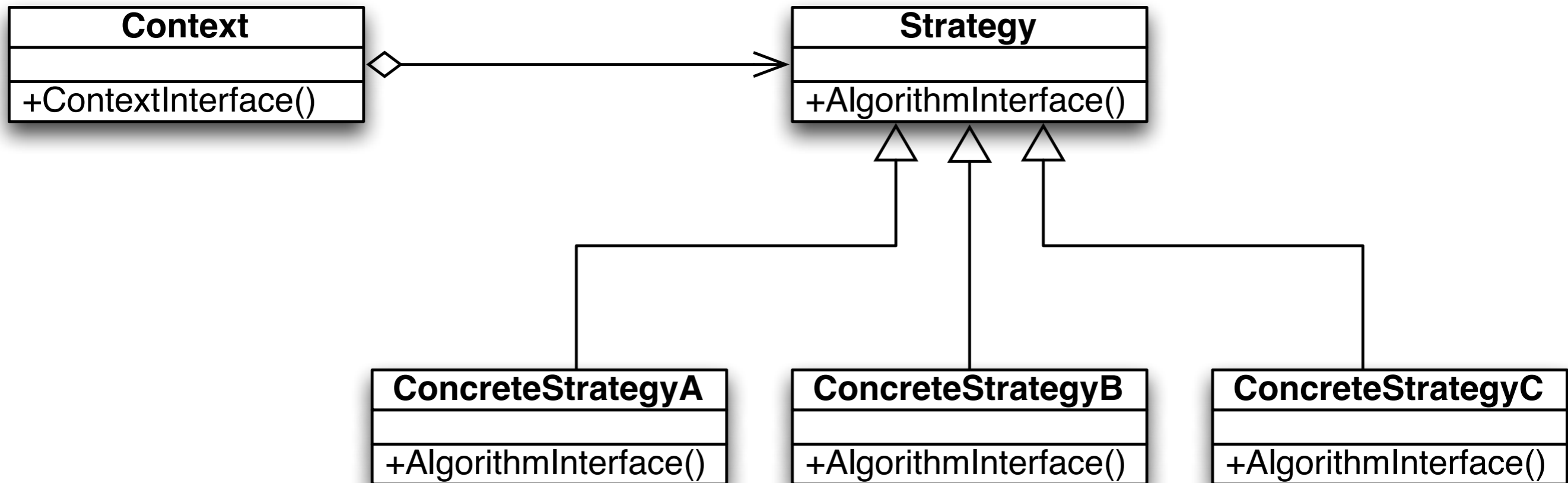
- First Name:** Invalid Name
- Email Address:** Invalid Email Address
- Phone Number:** 10 Digits for Phone Required (Note: 10 digits, no dashes)
- Password:** Password of 4 or more characters required
- Verify Password:** (Empty field)
- Date of Birth:** YYYY for year required (Month: 1-Jan, Day: 01)
- Weight:** Invalid weight (Unit: lb)

At the bottom of the form, there are three buttons: jVal POD validation, jVal COVER validation, and jVal BLANK validation.

Applications

- De nombreuses classes associées ne diffèrent que par leur comportement.
- Besoin de plusieurs variantes pour un même algorithme.
- Un algorithme utilise des données que les clients ne doivent pas connaître.
- Une classe définit plusieurs comportements.

Structure



Implémentation 1/2

- Le contexte maintient une référence à l'objet `strategy` et peut définir une interface pour lui permettre d'accéder à ses données.
- `Strategy` définit une interface commune à tous les algorithmes.
- `ConcreteStrategy` implémente une version de l'algorithme.

Implémentation 2/2

1. le client (absent de la structure) crée la ou les stratégies concrètes
2. puis il donne la classe au contexte (lors de la construction du contexte ou par un setter)
3. enfin le client interagit avec le contexte
4. lorsque le contexte reçoit une requête qui nécessite l'utilisation de la stratégie, la requête délègue à la stratégie cette requête.

Exemple 1/3

```
public class Field implements IField {  
    private String data;  
    private Validator validator;  
    public Field(Validator v) {  
        this.validator = v;  
    }  
    public void setData(String data) {  
        if (this.validator.validate(data)) {  
            this.data = data;  
        }  
    }  
}
```


Exemple 2/3

```
public abstract class Validator {  
    protected boolean testDataAgainstRegex(String  
data, String regex) {  
        Pattern p = Pattern.compile(regex);  
        Matcher m = p.matcher(data);  
        return m.matches();  
    }  
  
    public abstract boolean validate(String  
data);  
}
```

Exemple 3/3

```
public class StringValidator extends Validator {
    public boolean validate(String data) {
        return true;
    }
}

public class IntegerValidator extends Validator {
    public boolean validate(String data) {
        return this.testDataAgainstRegex(data, "[0-9]+");
    }
}

public class YearValidator extends Validator {
    public boolean validate(String data) {
        return this.testDataAgainstRegex(data, "[0-9]{1-4}");
    }
}
```

Exercice

- Modéliser une évaluation universitaire. En fonction de l'entité évaluatrice, l'université pourra être évaluée sur son budget (déficit ou bénéfice), ses résultats scientifiques (nombre de publications) ou ses résultats d'enseignements (nombre d'embauches après 3 ans).