

Programação Orientada a Objetos



Orientação a copiar e colar

- Existe algo implicitamente errado nisso?
- Quem foi o ordinário que inventou que isso era errado?
- A palavra chave na programação: “ALTERAÇÃO”

Glossário

- Estado
- Instância/Exemplar
- Abstração
- Modularidade



Filosofia 0.0.



Pascal e os Mundos das ideias



Pensando num mundo ideal...

- E se existisse uma máquina onde bastasse eu configurar, ela geraria quantos objetos eu desejasse?



*Programação orientada a objetos é um paradigma que usando objetos – **estrutura de dados** consistindo de **atributos** e **métodos** junto com suas interações- para projetar aplicação e programas de computadores*

Wikipedia

Olhando os objetos de longeee...



- Coleção de objetos com diferentes características e comportamentos.
- Cada um resolvendo um pequeno problema
- Juntos eles interagem para resolver um grande problema

Transcendental

O comportamento do sistema é algo que emerge das interações entre os objetos

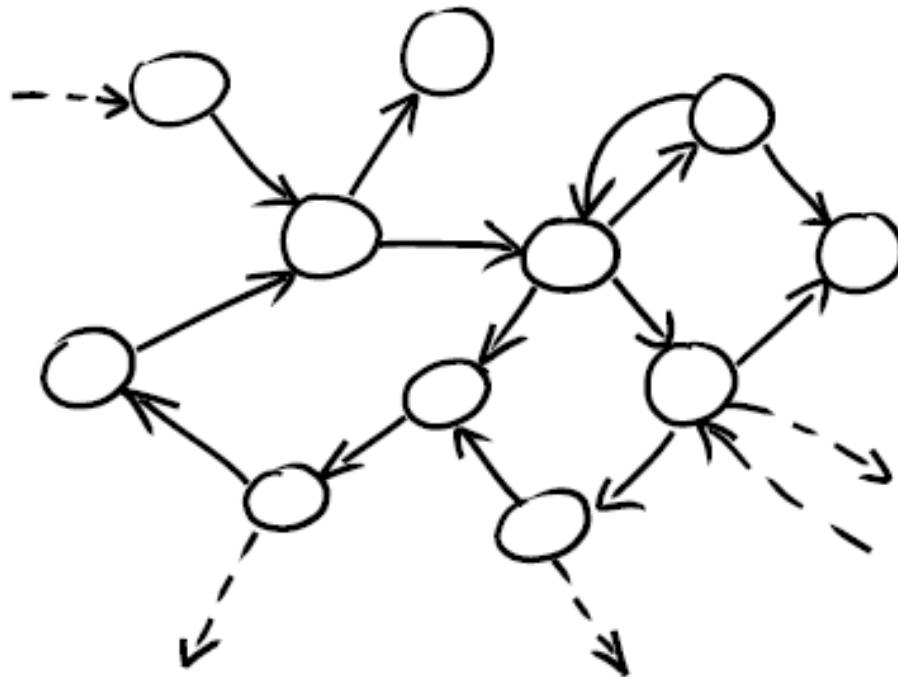


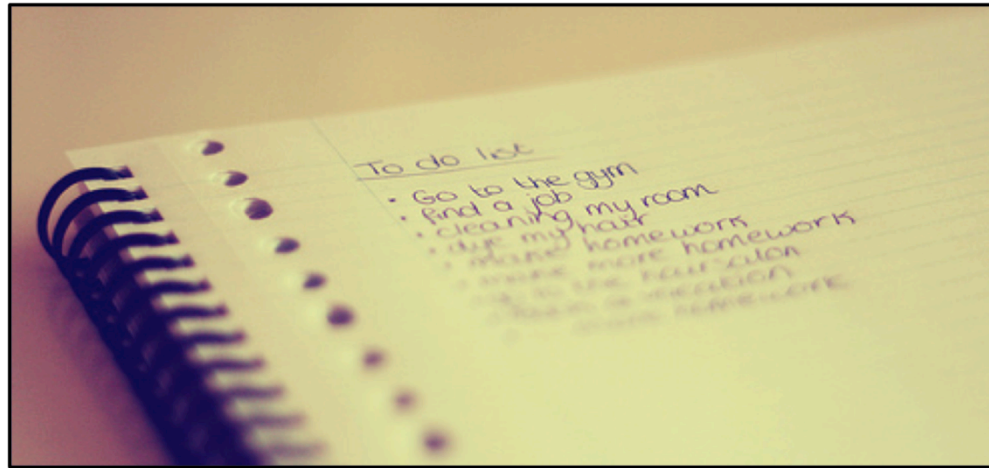
Plugando, removendo e substituindo objetos é possível mudar o comportamento do sistema.

