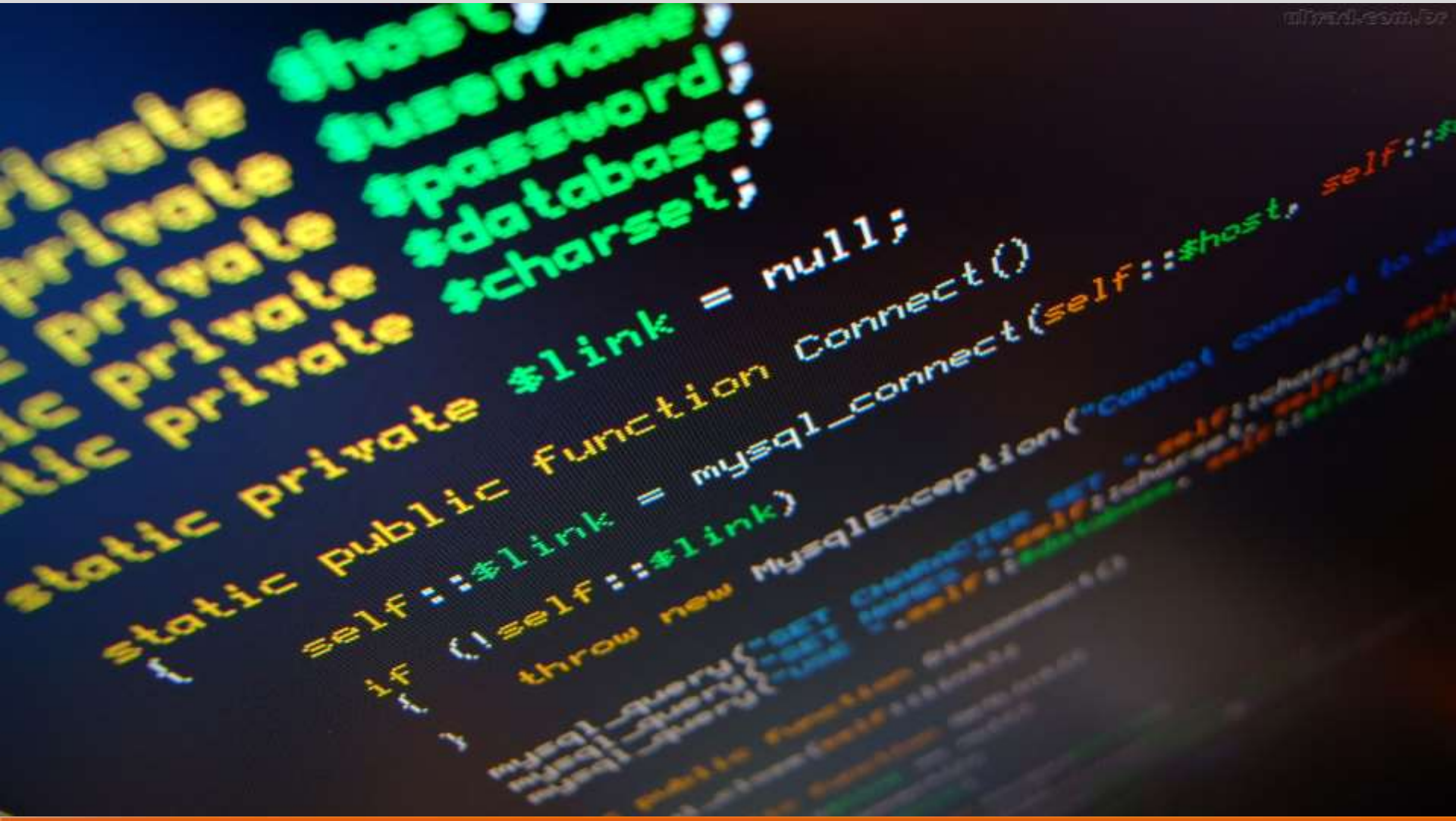
Programação Orientada a Objetos II

Prof. Walter Gima

walter.gima@anhanguera.com

Interfaces – Polimorfismo



Anhanguera

Agenda

- Interfaces
- Interfaces x Classes Abstratas
- Polimorfismo
- Herança Múltipla

Interfaces

- São estruturas de definem o padrão de atributos e métodos (modelo) que uma classe deve implementar ao utilizar uma interface.

- Interface é como um **contrato**, define regras (atributos e métodos) que as classes devem seguir.

- São muito utilizadas em grandes projetos para os desenvolvedores seguirem alguns padrões de projeto.

Interfaces

Exemplo: Em um projeto foi determinado que todas as classes DAO devem sempre implementar os métodos salvar, atualizar, deletar e etc. Para que nenhuma classe DAO seja criado sem esses métodos base deve-se implementar a interface BaseDAO em todas as classes DAO.

```
import java.util.List;

public interface BaseDAO {

    public void salvar(Object bean);
    public void atualizar(Object bean);
    public void deletar(int id);
    public Object getById(int id);
    public List<Object> getAll();
}
```

Programação Orientada a Objetos II

```
public class FuncionarioDAO implements BaseDAO {

    @Override
    public void salvar(Object bean) {
        // TODO Auto-generated method stub
    }

    @Override
    public void atualizar(Object bean) {
        // TODO Auto-generated method stub
    }

    @Override
    public void deletar(int id) {
        // TODO Auto-generated method stub
    }

    @Override
    public Object getById(int id) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public List<Object> getAll() {
        // TODO Auto-generated method stub
        return null;
    }

    //Método especifico de funcionario criado e implementado pelo próprio programador
    public void calcularSalario(){
    }
}
```

Classe FuncionarioDAO implementa interface BaseDAO.

