Aonde você quer chegar?

**Vai com a**

UniCesumar

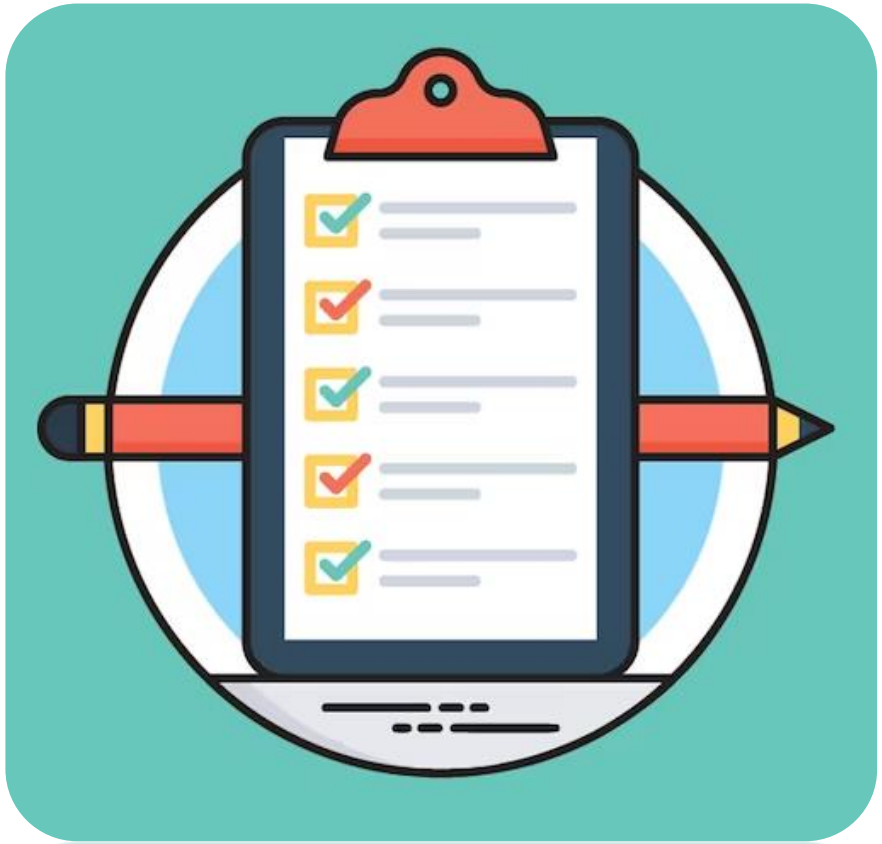
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA

# Programação Orientada a Objetos

## Aula 8

*Prof. Rômulo Santos*

UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA



# Sumário

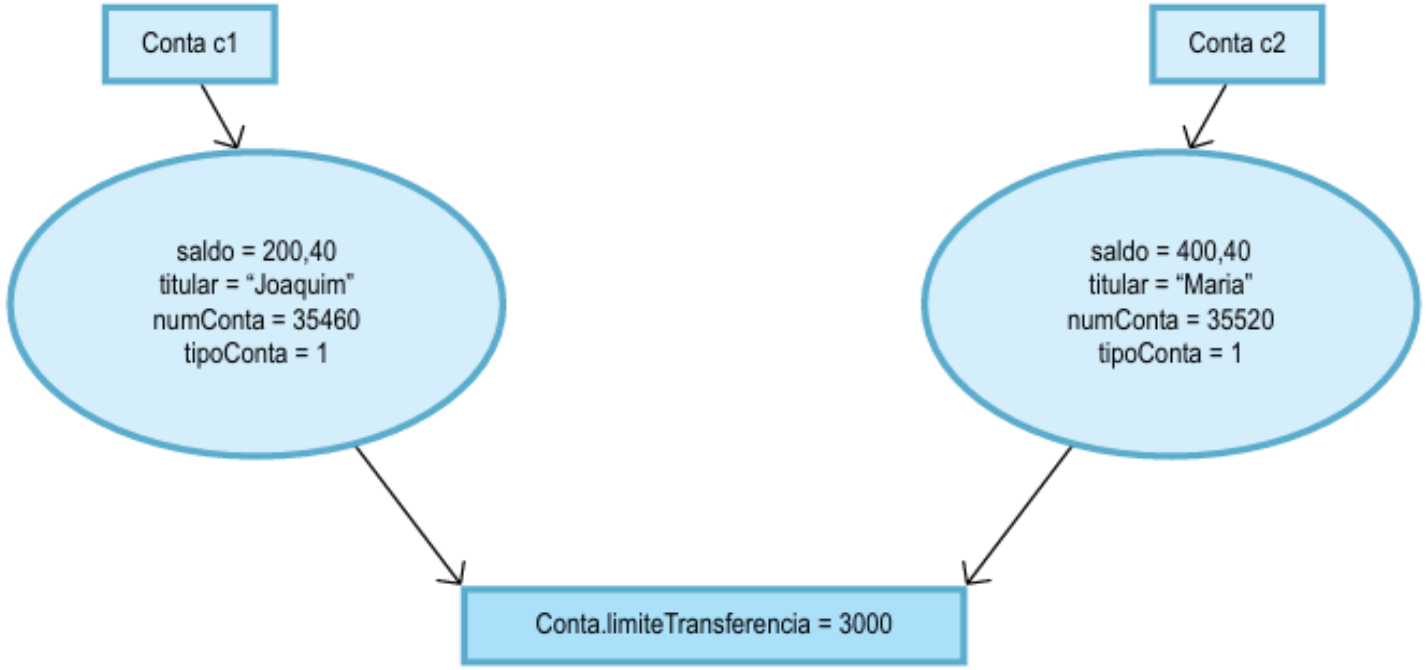
- Modificadores *static*, *final* e *abstract*.
- Construtores.
- Herança.

UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA

## **Modificador *static***:

O modificador *static* indica que para acessar o método da classe **não é necessário instanciá-lo**, e o que acontece é que métodos e variáveis *static* são alocadas em memória antes que qualquer objeto seja criado.

Exemplificando, imagine que haja um **limite** para realizar **transferências** entre contas e que este limite seja "**IGUAL**" para todas as contas.



## **Modificador *final***:

Restringem ainda mais o acesso aos elementos de uma classe, para atributos, ele faz com que o **atributo** não possa ser modificados em tempo de execução, ou seja, cria-se uma variável que terá um valor constante do início ao término da execução da aplicação.

Para **classes**, indica que esta não poderá ser herdada (não poderá ter filhos) e para **métodos**, indica que o mesmo não poderá ser sobrescrito (usar técnicas de polimorfismo).

```
public class Conta{
 …
 private final static float LIMITETRANSFERENCIA=3000;
 …
}
```

## **Modificador *abstract***:

O objetivo de criação de classes abstratas é fornecer uma **superclasse** apropriada a partir da qual outras classes podem herdar e assim poder compartilhar um *design* comum.

```
public abstract class Pessoa {
    protected String telefone;
    protected String nomePessoa;
    protected Endereco e = new Endereco();
    //Método abstrato para cadastro de pessoa
    public abstract void cadastra();
    public Endereco getE() {...}
    public void setE(Endereco e) {...}
    public String getNomePessoa() {...}
    public void setNomePessoa(String nomePessoa) {...}
    public String getTelefone() {...}
    public void setTelefone(String telefone) {...}
}
```

## Modificador *abstract*:

```
import cliente.Pessoa;

cliente.Pessoa is abstract; cannot be instantiated

public class CaixaEletronico {
    public static void main(String[] args) {
        Pessoa p = new Pessoa();


    }
}
```

Figura 71 – Tentativa de criação de um objeto da classe abstrata Pessoa

## Modificador *abstract*:

```
public class Fisica extends Pessoa{

    private String cpf;

    //implementacao do metodo abstrato é imprescindivel
    @Override
    public void cadastra() {
        //leitura via teclado
        Scanner tec = new Scanner(System.in);
        System.out.println("Digite o nome");
        nomePessoa = tec.nextLine();
        System.out.println("Digite o telefone");
        telefone = tec.nextLine();
        System.out.println("Digite o cpf");
        cpf = tec.nextLine();
        e.cadastra();
    }

}
```

Figura 72 – Classe Pessoa, subclasse da classe abstrata Pessoa

**Questão 01:**

**Considere as afirmativas abaixo, sobre os modificadores de acesso da linguagem de programação JAVA.**

I. O modificador de acesso *public* é o menos restritivo de todos, ou seja, uma classe Java com esse modificador fica visível para qualquer outra classe dentro do programa, independentemente de estar dentro do mesmo pacote ou não.

II. O modificador de acesso *default* é o menos restritivo de todos, ou seja, uma classe Java com esse modificador fica visível para qualquer outra classe dentro do programa, independentemente de estar dentro do mesmo pacote ou não.

III. O modificador de acesso *private* é o mais restritivo de todos.

IV. O modificador de acesso *protected* é o mais restritivo de todos.

Está correto somente o que se afirma em:

A) I
B) II
C) I e III
D) I e IV
E) II e IV

**Questão 02:**

**Com referência à linguagem de programação Java, julgue o item a seguir.**

Uma classe final não pode ser herdada, um método final não pode ser sobrescrito, e o valor de um atributo final não pode ser alterado.

Certo

Errado

**Questão 03:**

**Com relação à programação Java, julgue os próximos itens.**

Ao se declarar uma nova classe, é possível especificar um dos seguintes modificadores: *public*, *final* ou *abstract*. Uma classe *abstract* pode ser instanciada e derivada.

