Bloco de Java
Orientação a Objetos com Java - TP1
Prof. LP Maia
Aluno: João Pedro Martins Figueredo

1. O que é uma classe?

A classe é a unidade básica de programação no paradigma de orientação-objeto. Objetos similares são agrupados em algo que chamamos de classe.

2. O que é um objeto?

Objeto é a junção entre dados e algoritmos, os dados definem o estado do objeto, os algoritmos definem o comportamento do objeto.

3. Quais os principais componentes de uma classe?

Os principais componentes de uma classe são: Atributos, métodos e construtor.

4. Qual a diferença entre atributos públicos, privados e protegidos?

Público = Visto por qualquer classe

Privado = Apenas a classe na qual ela reside consegue acessá-lo

Protegidos = Pode ser acessado pela classe e classes derivadas como, por exemplo, classes filhos.

5. Por que os atributos de uma classe devem ser privados?

Para que nenhuma das classes possa alterar o estado dos atributos que residem nessa.

6. Explique o conceito de encapsulamento?

Segmentar o objeto para, assim, gerar “independência” entre seus algoritmos, fazendo com que, caso um “quebre”, o objeto não seja totalmente comprometido, melhorando manutenção de código e experiência de usuário.

7. Para que servem os gets e sets?

São métodos padrões para disponibilização de instancias de atributos de classes.

8. Por que os gets e sets são públicos?

Porque eles servem para instanciar um atributo privado de forma pública, para que não afete a classe e, ainda assim, consigamos utilizar seus atributos.

9. Para que serve um construtor?

O construtor é responsável por instanciar a classe definida, para que possamos acessá-la.

10. Podemos ter mais de um construtor por classe? Como se chama esta técnica?

Sim, chama-se sobrecarga de construtor.

11. O que é sobrecarga de métodos? De um exemplo.

É um conceito do polimorfismo, consiste em redefinir o mesmo método, porém, com alterações em seus argumentos. Podemos, por exemplo, definir 3 métodos iguais, alterando apenas o tipo do dado (INT, Double, String) e, assim, utilizar o mesmo algoritmo para 3 inputs com tipos diferentes.

Exemplo:

Na classe “Calculadora” temos o método “somar”, então:

```
public void soma() {
    result = op1 + op2;
}

public double soma(int op1, int op2) {
    return op1 + op2;
}

public String soma(String op1, String op2) {
    return op1 + op2;
}
```

Assim, definimos 3 possíveis entradas diferentes para um mesmo método.

12. O que é sobrescrita de método? De um exemplo.

Outro conceito de polimorfismo, consiste na reescrita de um método numa classe filho, diferente da sobrecarga, deve ter o mesmo nome, tipo de retorno e quantidade de parâmetros do método inicial.

```java
public class Garrafa{

    public void Finalidade(){
            System.out.println("Garrafa genérica");
    }

}

public class GarrafaTermica extends Garrafa{

    @Override
    public void Finalidade(){
            System.out.println("Manter a temperatura");
    }

}
```

Observamos que a String muda, porém o método permanece o mesmo.

13. Quando que um método de uma classe deve ser definido como privado?

Quando a função do método for auxiliar outro método da mesma classe ou quando não queremos chamar esse método em outras classes.

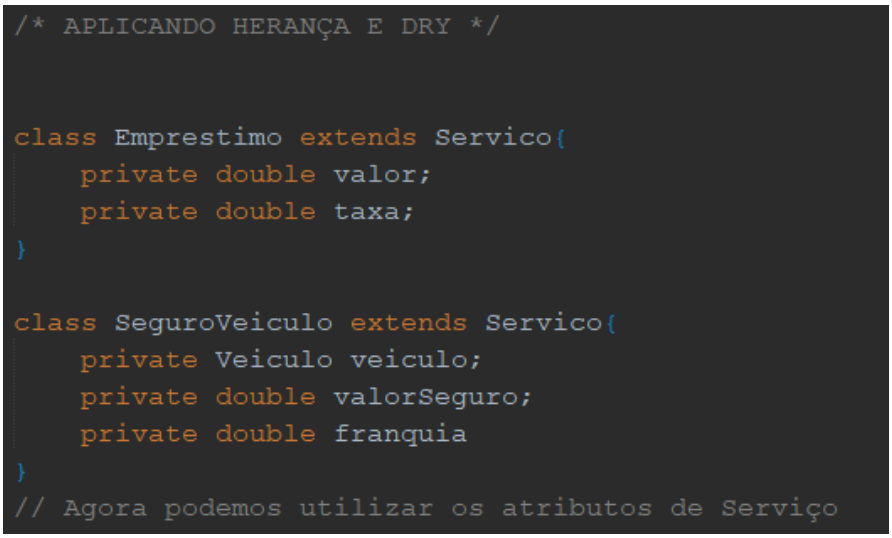
14. O que é herança?

Ideia diretamente ligada ao conceito Don't Repeat Yourself (DRY), que diz que devemos minimizar ao máximo a repetição de código. Diretamente conectada à encapsulamento, quando formos compartimentalizar o nosso código, como diz o conceito DRY "Não devemos nos repetir". Por isso, quando definimos, por exemplo, uma classe "ServiçosBancários()" e nela temos métodos que, posteriormente, definem "Emprestimo()" e "SeguroVeículo()", devemos separá-los em classes diferentes que herdam atributos da classe-mãe (ServiçosBancário).

Segue o exemplo:

```java
public class Servico {
    private Cliente cliente;
    private Funcionario funcionario;

    //Empréstimo
    private double valor;
    private double taxa;

    //Seguro Veiculo
    private Veiculo veiculo;
    private double valorSeguro;
    private double franquia
```

```java
/* SE TORNA */

class Emprestimo{
    private Cliente cliente;
    private Funcionario funcionario;
private double valor;
private double taxa;
}

class SeguroVeiculo{
    private Cliente cliente;
    private Funcionario funcionario;
    private Veiculo veiculo;
    private double valorSeguro;
    private double franquia
}
```



```
/* APLICANDO HERANÇA E DRY */


class Emprestimo extends Servico{
    private double valor;
    private double taxa;
}

class SeguroVeiculo extends Servico{
    private Veiculo veiculo;
    private double valorSeguro;
    private double franquia
}
// Agora podemos utilizar os atributos de Serviço
```

15. De um exemplo de herança utilizando uma classe e subclasses definidas por você e tente implementar todos os conceitos acima.

Arquivo em anexo.

REFERENCIAS:

https://www.devmedia.com.br/sobrecarga-e-sobreposicao-de-metodos-em-orientacao-a-objetos/33066

https://learning.oreilly.com/library/view/beginning-java-9/9781484229026/A323069_2_En_1_Chapter.html