Aula 07

Orientação a objetos

Rogério Pereira Junior

rogeriopereirajunior@gmail.com

21 de maio de 2024

SENAC
Santa Catarina
Florianópolis

Discussão

- Surgiu como uma alternativa as características da programação estruturada.
- O intuito da sua criação também foi o de se aproximar do mundo real, daí o nome "objeto"como uma algo genérico, que pode representar qualquer coisa tangível.
- Esse novo paradigma se baseia principalmente em dois conceitos chave: classes e objetos.
  - **Classe:** Descrição de um conjunto de objetos que compartilham as mesmas características e comportamentos
  - **Objeto:** Entidade que contem dados e comportamentos representando a classe em questão.
- Cada classe determina o comportamento do objeto definido por métodos e seus atributos
- Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.
  - Encapsulamento
  - Herança
  - Polimorfismo

Problema Exemplo

- Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos e dizer qual dos dois triângulos possui a maior área.
- A fórmula para calcular a área de um triângulo a partir das medidas de seus lados a, b e c é a seguinte (fórmula de Heron):

$$\text{area} = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{onde} \quad p = \frac{a+b+c}{2}$$

- **Exemplo:**

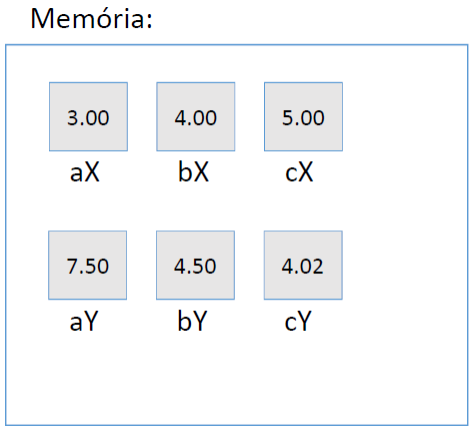
```
Entre com as medidas do triangulo X:
3.00
4.00
5.00
Entre com as medidas do trangulo Y:
7.50
4.50
4.02
Area do Triangulo X: 6.000
Area do Triangulo Y: 7.5638
Maior area: Y
```

Sem orientação a objeto

```
Scanner sc = new Scanner(System.in);
double xA, xB, xC, yA, yB, yC;

System.out.println("Entre com as medidas do triangulo X: ");
xA = sc.nextDouble();
xB = sc.nextDouble();
xC = sc.nextDouble();
System.out.println("Entre com as medidas do triangulo Y:");
yA = sc.nextDouble();
yB = sc.nextDouble();
yC = sc.nextDouble();

double p = (xA + xB + xC) / 2.0;
double areaX = Math.sqrt(p * (p - xA) * (p - xB) * (p - xC));

p = (yA + yB + yC) / 2.0;
double areaY = Math.sqrt(p * (p - yA) * (p - yB) * (p - yC));

System.out.printf("Area do Triangulo X: %.4f%n", areaX);
System.out.printf("Area do Triangulo Y: %.4f%n", areaY);

if (areaX > areaY) {
    System.out.println("Maior area: X");
}
else {
    System.out.println("Maior area: Y");
}
```

Discussão

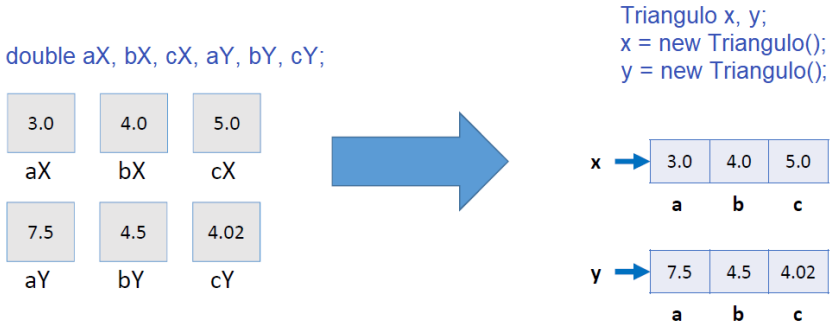
- Triângulo é uma entidade com três atributos: a, b, c.
- Estamos usando três variáveis distintas para representar cada triângulo:
- **Double aX, bX, cX, aY, bY, cY;**
- Para melhorar isso, vamos usar uma CLASSE para representar um triângulo.



