

# Trabalho Final

Heleny Bessa e Rithie Natan

# Linguagem Orientada a Objetos

A yellow square containing the letters 'JS' in a bold, black, sans-serif font, representing JavaScript.

**JS**


# Abstração

Representação de um objeto real usando identidades, propriedades e métodos

```
/*----- Class Pessoa -----*/  
class Pessoa  
{  
    /**  
     * Default constructor  
     *  
     * @param {string} name  
     * @param {number} age  
     */  
    constructor(name, age)  
    {  
        this.name = name;  
        this.age = age;  
    } //end constructor()  
  
    value = 10;  
} //end class
```

# Encapsulamento

Caixa preta

```
function main() 
{
  let array = new Array();

  for(let i = 0; i < TestsP.length; i++)
  {
    array.push(new Pessoa(TestsP[i].nome, TestsP[i].idade, TestsP[i].role));
  }

  for(let i = 0; i < TestsC.length; i++)
  {
    array.push(new Celebridade(TestsC[i].nome, TestsC[i].idade, TestsC[i].role));
  }

  console.log(array);
} //end main()
```

# Herança

Reuso de código

```
/*----- Class Celebridade -----*/  
class Celebridade extends Pessoa  
{  
    /**  
     * Default constructor  
     *  
     * @param {string} name  
     * @param {number} age  
     * @param {string} role  
     */  
    constructor(name, age, role)  
    {  
        super(name, age);  
  
        this.role = role;  
    } //end constructor()  
  
    value = this.value + 5;  
} //end class
```

# Polimorfismo

Alteração do funcionamento interno de um método herdado de um objeto pai

```
class Pessoa
{
  /**
   * Default constructor
   *
   * @param {string} name
   * @param {number} age
   */
  constructor(name, age)
  {
    this.name = name;
    this.age = age;
  } //end constructor()
}
```

```
  value = 10;
```

```
} //end class
```

```
class Celebidade extends Pessoa
```

```
{
  /**
   * Default constructor
   *
   * @param {string} name
   * @param {number} age
   * @param {string} role
   */
  constructor(name, age, role) ...
} //end constructor()
```

```
  value = this.value + 5;
```

```
} //end class
```

# Vantagens

- confiável: ao alterar uma parte nenhuma outra é afetada
- oportuno: partes podem ser desenvolvidas em paralelo
- flexível: fácil de estender e de manter
- fácil manutenção
- reutilizável
- menor esforço de codificação

# Desvantagens

- Complexidade no aprendizado para desenvolvedores de linguagens estruturadas
- Maior uso de memória (heap), por exemplo para aplicações móveis em JavaME
- Maior esforço na modelagem de um sistema OO do que estruturado (porém menor esforço de codificação, sendo uma vantagem)
- Funcionalidades limitadas por interface, quando estas estão incompletas (problemas na modelagem)
- Dependência de funcionalidades já implementadas em superclasses no caso da herança, implementações espalhadas em classes diferentes



Obrigado!