Notas de aula do capítulo 3

Curso: Programação Orientada a Objetos com C#

https://www.udemy.com/programacao-orientada-a-objetos-csharp

Prof. Nelio Alves

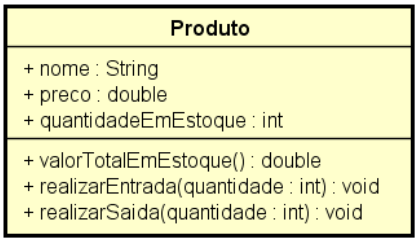
Construtores
Palavra “this”

Exemplo

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

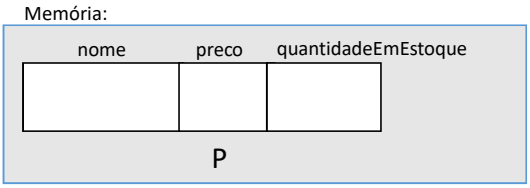
Para resolver este problema, você deve criar uma CLASSE conforme projeto ao lado:



1ª proposta de melhoria:

Quando executamos o comando abaixo, instanciamos um produto P com seus atributos "vazios":

P = new Produto();



Entretanto, faz sentido um produto que não tem nome? Faz sentido um produto que não tem preço?

Com o intuito de evitar a existência de produtos sem nome, sem preço e sem quantidade, é possível fazer com que seja "obrigatória" a iniciação desses valores?

Solução:

Para resolver isso, vamos criar um CONSTRUTOR para a classe Produto.

CONTRUTOR: é uma operação especial da classe, que executa no momento da instanciação de um objeto.

Palavra “this”:

É uma referência ao próprio objeto.

É como se o objeto falasse “eu mesmo”.

Observações:

Quando não se define nenhum construtor na classe, o C# coloca um CONSTRUTOR PADRÃO na classe, como estávamos usando antes:

P = new Produto();

2ª proposta de melhoria:

Não seria mais adequado que a quantidade em estoque de um novo produto inicie, por padrão, com o valor 0 (zero)?

Sobrecarga

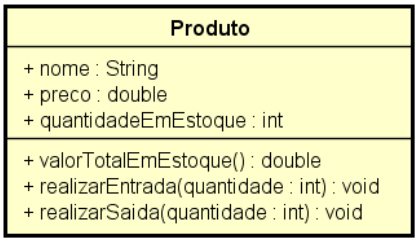
Prof. Nelio Alves – nelioalves.com

Exemplo

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

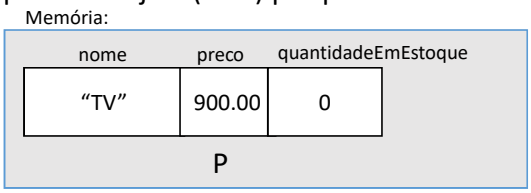
Para resolver este problema, você deve criar uma CLASSE conforme projeto ao lado:



2ª proposta de melhoria:

E se eu quiser informar no construtor somente o nome e preço do produto, de modo que a quantidade inicial de um produto seja 0 (zero) por padrão?

P = new Produto(nome, preco);



SOLUÇÃO:

SOBRECARGA: é o recurso que permite definir mais de uma versão da mesma operação, diferenciando-as pela lista de parâmetros.

Encapsulamento

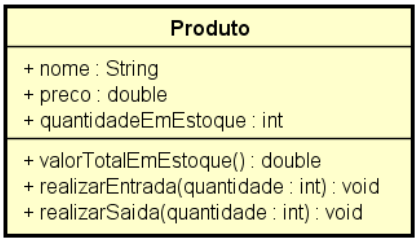
Prof. Nelio Alves – nelioalves.com

Exemplo

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

Para resolver este problema, você deve criar uma CLASSE conforme projeto ao lado:



3ª proposta de melhoria:

Do jeito que construímos nossa classe Produto, nada impede que sejam feitas alterações na quantidade em estoque acessando diretamente o atributo:

```
P.quantidadeEmEstoque = 20;
P.quantidadeEmEstoque = P.quantidadeEmEstoque * 3;
```

Esse tipo de operação é insegura, pois aumenta a possibilidade do programador cometer erros ao alterar a quantidade do estoque.

