### **Design Patterns**

Um guia para aplicar de forma correta

#### Motivação

- Código complexo e obscuro
- Classes e métodos gigantes
- Mudança em vários pontos do código
- Classes invejosas
- Todo software vai mudar (Novidades)
- Mudanças não planejadas

```
function register()
if (!ompty($_POST)) {
    Smag = '';
     if (S POST['user name']) {
        if ($ POST['user password new']) {
             if ($ POST['user_password_new'] === $ POST['user_password_repeat']) {
                 if (strlen($ POST['user_password_new']) > 5) {
                    if (strlen($ POST['user_name']) < 65 && strlen($ POST['user_name']) > 1) {
                         if (preg_match('/"[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                             Suser = read user($ POST['user name']);
                            if (!isset(@user['uner_name'])) {
                                if ($ POST['unor_email']) (
                                    if (strlen($_POST['user_email']) < 65) {
                                        if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                             S SESSION['mag'] = 'You are now registered so please login';
                                            header('Location: ' . S_SERVER['PHP_SELF']);
                                         } else Smag = 'You must provide a valid enail address';
                                    } else $msg = 'Enail must be less than 64 characters';
                                } else Smsq = 'Email cannot be empty';
                            ) else Smsq - 'Username already exists';
                        ) else Smag = 'Unorname must be only a-z, A-2, 0-9';
                     } else $mag = 'Daername must be between 2 and 64 characters';
                 } else $mag = 'Password must be at least 6 characters';
            } else Smsg = 'Passwords do not match';
        } else Smsq = 'Empty Password';
     } else $msg = 'Empty Username';
     S SESSION['mag'] - Smag;
 return register_form();
```

### O que são Design Patters (Padrões de projeto)?

# Solução elegante para problemas que são recorrentes

(Eu explico!)



#### 1. Introdução

Quantos padrões ou filosofias você precisa saber para para escrever um bom código?

#### \_\_

# Apenas dois princípios! A alta coesão e o baixo acoplamento.

(Eu explico!)

Alta coesão, a classe tem poucos **OBJETIVOS** 



baixo acoplamento, a classe tem poucas

#### **DEPENDÊNCIAS**





#### 2. Alguns Padrões



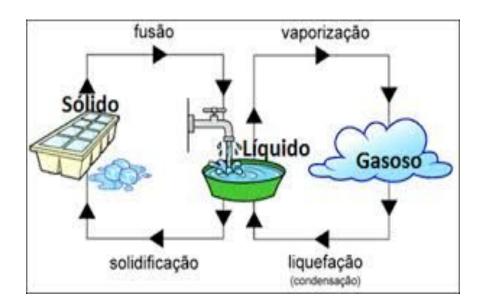
#### strategy pattern

Para uma mesma ação gera comportamentos diferentes.



## TemplateMethod pattern

Define um esqueleto para o desejado.



#### **State pattern**

Define um estado para o pattern



#### **Observer pattern**

Conjunto de ações executadas após uma ação principal.



#### **Decorator pattern**

Encapsula a regra de outro objeto.



#### **Builder pattern**

Criar um objeto com uma interface fluente.



#### Adapter pattern

Transforma um objeto em outro



#### Bridge pattern

Criar uma ponte entre dois sistemas



#### **Muito Obrigado!**

Espero que vocês apliquem essas dicas no seu dia a dia :)