Certo

Errado

## **Construtores**:

Um construtor é o primeiro “método” que é executado sempre que uma classe é **instanciada**. Quando se utiliza a palavra chave ***new***, o construtor será executado e inicializará o objeto.

```
public abstract class Pessoa {
 protected String telefone;
 protected String nomePessoa;
 protected Endereco e = new Endereco();

 public Pessoa(){
 super();
 System.out.println(“Executando o construtor de Pessoa”);
 }
 ...//outros métodos da classe
}
```

## **Construtores**:

Objetos do tipo Física são filhos da classe Pessoa, quando criamos uma instância de Física os construtores são executados em forma de pilha: **Física** chama **Pessoa**, que chama ***Object***.

```
public class Fisica extends Pessoa{
 private String cpf;
 public Fisica(){
 System.out.println("Pessoa Fisica");
 }
}
```

```
public class CaixaEletronico {
 public static void main(String[] args) {
 Fisica f = new Fisica();

 }
}
```

## **Construtores**:

Uma classe pode possuir **mais** de um construtor.

```
public class Fisica extends Pessoa{
 private String cpf;
 public Fisica(){
 System.out.println("Pessoa Fisica");
 }
 public Fisica(String nome){
 nomePessoa = nome;
 }
}
```

```
public class CaixaEletronico {
 public static void main(String[] args) {
 Fisica f = new Fisica();
 Fisica f2 = new Fisica("Joaquim");
 System.out.println(f2.getNomePessoa());

 }
}
```

## **Construtores**:

Um ponto importante é que enquanto o construtor **não** for executado, nenhum acesso à variável de instância ou método será possível.

```
public class Fisica extends Pessoa{
 private String cpf;
 public Fisica(){
 System.out.println("Pessoa Fisica");
 }
 public Fisica(String nome){
 nomePessoa = nome;
 super();
 }
}
```

**Questão 04:**

**Sobre construtores em Java, analise as assertivas abaixo.**
**1) Um construtor é usado para criar objetos.**
**2) Um construtor deve ter o mesmo nome da classe em que é declarado.**
**3) Um construtor pode ser declarado como privado.**

Estão corretas:

A) 1, 2 e 3.
B) 1 e 2, apenas.
C) 1 e 3, apenas.
D) 2 e 3, apenas.
E) 3, apenas.

**Questão 05:**

**O método construtor é um tipo especial de rotina que toda classe possui. É uma característica de todo método construtor na linguagem Java:**

A) obrigatoriedade de sua declaração.

B) desnecessária alocação de memória para sua execução.

C) atribuição de nome diferente da classe a que pertence.

D) ausência de especificação de tipo de retorno.

**Questão 06:**

**Em uma subclasse de uma relação de herança em Java, o construtor recebeu como parâmetro valores para seus atributos e para os atributos da superclasse. No corpo deste construtor, para encaminhar os atributos direcionados à superclasse para o construtor da superclasse utiliza-se a instrução**

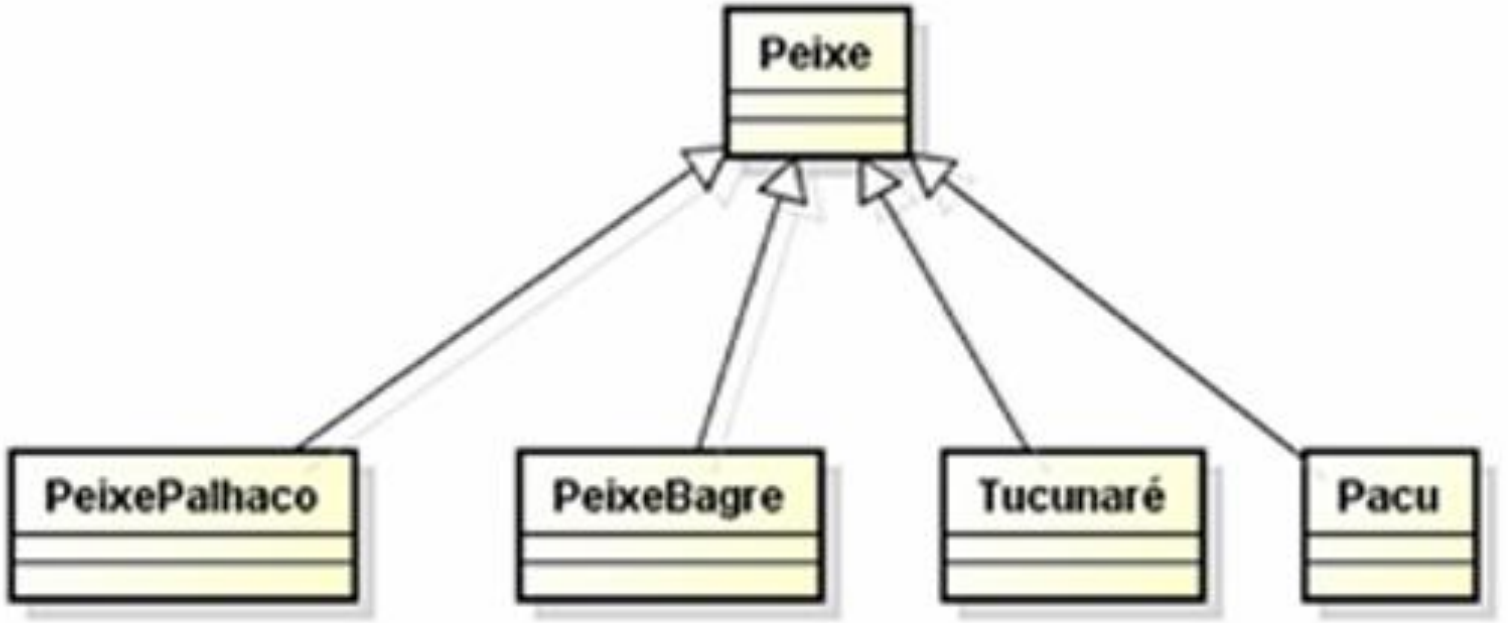
A) *super*
B) *this*
C) *send*
D) *that*
E) *root*

