Programação I

GABARITO – EXERCÍCIOS DE FIXAÇÃO
(classes, objetos, métodos, main)

Exercício 1. Crie uma classe chamada Cor, que possui 3 inteiros como atributos, sendo eles r, g e b.

```
public class Cor{

    private int r, g, b;

}
```

Exercício 2. Crie uma classe chamada Computador, que possui 4 atributos, sendo eles a marca, a velocidade, o ano de fabricação e um atributo que indica se o computador é novo.

```
public class Computador{

    private String marca;

    private int anoFabricacao;

    private double velocidade;

    private boolean novo;

}
```

Exercício 3. Crie uma classe chamada Lampada, que possui um atributo indicando se a lâmpada está ligada.

```
public class Lampada{

    private boolean ligada;

}
```

Exercício 4. Crie uma classe chamada Produto, que possui um nome, um valor, uma descrição (tipo String) e uma quantidade em estoque.

```
public class Produto{

    private String nome, descricao;

    private double valor;

    private int quantidadeEstoque;

}
```

Exercício 5. Crie um construtor para cada uma das classes criadas.

Classe Cor:

```
    public Cor(int r, int g, int b){

        this.r = r;

        this.g = g;

        this.b = b;

    }
```

Classe Computador:

```
    public Computador(String marca, int anoFabricacao, double
velocidade, boolean novo){

        this.marca = marca;

        this.anoFabricacao = anoFabricacao;

        this.velocidade = velocidade;

        this.novo = novo;

    }
```

Classe Lampada:

```
    public Lampada(boolean ligada){
```

```
        this.ligada = ligada;

    }
```

Classe Produto:

```
    public Produto(String nome, String descricao, double valor,
int quantidadeEstoque){

        this.nome = nome;

        this.descricao = descricao;

        this.valor = valor;

        this.quantidadeEstoque = quantidadeEstoque;

    }
```

Exercício 6. Pesquise (se necessário) e responda: o que é sobrecarga de métodos?

Sobrecarga é quando temos dois ou mais métodos com o mesmo nome, mas que diferenciam nos parâmetros ou tipo de retorno. Por exemplo, é possível criar mais de um construtor para uma classe, cada um recebendo parâmetros diferentes.

Exemplo:

```
public class Teste{

     private int x;

     public Teste(int x){

          this.x = x;

     }

     public Teste(){

     }

}
```

Exercício 7. Sobrecarregue o método construtor das classes criadas nos exercícios 1, 2, 3 e 4, criando mais um método deste tipo em cada uma delas.

Classe Cor:

```
    public Cor(int r, int g, int b){

        this.r = r;

        this.g = g;

        this.b = b;

    }


    public Cor(){


    }
```

Classe Computador:

```
   public Computador(String marca, int anoFabricacao, double
velocidade, boolean novo){

        this.marca = marca;

        this.anoFabricacao = anoFabricacao;

        this.velocidade = velocidade;

        this.novo = novo;

    }


    public Computador(String marca, boolean novo){

        this.marca = marca;

        this.novo = novo;

    }
```

Classe Lampada:

```
    public Lampada(boolean ligada){
        this.ligada = ligada;
    }

    public Lampada(){

    }
```

Classe Produto:

```
    public Produto(String nome, String descricao, double valor,
int quantidadeEstoque){
        this.nome = nome;
        this.descricao = descricao;
        this.valor = valor;
        this.quantidadeEstoque = quantidadeEstoque;
    }

    public Produto(String nome, String descricao){
        this.nome = nome;
        this.descricao = descricao;
    }
```

Exercício 8. O que são atributos? E o que são métodos?

Na orientação a objetos, atributos são as características de uma determinada classe, enquanto os métodos são as ações que os objetos desta classe poderão realizar.

Exercício 9. O que é e para que serve um método construtor? Quais as duas características necessárias para que um método seja o construtor da classe? Exemplifique sua resposta.

O método construtor serve para inicializar os atributos da classe. Para ser o método construtor da classe precisa ter o mesmo nome da classe e não possuir tipo de retorno.

Exemplo:

```
public class Teste{

     private int x;

     public Teste(int x){

          this.x = x;

     }

}
```

Exercício 10. O que são e para que servem os métodos de acesso de uma classe? Exemplifique sua resposta.