O correto, neste caso, é que a quantidade do estoque somente possa ser alterada por meio de entradas e saídas:

```
P.realizarEntrada(5);
```

Discussão

O adequado é que toda lógica relacionada a produto esteja implementada na classe **Produto**.

O programador que vai usar a classe **Produto** não precisa conhecer **COMO** essa lógica foi implementada. Ele deve conhecer apenas **O QUÊ** a classe faz.

ENCAPSULAMENTO: consiste em esconder detalhes de implementação de uma classe.

Padrão usado:

- Os detalhes de implementação devem ser escondidos do programador.
- A classe deve expor **somente métodos** para o programador usar.

Padrão de implementação 1 (Java)

```
class Produto {

        private double preco;

        public double getPreco() {
                return this.preco;
        }
}
```

```
class Produto {

        private double preco;

        public double getPreco() {
                return this.preco;
        }

        public void setPreco(double preco) {
                this.preco = preco;
        }
}
```

Padrão de implementação 2 (C#)

```
class Produto {

        private double preco;

        public double Preco {
                get {
                        return this.preco;
                }
        }
}
```

```
class Produto {

        private double preco;

        public double Preco {
                get {
                        return this.preco;
                }

                set {
                        this.preco = value;
                }
        }
}
```

Padrão de implementação 3 (C#)

```
class Produto {

       public double preco { get; private set; }
}
```

```
class Produto {

       public double preco { get; set; }
}
```

Níveis de acesso em C# para atributos e métodos:

public	Sem restrição
private (padrão)	Somente a própria classe pode acessar
protected	Somente a própria classse e suas subclasses podem acessar
internal	Somente as classes do mesmo projeto podem acessar
protected internal	Somente as classes do mesmo projeto e subclasses podem acessar

Tipos referência e Tipos valor

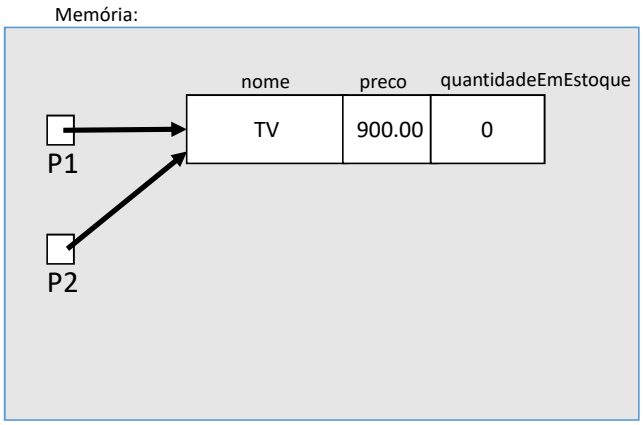
Prof. Nelio Alves – nelioalves.com

Classes são tipos REFERÊNCIA

Variáveis cujo tipo são classes não devem ser entendidas como caixas, mas sim "tentáculos" (ponteiros) para caixas

```
Produto P1, P2;

P1 = new Produto("TV", 900.00, 0);

P2 = P1;
```

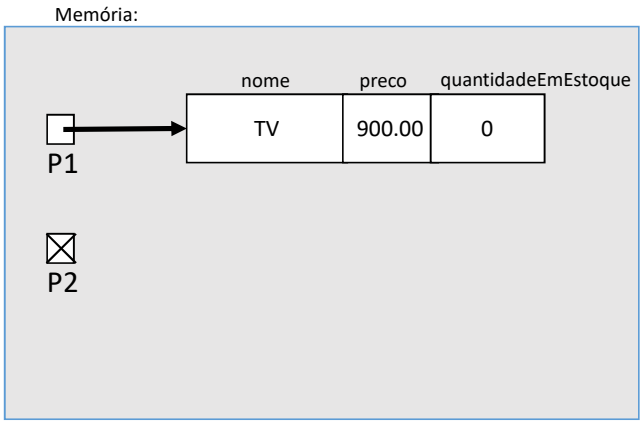
P2 = P1;
"P2 passa a apontar para onde P1 aponta"