Criação da classe

- Temos uma classe que é um **modelo** do que queremos representar

```
public class Triangulo {

    public double a;
    public double b;
    public double c;
}
```



- O modelo até agora inclui apenas características (atributos)
- Vamos adicionar um comportamento (método)

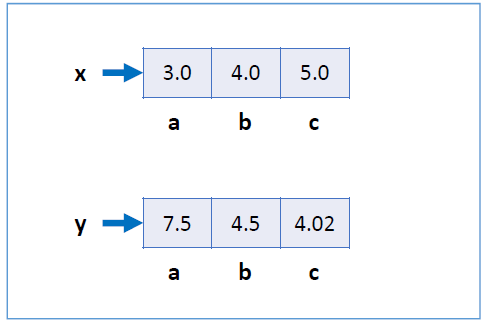
Discussão

- Com o uso de CLASSE, agora nós temos uma variável composta do tipo "**Triangulo**"para representar cada triângulo:

```
Triangulo x, y;
x = new Triangulo();
y = new Triangulo();
```

- Agora vamos melhorar nossa CLASSE, acrescentando nela um MÉTODO para calcular a área.

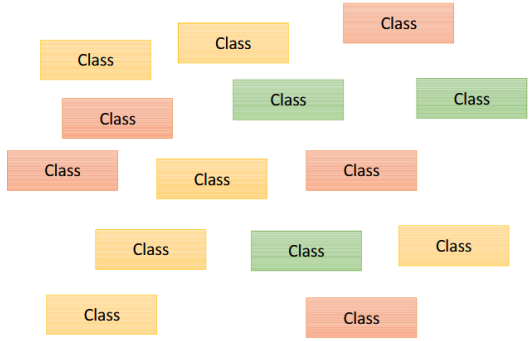
Memória:




```
public class Triangulo {
    public double a;
    public double b;
    public double c;

    public double area() {
        double p = (a + b + c) / 2.0;
        return Math.sqrt(p * (p - a) * (p - b) * (p - c));
    }
}
```

Estrutura de uma aplicação Java

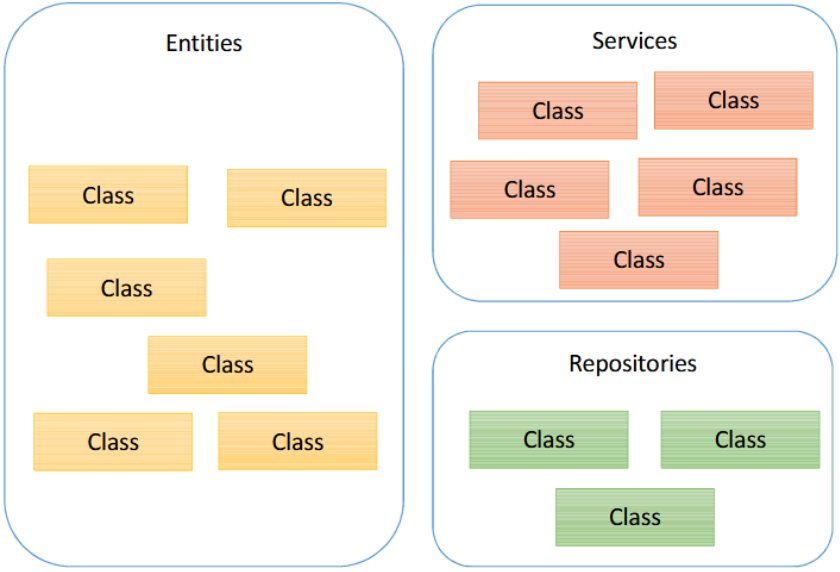
Programa Java: um conjunto de uma ou mais classes.



- Programação orientada a objetos:
  - Preocupa-se com os dados e com as operações que podem ser executadas sobre estes dados que o programa irá tratar.

Programação em JAVA

package = agrupamento LÓGICO de classes relacionadas



Discussão

- Quais são os benefícios de se calcular a área de um triângulo por meio de um MÉTODO dentro da CLASSE Triangulo?
  1. **Reaproveitamento de código:** nós eliminamos o código repetido (cálculo das áreas dos triângulos x e y) no programa principal
  2. **Delegação de responsabilidades:** quem deve ser responsável por saber como calcular a área de um triângulo é o próprio triângulo. A lógica do cálculo da área não deve estar em outro lugar.