O Desenvolvedor deve seguir o contrato e “programar” os métodos definidos pela interface.

O programador pode criar novos atributos e métodos específicos da classe.

Interfaces

- Interfaces são declaradas com a **palavra chave interface** e não com class.

- A declaração de métodos é feita apenas com sua assinatura, pois são apenas modelos a ser seguidos, não se deve implementar funcionalidade no métodos.

- Não se deve utilizar o modificador abstract nos métodos de uma interface.

Interfaces

- Uma interface não deve ter construtor.

- Métodos em interfaces não pode ter modificadores como public e static.

Interfaces x Classes abstratas

- Interface não é uma classe, é um modelo e utiliza palavra reservada interface. Classe abstrata é uma classe que utiliza modificador abstract.
- Métodos em interfaces não pode ter modificadores como public e static.
- Classe abstrata pode ter métodos abstratos (apenas assinatura) e métodos comuns com corpo (implementação).
- Métodos em interfaces não precisam ser declarados como public e abstract, pois são implícitos.
- Classes abstratas são herdadas (**extends**) por suas classes filhas. Para uma classe utilizar interface deve utilizar palavra reservada **implements**.

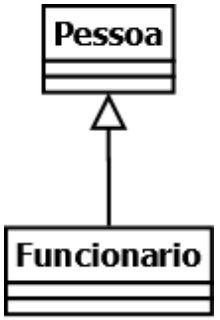
Polimorfismo

- O termo polimorfismo é originário do grego e significa "muitas formas" (poli = muitas, morphos = formas).

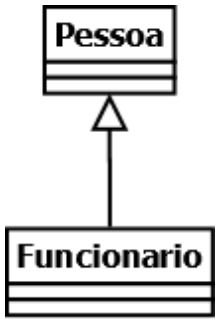
- Na programação orientada a objetos, o polimorfismo permite um objeto assumir várias formas/características/tipo ao mesmo tempo.

Exemplo:

Um objeto é do tipo funcionário mas também é do tipo pessoa.



Polimorfismo



- Exemplo:

- Pessoa é uma classe abstrata.

```
Pessoa pf = new Funcionario();
pf.nome = "Sonic";
pf.email = "sonic@sega.com.br";
```

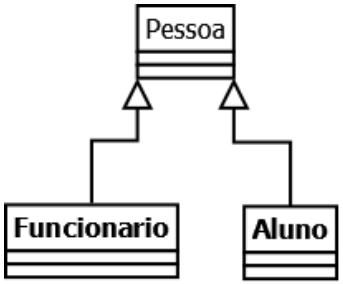
- Foi possível instanciar um objeto referenciando classe abstrata Pessoa utilizando a classe Funcionario.

Polimorfismo

- Exemplo2:

- Pessoa é uma classe abstrata.
- Conforme diagrama: Funcionario e Aluno herdam da classe abstrata Pessoa.

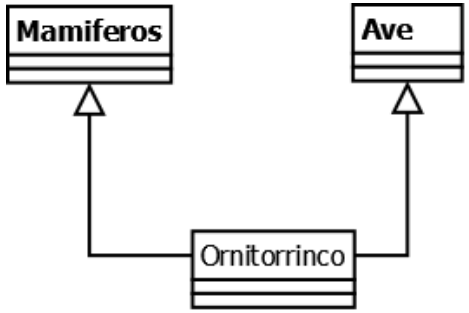
Se Criarmos um método que recebe objeto Pessoa como parametro, esse método vai aceitar objetos do tipo Funcionario e do tipo Aluno também por causa do polimorfirmo.




```
public static void mostraPessoas(Pessoa p){
        System.out.println(p.toString());
}

public static void main(String[] args) {

        Funcionario f = new Funcionario();
        f.nome = "Mario";
        f.email = "mario@armario.com.br";
        f.salario = 7000;

        Aluno a = new Aluno();
        a.nome = "Chiquinho";
        a.email = "chico@ig.com.br";
        a.RA = "121212";

        mostraPessoas(a);
        mostraPessoas(f);
}
```

Herança Múltipla em Java

- Herança múltipla é quando uma classe pode herdar características de mais de uma classe.



- Java não implementa herança múltipla nativamente, porém é possível ter resultado parecido utilizando combinação do uso de herança (extends) e interface (implements)

Exemplo:

```
public class Ornitorrinco extends Mamferos implements Ave {


}
```

Bibliografia

1) SANTOS, Rafael. Introdução à Programação Orientada a Objetos Usando Java. 2ª ed. Rio de Janeiro: Campus - Elsevier, 2013.

2) Serson,Roberto Rubinstein. Programação Orientada a Objetos com Java 6. Brasport, 2007.

3) Devmedia. http://www.devmedia.com.br/java-interface-aprenda-a-usar-corretamente/28798. Acesso em 15/11//2016.

Dúvidas ?

walter.gima@anhanguera.com