Valor "null"

Tipos referência aceitam o valor "null", que indica que a variável aponta pra ninguém.

```
Produto P1, P2;

P1 = new Produto("TV", 900.00, 0);

P2 = null;
```



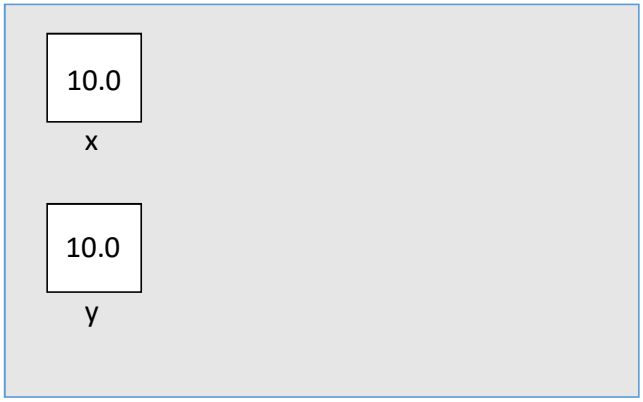
Tipos VALOR são caixas

Tipos primitivos no C# não são classes. Eles são tipos VALOR e devem ser entendidos como caixas.

```
double x, y;

x = 10.0;

y = x;
```

y = x;
"y recebe uma CÓPIA de x"

Memória:



Em C# tipos primitivos são na verdade STRUCTS

Nome abreviado	Classe do .NET	Type (Tipo)	Width	Intervalo (bits)
byte	Byte	Inteiro sem sinal	8	0 a 255
sbyte	SByte	inteiro com sinal com sinal	8	-128 a 127
int	Int32	inteiro com sinal com sinal	32	-2,147,483,648 to 2,147,483,647
uint	UInt32	Inteiro sem sinal	32	0 a 4294967295
short	Int16	inteiro com sinal com sinal	16	-32.768 a 32.767
ushort	UInt16	Inteiro sem sinal	16	0 a 65535
long	Int64	inteiro com sinal com sinal	64	-922337203685477508 to 922337203685477507
ulong	UInt64	Inteiro sem sinal	64	0 a 18446744073709551615
float	Single	Tipo de ponto flutuante de precisão simples	32	-3.402823e38 para 3.402823e38
double	Double	Tipo de ponto flutuante de precisão dupla	64	-1.79769313486232e308 para 1.79769313486232e308
char	Char	Um único caractere Unicode	16	Unicode símbolos usados no texto
bool	Boolean	Tipo booliano lógico	8	True ou false

É possível criar seus próprios structs

```
namespace curso {
    struct Ponto {

        public double x, y;

        public override string ToString() {
            return "(" + x + "," + y + ")";
        }
    }
}
```

Diferenças entre classe e struct

CLASSE	**STRUCT**
Variáveis são tentáculos (ponteiros)	Variáveis são caixas
Variáveis precisam ser instanciadas, ou apontar para um objeto já existente	Em muitos casos não precisa instanciar
Aceita valor **null**	Não aceita valor null
Aceita herança	Não aceita herança
Y = X; "Y passa a apontar para onde X aponta"	Y = X; "Y recebe uma cópia de X"
Vantagem: usufrui de todos recursos OO	Vantagem: é mais simples e mais performático

Vetores

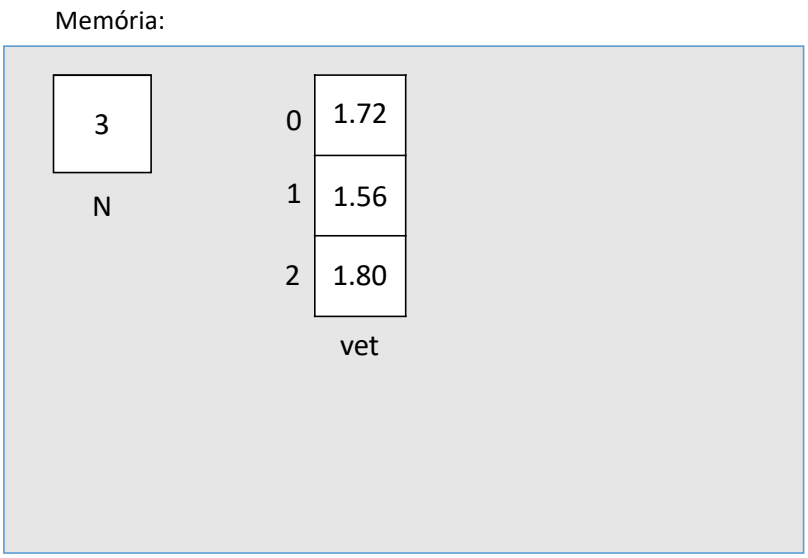
Prof. Nelio Alves – nelioalves.com

Exemplo 1:

Fazer um programa para ler um número inteiro N e a altura de N pessoas. Armazene as N alturas em um vetor. Em seguida, mostrar a altura média dessas pessoas.

Exemplo:

Entrada:	Saída:
3 1.72 1.56 1.80	ALTURA MEDIA = 1.69



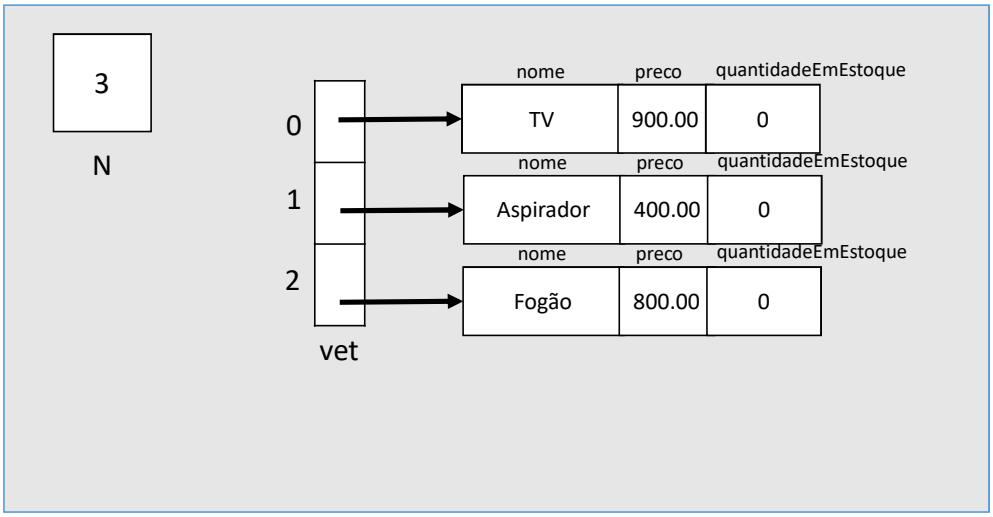
Exemplo 2:

Fazer um programa para ler um número inteiro N e os dados (nome e preço) de N Produtos. Armazene os N produtos em um vetor. Em seguida, mostrar o preço médio dos produtos.

Exemplo:

Entrada:	Saída:
3 TV 900.00 Aspirador 400.00 Fogão 800.00	PRECO MEDIO = R$ 700.00

Memória:



Matrizes

Prof. Nelio Alves – nelioalves.com

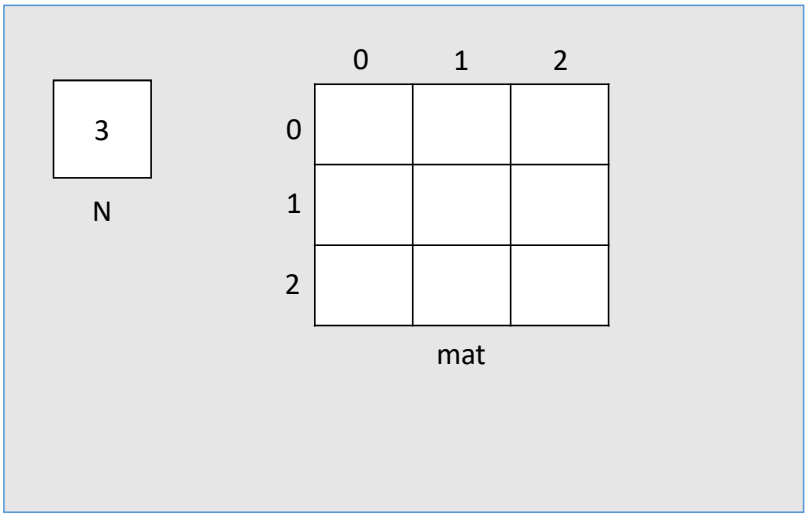
Exemplo:

Fazer um programa para ler um número inteiro N e uma matriz de ordem N contendo números inteiros. Em seguida, mostrar a diagonal principal e a quantidade de valores negativos da matriz.