Servem para pegar ou modificar informações de um determinado atributo da classe. O método de acesso GET serve para pegar a informação armazenada no atributo, enquanto o SET serve para modificar a informação do atributo.

Por exemplo, os métodos de acesso de um atributo nome do tipo String em uma classe qualquer seria desta forma:

```
public String getNome(){

     return nome;

}

public void setNome(String nome){

     this.nome = nome;

}
```

Exercício 11. Seja x uma variável inteira. Pesquise e responda: qual a diferença entre a utilização de x++ e ++x no código? Exemplifique sua resposta.

A utilização de ++x faz a adição de 1 na variável x (incremento) e na mesma linha esse valor já pode ser usado com essa modificação (ou seja, primeiro soma e depois usa). Já x++ também faz a adição de 1 na variável x, porém essa modificação ocorrerá após a utilização da variável (ou seja, usa e depois soma).

Por exemplo:

```
int x = 10;

System.out.println(++x); //imprime 11

System.out.println(x++); //imprime 11

System.out.println(x); //imprime 12
```

Exercício 12. Suponha a existência de uma classe chamada ContaCorrente, com os atributos número da conta (do tipo int) saldo atual (do tipo double). Sabendo que **NÃO FOI CRIADO UM CONSTRUTOR NA CLASSE ContaCorrente**, apenas os métodos de acesso, escreva o método main para criar 3 contas correntes com as seguintes informações (número, saldo):

Conta 1: 1234, 100.00
Conta 2: 8765, -98.00
Conta 3: 3342, 3445.80

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

    //Conta 1

    ContaCorrente c1 = new ContaCorrente();

    c1.setNumeroConta(1234);

    c1.setSaldoAtual(100.00);

    //Conta 2

    ContaCorrente c2 = new ContaCorrente();

    c2.setNumeroConta(8765);

    c2.setSaldoAtual(-98.00);
```

```
        //Conta 3

        ContaCorrente c3 = new ContaCorrente();

        c3.setNumeroConta(3342);

        c3.setSaldoAtual(3445.80);

}
```

Exercício 13. Pesquisa (se necessário) e responda: o que é o método toString()? Como seria um exemplo de método toString() dentro da classe ContaCorrente descrita no exercício 12?

O método toString retorna uma representação no formato de texto de determinado objeto de uma classe. Por exemplo, na classe ContaCorrente (do exercício 12), poderemos fazer o método toString retornar de várias formas, uma delas pode ser vista abaixo:

```
        public String toString(){

                return "Número da conta: " + getNumeroConta() + "Saldo
                atual: " + getSaldoAtual();

        }
```

Desta forma, quando fizermos:

```
        ContaCorrente c = new ContaCorrente();

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

A impressão na tela será o retorno do método toString (não importando o conteúdo dos atributos). Caso não houvesse a implementação explícita do toString, a impressão na tela, neste exemplo, seria algo como ContaCorrente@CODIGO_HEXADECIMAL, mostrando onde a referência está alocada na memória.

Exercício 14. Considerando a classe abaixo, crie um método main que chama todos os métodos desta classe. A última linha de código deve ser a chamada ao método imprimeInformacoes(), na qual todas as informações devem aparecer corretamente.

```
public class Empregado{
   private String nome;
   private char turno;
```

```
    private double salario;
    public void aumentaSalario(double aumento){
       this.salario += aumento;
    }
    public double calculaAdicionalNoturno(){
       if(turno == 'n')
          turno = 'N';
       if(turno == 'N')
          return salario + salario*0.3;
       return 0;
    }
    public void imprimeInformacoes(){
       System.out.println("Nome: "+nome);
       System.out.println("Turno: "+turno);
       System.out.println("Salário: "+salário);
       System.out.println("Adicional noturno: "+calculaAdicionalNoturno());
    }
    public Empregado(String nome){
       this.nome = nome;
    }
    public void setTurno(char turno){
       this.turno = turno;
    }
    public void setSalario(double salario){
       this.salario = salario;
    }
}
```

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

     Empregado empregado1 = new Empregado("Teste");

     empregado1.setSalario(50);

     empregado1.aumentaSalario(100);

     empregado1.setTurno('N');

     empregado1.calculaAdicionalNoturno();

     empregado1.imprimeInformacoes();

}
```