Construtor

- Nota-se que criamos o objeto e só em seguida que inserimos valores a seus atributos
- Nós podemos utilizar um **construtor** para que possamos inserir esses valores no **momento de criação do objeto**

```
public class Triangulo {
    public double a;
    public double b;
    public double c;
    public Triangulo(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
}
```

```
System.out.println("Entre com as medidas do triangulo X: ");
double a = sc.nextDouble();
double b = sc.nextDouble();
double c = sc.nextDouble();
Triangulo x = new Triangulo(a,b,c);
```

Construtor

- É uma operação especial da classe, que executa no momento da instanciação do objeto
- Determina que ações devem ser executadas quando a criação de um objeto

Construtor - Problema 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

```
Insira os dados do produto:
Nome: TV
Preço: 900,00
Quantidade em estoque: 10

Dados do produto: TV, R$ 900,00, 10 unidades, Total: R$ 9.000,00

Insira a quantidade de produtos a serem adicionados em estoque: 5
Dados atualizados: TV, R$ 900,00, 15 unidades, Total: R$ 13.500,00

Insira a quantidade de produtos a serem retirados do estoque: 3
Dados atualizados: TV, R$ 900,00, 12 unidades, Total: R$ 10.800,00
```

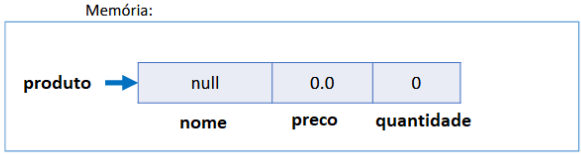
Construtor - Problema Exemplo

```
public class Produto {
    public String nome;
    public double preco;
    public int quantidade;

    public double precoTotalEstoque() {
        return preco * quantidade;
    }
    public void adicionarProduto(int quantidade) {
        this.quantidade += quantidade;
    }
    public void removerProduto(int quantidade) {
        this.quantidade -= quantidade;
    }
}
```

Proposta de melhoria

Quando executamos o comando abaixo, instanciamos um produto com seus atributos “vazios”:

```
Produto produto = 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 e sem preço, é possível fazer com que seja “obrigatória” a iniciação desses valores?

Construtor - Problema Exemplo

```java
public class Produto {
    public String nome;
    public double preco;
    public int quantidade;

    public Produto(String nome, double preco, int quantidade) {
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }
    (...)
}
```

```java
System.out.println("Insira os dados do produto: ");
System.out.print("Nome: ");
String nome = sc.nextLine();
System.out.print("Preço: ");
double preco = sc.nextDouble();
System.out.print("Quantidade em estoque: ");
int quantidade = sc.nextInt();
Produto produto = new Produto(nome, preco, quantidade);
```

Exercício

Criar uma classe Pessoa com os seguintes atributos:

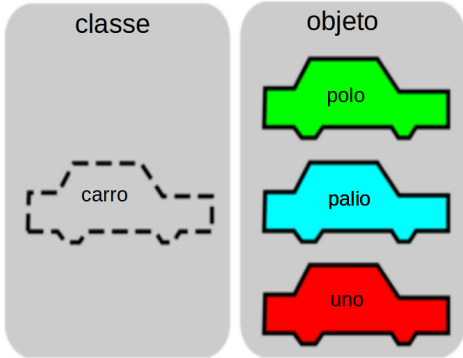
- Nome
- Altura
- Idade
- Peso

A classe também necessita dos seguintes comportamentos

- calcularIMC - Calculo: peso / (altura * altura)
- ehMaiorDeIdade

Programação orientada a objeto

- Você comprou um carro
  - Características: um motor 2.0 híbrido, azul escuro, quatro portas
  - Comportamentos: acelerar, desacelerar, acender os faróis
- Seu carro é um objeto seu mas na loja onde você o comprou existiam vários outros, muito similares
- Seu objeto pode ser classificado como um carro
- Seu carro nada mais é que uma instância dessa classe chamada "carro".



Classe Carro com construtor