Exemplo:

Entrada:	Saída:
3 5 -3 10 15 8 2 7 9 -4	DIAGONAL PRINCIPAL: 5 8 -4 QUANTIDADE DE NEGATIVOS = 2

Memória:



Listas

Prof. Nelio Alves – nelioalves.com

Lista

É uma coleção de objetos que podem ser acessados por sua posição.

Diferentemente do vetor, a lista é redimensionada dinamicamente conforme se adiciona novos elementos.

Conjuntos

Prof. Nelio Alves – nelioalves.com

Conjunto

Classe: HashSet

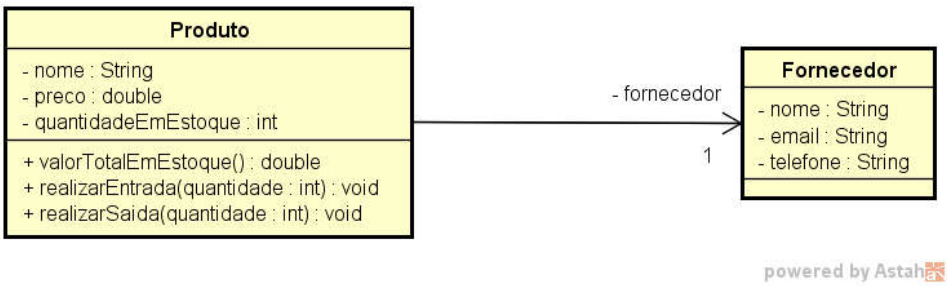
É uma coleção de objetos similar ao conceito de conjunto da Matemática.

- Não há repetição de elementos
- Os elementos não tem uma "posição" dentro do conjunto
- Oferece operações eficientes de conjunto: interseção, união, diferença, etc.

Associações

Prof. Nelio Alves – nelioalves.com

Exemplo



Herança
Classe abstrata
Sobreposição
Polimorfismo
Interface

Prof. Nelio Alves – nelioalves.com

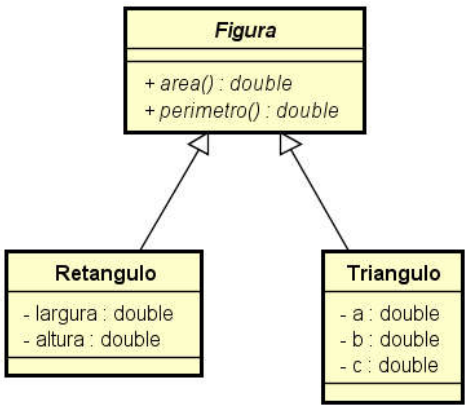
Problema:

Fazer um programa para ler as medidas de N figuras, podendo ser triângulos ou retângulos, conforme o usuário informar. Depois de ler os dados de todas figuras, mostrar as áreas das figuras na mesma ordem em que foram digitadas.

Exemplo:

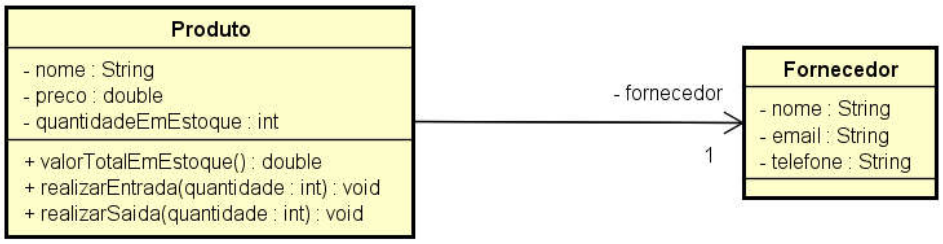
```
Quantas figuras você vai digitar? 3
Figura 1 – triângulo ou retângulo (t/r)? r
Largura: 10.0
Altura: 5.0
Figura 2 – triângulo ou retângulo (t/r)? t
Lado a: 3.0
Lado b: 4.0
Lado c: 5.0
Figura 3 – triângulo ou retângulo (t/r)? r
Largura: 4.0
Altura: 2.0

Áreas das figuras:
Figura 1: 50.0
Figura 2: 6.0
Figura 3: 8.0
```