## **Herança**:

Permite a criação de novas classes a partir de outras classes, reutilizando, estendendo ou modificando o seu comportamento:

- A classe que herda os membros é chamada classe **derivada**.
- A classe cujos membros são herdados é chamada classe **base**.
- Permite que as classes **filhas** reutilizem o **código** da classe pai e adicionem novos recursos.
- Permite que uma classe herde todos os **comportamentos** e atributos de outra classe.
- Permite que uma classe tenha imediatamente todas as **funcionalidades** de uma classe já existente.
- Permite criar uma hierarquia entre os objetos, onde os descendentes herdam dos seus ancestrais.

**Herança**:




Superclasse Peixe e respectivas subclasses

## **Herança**:

A linguagem Java permite o uso de herança **simples**, mas não permite a implementação de herança múltipla, que significa que uma classe pode herdar métodos e atributos de mais de uma classe.

Para superar essa limitação, o Java faz uso de **interfaces**, o qual será descrito mais adiante.

Em Java, a palavra reservada que define que uma classe herda as características de outra é **"*extends*"**, ela deve ser utilizada assim que a classe for criada.

# Herança:

```
public class Peixe {
 //atributos
 private String tipoPele;
 private int numDentes;
 //metodos
 public void nadar(){
 System.out.println("Mecher as barbatanas");
 }
 public void comer(){
 System.out.println("Procurar comida e comer");
 }
}
```

```
public class PeixePalhaco extends Peixe{
 //implementação da classe peixe palhaco
 private int numeroListras;
}
```

```
public class Principal
 public static void main(String args[]){
 PeixePalhaco pp = new PeixePalhaco();
 pp.nadar();
 PP.comer();
 }
}
```

**Questão 07:**

**Herança é uma forma de reutilização de *software* na qual uma nova classe é criada, absorvendo membros de uma classe existente e aprimorada com capacidades novas ou modificadas. Sobre herança, analise as assertivas e, em seguida, assinale a alternativa que apresenta as corretas.**

**I. No caso de herança simples, uma classe é derivada de uma superclasse indireta.**
**II. A superclasse direta é a superclasse a partir da qual a subclasse herda explicitamente.**
**III. A superclasse indireta é qualquer superclasse acima da classe direta na hierarquia de classe.**
**IV. Os relacionamentos de herança formam estruturas hierárquicas do tipo árvore.**

A) Apenas I, II e III.
B) Apenas I, II e IV.
C) Apenas II, III e IV.
D) Apenas I e II.
E) Apenas III e IV.

**Questão 08:**

**Na programação orientada a objeto, a habilidade de estender de mais de uma classe é conhecida como herança múltipla. Na linguagem de programação Java, há uma restrição para se utilizar a herança múltipla. Sendo assim, em Java, a herança múltipla só é permitida para**

A) classes.
B) atributos
C) métodos.
D) interfaces
E) permissões.

UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA

**Questão 09:**

**Na programação orientada a objetos, a herança é uma técnica de abstração que permite categorizar as classes de objetos sob certos critérios, especificando-se as características dessas classes. As classes que são vinculadas por meio de relacionamentos de herança formam uma hierarquia de herança. Na linguagem de programação Java, o relacionamento de herança é definido pela palavra-chave**

A) *static*
B) *extends*
C) *public*
D) *new*
E) *this*


UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA