Exercícios Sobre Java e o Paradigma Orientado a Objeto

---

1) Qual a diferença entre classe e objeto?
   A classe representa alguma entidade do mundo real, enquanto o objeto representa a instanciação dessa entidade (ex: Classe = carro, Objeto = Modelo:Fusca,Ano:1990)

2) Qual é a finalidade do método construtor?
   O método construtor serve para instanciar um objeto da classe

3) Identifique na instrução abaixo: a classe, o objeto, o construtor e a operação de instanciação.

```
Computador computador = new Computador();
```

Computador = Classe
computador = Objeto
New = Operação de instanciação
Computador() = Construtor

4) O que diferencia um construtor de um método qualquer?
   O construtor leva o mesmo nome da classe, geralmente com a letra inicial maiscula

5) Observe que a classe abaixo não possui um construtor, porém para que seja criado um objeto sempre é necessário utilizar o operador new seguido do nome do construtor. Desta forma, não será possível criar um objeto desta classe? Explique.

```
public class Garrafa{
    private String tipo;

    public void setTipo (String tipo){
          this.tipo = tipo;
    }

    public String getTipo(){
    return tipo;
    }
}
```

Todas as classes tem um método construtor como padrão, então mesmo que nós não o criemos esse código ainda poderá funcionar

6) A classe abaixo é parecida com a classe do exercício anterior, porém agora ela possui um construtor alternativo. O que irá acontecer se em outra classe você desejar criar um objeto da mesma com a seguinte instrução:

Garrafa gar = new Garrafa();

```
public class Garrafa{
      private String tipo;

      public Garrafa (String tipo){
      this.tipo = tipo;
     }

      public void setTipo (String tipo){
      this.tipo = tipo;
     }

     public String getTipo(){
           return tipo;
     }
}
```

Neste caso acusará erro já que você criou um outro construtor o que faz com que o padrão seja desconsiderado, para resolver isso é só adicionar agora no código o construtor padrão

7) As duas classes abaixo não estão no mesmo pacote. Faça uma análise das mesmas e corrija possíveis erros de compilação.

```
public class Carro{
      protected int litrosNoTanque;
      protected boolean carroLigado;

       public void encherTanque(int litros){
            litrosNoTanque = litros;
      }
}
```

```
public class TesteCarro{      public static void main (String args[]){
            Carro carro = new Carro();

            carro.encherTanque(10);
            carro.carroLigado = true;
      }
}
```

8) Observe as classes Circulo e TesteCirculo abaixo e realize as seguintes tarefas:

- Declare o atributo raio da classe Circulo como privado;
- Crie os métodos necessários na classe Circulo de modo que seja possível obter ou alterar o valor do atributo raio pela classe TesteCirculo;
- Caso o objetivo seja alterar o valor do atributo raio, faça a seguinte verificação: Se o valor do novo raio for positivo, faça a atribuição, caso contrário não faça;
- Siga as orientações presentes na classe TesteCirculo e execute cada tarefa;

```
public class Circulo{
      double raio;

    public double getRaio() {
        return raio;
    }
    public void setRaio(double raio) {
        if(raio >= 0){
            this.raio = raio;
        }
        else{
            System.out.println("Valor inválido");
        }
    }
}
```

```
public class testeCirculo{
            public static void main (String
args[]){
            Circulo c = new Circulo();
        c.setRaio(10);
        System.out.println(c.getRaio());

      }
}
```

9) Observe as classes AcessaBanco e TesteBanco abaixo e realize as seguintes tarefas:
   - Declare os atributos login e conectado da classe AcessaBanco como privado;
   - Crie os métodos necessários na classe AcessaBanco de modo que seja possível obter ou alterar o valor

dos atributos login e conectado pela classe TesteBanco;

- Siga as orientações presentes na classe TesteBanco e execute cada tarefa;

```
public class AcessaBanco{
      private String login;
        private boolean conectado;

        public String getLogin() {
            return login;
        }
        public void setLogin(String login) {
            this.login = login;
        }

        public void setConectado(boolean conectado) {
            this.conectado = conectado;
        }
        public boolean getConectado() {
            return conectado;
        }
}
```

```
public class TesteBanco{
      public static void main (String args[]){


      /* Crie um objeto da classe AcessaBanco */
        AcessaBanco ab = new AcessaBanco();
      /* Coloque aqui o codigo para alterar o
login para o seu nome */
       ab.setLogin("Andre");
      /* Coloque aqui o codigo para ajustar
conectado para true */
       ab.setConectado(true);
      }
}
```

10) Analise o código abaixo e descreva em poucas linhas onde está o erro de compilação e qual seria a solução para corrigir o mesmo.

```
public class TesteStatic{
      private int a1;
      private int a2;

      public static void main (String args[]){
            iniciar(10,40);
      }

      public void iniciar(int v1, int
v2){              int b1 = v1;
      int b2 = v2;
      }
}
```

Um método não estático não consegue ser visto por um método estático, a solução seria adicionar o static no método iniciar

11) Analise as classes abaixo e descreva em poucas linhas se existe erro de compilação ou não. E caso exista algum, descreva qual deveria ser a solução.

```
public final class Veiculo{
      private String chassi;

      public String getChassi(){
            return chassi;
      }
}
```

```
public class Carro extends Veiculo{
      private String cad;

      public String getCad(){             return cad;
      }
}
```

Uma classe final não consegue ser herdada, então a solução seria coloca-la como public

12) A classe Jogo abaixo foi descrita sem atributos e métodos.

```
public class Jogo{ }
```

Logo após o processo de compilação foi utilizado o utilitário javap (que transforma código binário em código Java) e o mesmo retornou o código abaixo.

```
public class Jogo extends java.lang.Object{
public Jogo(); }
```

Note que existem mais informações no código gerado pelo utilitário javap do que o código original. O que aconteceu?

Todo e qualquer método sempre é filho do java.lang.Object e o public Jogo() é o construtor padrão que normalmente fica oculto

13) Analisando a classe Java abaixo podemos observar que a mesma possui apenas um atributo, um construtor e dois métodos.

Perceba que dentro do método main estão sendo invocados métodos e atributos que não pertencem à classe. Isto é um erro de compilação? Justifique sua resposta.

```
public class PessoaFisica extends Pessoa{
      private String RG;

      public PessoaFisica(){
            super();
      }
      public String getRG(){              return RG;
      }

      public static void main (String args[]){
PessoaFisica pf = new PessoaFisica();

            pf.setEndereco("Rua XV n. 10");
pf.setFone("2546-3274");

            System.out.println(pf.endereco);
            System.out.println(pf.fone);

      }
}
```

Não, a classe pessoa física extende a classe Pessoa, então como no construtor a ela utiziliza o super() ela basicamente está reutilizando o construtor lá da classe Pessoa como herança

14) Faça um programa em Java para implementar uma calculadora simples com as quatro operações básicas da matemática. Crie três métodos para cada uma das operações e cada método deverá ser sobrecarregado, pois um deles deve receber apenas dois parâmetros do tipo int, o outro apenas dois parâmetros do tipo float e o último apenas dois parâmetros do tipo String. Quando os parâmetros

forem do tipo String, os mesmos deverão ser convertidos para o tipo int.

```
public class Calculadora {
    public int soma(int a, int b) {
        return a + b;
    }
    public float soma(float a, float b) {
        return a + b;
    }
    public int soma(String a, String b) {
        return Integer.parseInt(a) + Integer.parseInt(b);
    }

public static void main(String[] args) {
    Calculadora calc = new Calculadora();
    System.out.println(calc.soma(5, 10));
    System.out.println(calc.soma(5.5f, 10.2f));
    System.out.println(calc.soma("5", "10"));
}
}
```

15) Crie uma classe Java chamada ClienteBanco com os seguintes atributos (privados):
    - nome;
    - CPF;
    - endereço;

Além do construtor padrão, crie um construtor alternativo para iniciar cada um dos atributos. Crie os métodos necessários para acessar estes atributos. Faça a sobrescrita do método toString() da classe Object para o mesmo retornar a seguinte mensagem:
"O Sr." <nome> "portador do CPF n." <CPF> " residente e domiciliado a " <endereço> "vem por meio desta solicitar o encerramento de sua conta corrente".

Crie um método main e dentro do mesmo faça com que a mensagem gerada pelo método toString() seja impressa na tela.

```
public class ClienteBanco {
  private String nome;
```

```
    private String cpf;
    private String endereco;


     public ClienteBanco(){

    }

    public ClienteBanco(String nome, String cpf, String endereco) {
        this.nome = nome;
        this.cpf = cpf;
        this.endereco = endereco;
    }


    public String getCpf() {
        return cpf;
    }
    public String getEndereco() {
        return endereco;
    }
    public String getNome() {
        return nome;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

    @Override
    public String toString(){
        return "O Sr(a). " + nome + " portador do CPF n. " + cpf + " residente e
domiciliado a " + endereco + " vem por meio desta solicitar o encerramento da sua
conta corrente";
    }

}


public class TesteBanco{
     public static void main (String args[]){

          ClienteBanco c1 = new ClienteBanco("Andre",
"427.946.588.67","Rua Maria Clara");
```

```
        System.out.println(c1);

}
}
```