KISS - Observer

Grupo: Dener, Henrique, Mathias, Rita

KISS - Origem e História

→ “If you can’t explain it simply, you don’t understand it well enough.” - Einstein, Albert

→ Frase criada por Kelly Johnson (Engenheiro Chefe da Lockheed Skunk Works)

→ Seus projetos deveriam ser simples o suficiente para serem reparados por um homem em uma situação de combate com apenas alguns treinamentos mecânicos básicos e ferramentas simples.

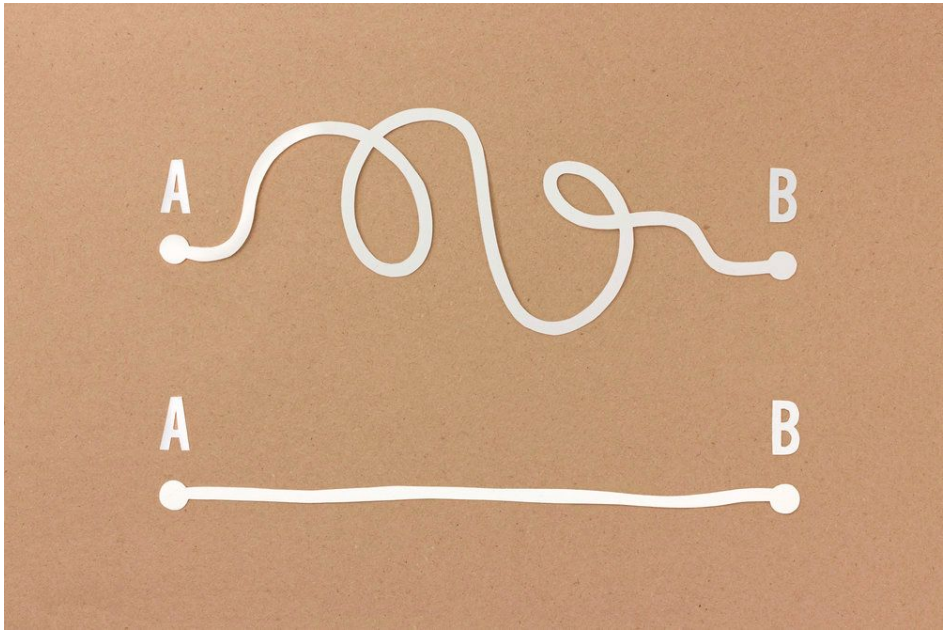
→ Caso contrário estes ficariam obsoletos rapidamente e custam vidas.

Para que ele serve?

→ Criar software enxuto e mais leve.

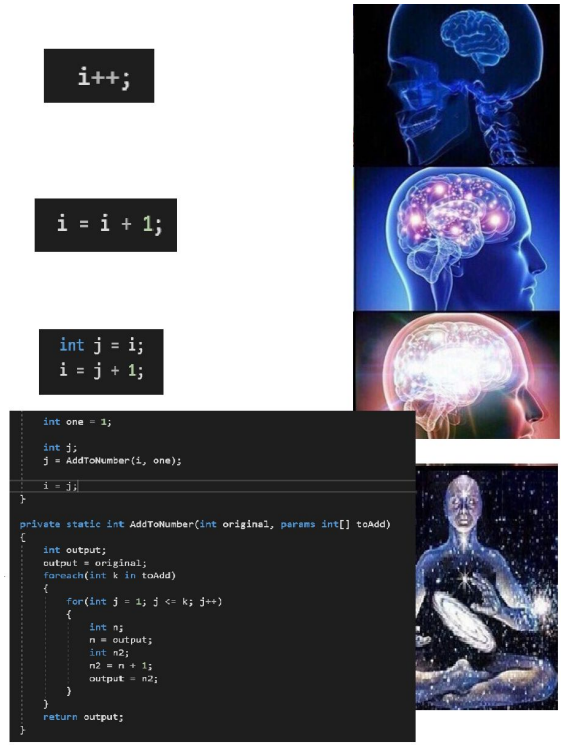
→ Não exige que a curva de aprendizado do usuário seja grande, melhorando então a experiência do usuário ao utilizar a aplicação.

→ Facilitar a manutenção do software.



Exemplos


que tal colocar uma bussola no site?
Cliente explicando feature mirabolante sem levar em consideração se é útil ou não
Dev esperando ele terminar pra mostrar que não é uma boa ideia
MemeCode
30 de julho ·
Curtir Página
lembro uma vez q sugeriram um campo de senha giratório igual de um cofre


i++;
i = i + 1;
int j = i;
i = j + 1;
int one = 1;
int j;
j = AddToNumber(i, one);
i = j;
}
private static int AddToNumber(int original, params int[] toAdd)
{
int output;
output = original;
foreach(int k in toAdd)
{
for(int j = 1; j <= k; j++)
{
int n;
n = output;
int n2;
n2 = n + 1;
output = n2;
}
}
return output;
}

Quando devo aplicá-lo?

→ Design de interface

→ Design de produto

→ Desenvolvimento do software.

Observer

→ O Observer é uma solução para casos em que múltiplas fontes dependem da mesma informação que sempre deve estar atualizada.

→ Sem o uso do observer todas as múltiplas fontes precisam monitorar constantemente o dado que precisa ser mantido atualizado.

→ O Observer propõe um intermediário que notifique as fontes que precisam da informação para que eles atualizem a mesma.

```
class Observable{

    constructor(){
        // Array de observadores que serão notificados com a mudança dos dados
        this.observers = [];
    }

}
```

```
subscribe(Observador){
    //Adiciona a função de notificação do Observador
    this.observers.push(Observador);
}

unsubscribe(Observador){
    //Cria um novo arrai de observadores
    let observadores = [];
    for(let i =0;i<this.observers.length;i++){
        //Procura os observadores a não serem removidos
        if(this.observers[i] != Observador){
            //Adiciona os observadores que ainda devem ficar no array
            observadores.push(this.observers[i])
        }
    }
    // Atualiza o array de observadores
    this.observers = observadores;
}
```

```
notify(valor){
    //Loop por todos os observadores
    for(let i =0;i<this.observers.length;i++){
        //Chama a função de notificar cada um
        this.observers[i](valor);
    }
}
```

```
//Inicia Variaveis
const observable = new Observable();
const input = document.getElementById("input");
let a = 0;
let b = 0;
let c = 0;
```

```
// Cria as funções do observable
const updateA = function(data){
    a = data;
}
const updateB = function(data){
    b = data;
}
const updateC = function(data){
    c = data;
}
```

```
// Inscreve as funções de observable
observable.subscribe(updateA)
observable.subscribe(updateB)
observable.subscribe(updateC)
```

```
// Notifica todos os observables da mudança de dados
input.addEventListener('keyup', function(e){
    let texto = e.target.value
    observable.notify(texto);
    console.table({
        a:a,
        b:b,
        c:c
    });
})
```

Demonstração

Referências:

→ Slides da disciplina

→ https://uxdesign.blog.br/a-origem-do-keep-it-simple-stupid-kiss-b24085dc1327 (Texto acessado em 12 de setembro de 2018)

→ https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle (Texto acessado em 12 de setembro de 2018)

→ https://www.techopedia.com/definition/20262/keep-it-simple-stupid-principle-kiss-principle (Texto acessado em 12 de setembro de 2018)