Só me preocupo com o que o objeto faça, não como ele faz



Uma teia de objetos





Código procedural é escrito num sentido mais declarativo, definindo quem vai interagir com quem

O foco, no momento desta definição, deveria ser em “o que fazer”, não “como fazer”

Lembrando...

Nosso código deveria ser apenas declarativo

Os objetos devem ter conexões explícitas

Coesão

Faça apenas uma coisa!



Acoplamento



Nós só poderemos obter o benefício desta abordagem se os nossos objetos forem facilmente plugáveis



Do outro lado da força...



- Programas tendem a virar um longa lista de comandos
- Funções e variáveis são criadas, mas elas tendem a ser globais e serem acessadas por qualquer um

Mas o que era uma classe mesmo..

Classe é um molde que será usado para construir objetos que representam elementos da vida real.

Classe = Características + Comportamentos

Atributos

Atributos armazenam os dados de um objeto, e definem seu estado.

```
3
4 public class Pessoa {
5     private String nome;
6     private int idade;
7     private static int MAIORIDADE = 18;
8
9     public Pessoa(String nome, int idade) {
10         this.nome = nome;
11         this.idade = idade;
12     }
13
14     public void incrementaIdade() {
15         idade = idade + 1;
16
17     }
18
19     public int retornaIdade() {
20         return idade;
21     }
22
23     public int retornaNome() {
24         return nome;
25     }
26
27     public static void retornaMaioridade() {
28         return MAIORIDADE;
29     }
30 }
```

Estático e Instância

- Eu preciso de um exemplar de uma pessoa para perguntar sua idade?
- Eu preciso de um exemplar em particular para saber a maioridade?

Brincando com um objeto

```
public static void main(String args[]) {  
    Pessoa riquelme = new Pessoa("Juan Riquelme", 34);  
    Pessoa emerson = new Pessoa("Emerson Sheik", 33);  
  
    riquelme.incrementaIdade(); //Alterando estado do exemplar riquelme  
  
    System.out.println(Pessoa.retornaMaioridade());  
}
```

Igualdade entre objetos

```
class EqualsNotEqualTo {  
    public static void main(String args[]) {  
        String s1 = "Hello";  
        String s2 = new String(s1);  
  
        System.out.println(s1 + " equals " + s2 + " -> " +  
            s1.equals(s2));  
  
        System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));  
    }  
}
```

Navio de Teseu



Polimorfismo

```
public class Animal {  
    public void incomodar() {  
        System.out.println( "Animal incomodando" );  
    }  
}  
  
public class Cao extends Animal {  
    public void comer() {  
        System.out.println( "AU!" );  
    }  
}  
  
public class Gato extends Animal {  
    public void comer() {  
        System.out.println( "MIAU!" );  
    }  
}
```

Polimorfismo

```
public class Animal {  
    public void incomodar() {  
        System.out.println( "Animal incomodando" );  
    }  
}
```

```
public class Cao extends Animal {  
    public void comer() {  
        System.out.println( "AU!" );  
    }  
}
```

```
public class Gato extends Animal {  
    public void comer() {  
        System.out.println( "MIAU!" );  
    }  
}
```

```
Animal a = new Cao();  
a.incomodar();  
a = new Gato();  
a.incomodar();
```

Herança

- Clientes
- Funcionários



Referências

- <http://www.java-samples.com/showtutorial.php?tutorialid=221>
- *Design Patterns*, by Erich Gamma, et. Al
- <http://www.amazon.com/Growing-Object-Oriented-Software-Guided-Signature/dp/0321503627>
- Schach, Stephen (2006). *Object-Oriented and Classical Software Engineering, Seventh Edition*