

IPO Model

O, en español,

Modelo EPS

Que es EPS?

ENTIDAD PROMOTORA DE SALUD - EPS

Entidades responsables de la afiliación y registro de los afiliados al sistema de la regularidad social en Colombia.

Que es el IPO Model?

Es un patron reconocido para el diseño y desarrollo de algoritmos.

IPO significa...

INPUT

PROCESS

OUTPUT

EPS significa...

ENTRADA

PROCESO

SALIDA

Es decir,

Para desarrollar un algoritmo que solucione un problema (pequeño),

Paso 1: Obtener los datos necesarios (input)

Paso 2: Procesar o modificar los datos (process)

Paso 3: Devolver un resultado (output)

Ejemplo

```
function suma(a, b) {  
  const operando1 = a;  
  const operando2 = b;  
  
  const resultado = operando1 + operando2;  
  
  return resultado;  
}
```


Ejemplo

```
function suma(a, b) {  
  // input  
  const operando1 = a;  
  const operando2 = b;  
  
  // process  
  const resultado = operando1 + operando2;  
  
  // output  
  return resultado;  
}
```

Ejemplo

```
function suma(a, b) {  
  // input  
  // const operando1 = a;  
  // const operando2 = b;  
  
  // process  
  const resultado = a + b;  
  
  // output  
  return resultado;  
}
```

Ejemplo

```
function suma(a, b) {  
  // input  
  // const operando1 = a;  
  // const operando2 = b;  
  
  // process  
  // const resultado = a + b;  
  
  // output  
  return a + b;  
}
```

Problema

Dada una lista de estudiantes y su nota final, cuantos estudiantes sacaron la mayor nota, que no es necesariamente la nota maxima posible (5)?

Problema

```
function suma(estudiantes) {  
  // input  
  .  
  .  
  
  // process  
  .  
  .  
  .  
  
  // output  
  .  
}
```

Problema

```
function suma(estudiantes) {  
  // input  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  // process  
  .  
  .  
  .  
  
  // output  
  .  
}
```

Problema

```
function suma(estudiantes) {  
  // input  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  // process  
  const estudiantesConMayorNota = estudiantes.filter(  
    (estudiante) => estudiante.notaFinal === mayorNota  
  );  
  
  // output  
  .  
}
```

Problema

```
function suma(estudiantes) {  
  // input  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  // process  
  const estudiantesConMayorNota = estudiantes.filter(  
    (estudiante) => estudiante.notaFinal === mayorNota  
  );  
  
  // output  
  return estudiantesConMayorNota.length;  
}
```


Y si las entradas son invalidas?

```
function suma(estudiantes) {  
  // ??  
  if (estudiantes.length === 0) {  
    return 0;  
  }  
  
  // input  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  // process  
  const estudiantesConMayorNota = estudiantes.filter(  
    (estudiante) => estudiante.notaFinal === mayorNota  
  );  
  
  // output  
  return estudiantesConMayorNota.length;  
}
```

Y si las entradas son invalidas?

```
function suma(estudiantes) {  
  // early return  
  if (estudiantes.length === 0) {  
    return 0;  
  }  
  
  // input  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  // process  
  const estudiantesConMayorNota = estudiantes.filter(  
    (estudiante) => estudiante.notaFinal === mayorNota  
  );  
  
  // output  
  return estudiantesConMayorNota.length;  
}
```

Early Return

Los `else` son, hasta cierto punto, considerados "malos".

Porque?

1. Si la condicion del `if` es complicadita, es complicado entender cuando cae el `else`.
2. Si el codigo dentro del `if` es considerable, es facil de olvidar cual era la condicion.

Alternativa?

```
function suma(estudiantes) {  
  if (estudiantes.length !== 0) {  
    // input  
    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
    const mayorNota = Math.max(...notasFinales);  
  
    // process  
    const estudiantesConMayorNota = estudiantes.filter(  
      (estudiante) => estudiante.notaFinal === mayorNota  
    );  
  
    // output  
    return estudiantesConMayorNota.length;  
  } else {  
    return 0;  
  }  
}
```

Y si son varias condiciones?

Terminamos con una flecha. (anti-patron)

Flecha

```
function proceso(param1, param2) {  
  if (isValid(argument1)) {  
    if (isValid(argument2)) {  
      const otherVal1 = doSomeStuff(param1, param2);  
  
      if (isValid(otherVal1)) {  
        const otherVal2 = doAnotherStuff(otherVal1);  
  
        if (isValid(otherVal2)) {  
          return "Stuff";  
        } else {  
          throw new Error();  
        }  
      } else {  
        throw new Error();  
      }  
    } else {  
      throw new Error();  
    }  
  } else {  
    throw new Error();  
  }  
}
```

```
public String returnStuff(SomeObject argument1, SomeObject argument2){  
    if (!argument1.isValid()) {  
        throw new Exception();  
    }  
  
    if (!argument2.isValid()) {  
        throw new Exception();  
    }  
  
    SomeObject otherVal1 = doSomeStuff(argument1, argument2);  
  
    if (!otherVal1.isValid()) {  
        throw new Exception();  
    }  
  
    SomeObject otherVal2 = doAnotherStuff(otherVal1);  
  
    if (!otherVal2.isValid()) {  
        throw new Exception();  
    }  
  
    return "Stuff";  
}
```

Early Return

```
function suma(estudiantes) {  
  if (estudiantes.length === 0) {  
    return 0;  
  }  
  
  const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
  const mayorNota = Math.max(...notasFinales);  
  
  const estudiantesConMayorNota = estudiantes.filter(  
    (estudiante) => estudiante.notaFinal === mayorNota  
  );  
  
  return estudiantesConMayorNota.length;  
}
```