Diagrama:



RELAÇÃO "É UM"

No vídeo anterior:



RELAÇÃO "TEM UM"

Definições

HERANÇA: é uma relação generalização-especialização entre duas classes. Também chamada de relação "é um".

A classe genérica é chamada de "superclasse"
A classe específica é chamada de "subclasse"

SOBREPOSIÇÃO: é a reimplementação, na subclasse, de um método da superclasse.

POLIMORFISMO: é a capacidade de objetos do mesmo tipo se comportarem de maneira diferente, dependendo de como eles foram instanciados durante a execução do programa.

Definições

CLASSE ABSTRATA: é uma classe que possui pelo menos um método abstrato (sem implementação).

INTERFACE: é um tipo que possui apenas métodos abstratos.

Atenção: classes abstratas e interfaces não podem ser instanciadas.

Elementos estáticos

Prof. Nelio Alves – nelioalves.com

Elementos estáticos

Elementos (atributos ou métodos) estáticos são aqueles que não dizem respeito ao próprio objeto, mas que existem e podem ser usados independentemente dos objetos.

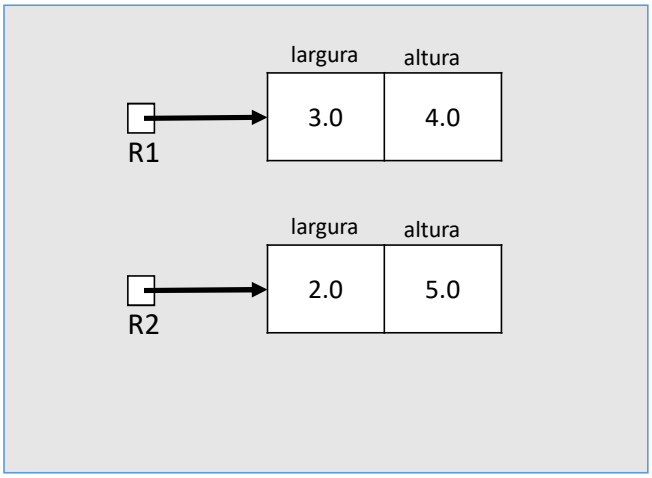
Também chamados elementos "de classe".

Elementos = atributos ou métodos

Exemplo do que NÃO é estático: o método "area"

```
Retangulo R1, R2;

R1 = new Retangulo(3.0, 4.0);

R2 = new Retangulo(2.0, 5.0);
```

Memória:



Problema:

Ler três números inteiros. Depois, informar qual é o menor dentre os três.

Exemplo:

Entrada:	Saída:
9 4 12	`MENOR = 4`

Tipos enumerados

Prof. Nelio Alves – nelioalves.com

Tipos enumerados

É um tipo especial adequado para representar dados que possuem um número finito de valores constantes.

```
enum Mes {
    Janeiro,
    Fevereiro,
    Marco,
    Abril,
    Maio,
    Junho,
    Julho,
    Agosto,
    Setembro,
    Outubro,
    Novembro,
    Dezembro
};
```

Exceções

Prof. Nelio Alves – nelioalves.com

Problema

Ler os dados de uma conta bancária (número da conta, nome do titular, limite de saque). Atenção: a conta inicia com saldo 0 (zero). Em seguida, efetuar um depósito na conta e mostrar o saldo. Em seguida, efetuar um saque e mostrar o novo saldo. As regras para saque são:

- Um saque não pode ser superior ao limite de saque da conta
- Deve haver saldo na conta