```java
public class Carro {
  // Atributo
  public String modelo;
  public String cor;
  public int ano;
  public int velocidade;

  // Construtor
  public Carro(String modelo, String cor, int ano, int velocidade) {
    this.modelo = modelo;
    this.cor = cor;
    this.ano = ano;
    this.velocidade = velocidade;
  }

  // Comportamentos
  public void acelera() {
    velocidade++;
  }

  public void freia() {
    velocidade--;
  }
}
```

Construtor determina que ações devem ser executadas quando a criação de um objeto

Construtor

- Usos comuns:
  - Iniciar valores dos atributos
  - Permitir ou obrigar que o objeto receba dados no momento de sua instanciação
- No programa principal

```
static void Main(string[] args)
{
   // Instâncias
   Carro polo =    new Carro("polo", "verde", 2010, 10);
   Carro uno =     new Carro("uno", "branco", 2016, 5);

   polo.acelera();
   uno.aceleta();
}
```

Encapsulamento

- É um princípio que consiste em esconder detalhes de implementação de uma classe, expondo apenas operações seguras e que mantenham os objetos em um estado consistente.
- É uma forma eficiente de proteger os dados manipulados dentro da classe, além de determinar onde esta classe poderá ser manipulada.

```
static void Main(string[] args)
{
   // Instâncias
   Carro polo =    new Carro("polo", "verde", 2010, 10);
   Carro uno =     new Carro("uno", "branco", 2016, 5);

   // É possivel acessar e modificar atributos do objeto
   uno.ano = -2020;
   polo.velocidade = -1000;
}
```

- Um objeto NÃO deve expor nenhum atributo (modificador de acesso private)
- Os atributos devem ser acessados por meio de métodos get e set

Porque usar o encapsulamento??

- Proteção dos dados: O encapsulamento protege os dados de uma classe, impedindo que eles sejam modificados diretamente por código externo. Apenas os métodos da própria classe podem acessar e modificar esses dados, o que ajuda a evitar alterações acidentais ou não autorizadas nos dados.
- Abstração: Encapsulamento permite que os detalhes internos de implementação de uma classe sejam ocultados. Isso significa que os usuários da classe só precisam saber como usar os métodos públicos disponíveis, sem precisar entender como eles são implementados internamente.

Porque usar o encapsulamento??

- Classe **ContaBanco**

```
public class ContaBanco {
    public String cliente;
    public double saldo;

    public void depositar(double valor) {
        saldo += valor;
    }

    public void sacar(double valor) {
        saldo -= valor;
    }
}
```

- Na classe principal

```
    ContaBanco conta = new ContaBanco();
conta.saldo = -1000000; // definindo saldo diretamente
```

Porque usar o encapsulamento??

Agora, os atributos cliente e saldo são privados e só podem ser acessados e modificados por métodos da classe. Isso garante que o saldo seja sempre consistente e evita acesso direto a ele de fora da classe.

```
public class ContaBanco {
    private String cliente;
    private double saldo;
    public ContaBanco(String cliente, double saldoInicial) {
        this.cliente = cliente;
        this.saldo = saldoInicial;
    }
    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
        } else {
            System.out.println("Valor inválido para depósito.");
        }
    }
    public void sacar(double valor) {
        if (valor > 0 && saldo >= valor) {
            saldo -= valor;
        } else {
            System.out.println("Saldo insuficiente ou valor inválido para saque.");
        }
    }
    public double getSaldo() {
        return saldo;
    }
}
```

Padrão para implementação de getters e setters

```
public class Produto {
    private String nome;
    private double preco;
    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }

    public double getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public double getPreco() {
        return preco;
    }

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

Encapsulamento - Classe Pessoa

```
public static void main(String[] args) {
        // Criando objetos Pessoa usando o construtor
        Pessoa pessoa1 = new Pessoa("João", 1.70 , 25, 85);
        Pessoa pessoa2 = new Pessoa("Maria", 1.65, 30, 65);

        // Acessando informações das pessoas
        System.out.println("Pessoa 1:");
        System.out.println("Nome: " + pessoa1.getNome());
        System.out.println("Idade: " + pessoa1.getIdade());

        System.out.println("\nPessoa 2:");
        System.out.println("Nome: " + pessoa2.getNome());
        System.out.println("Idade: " + pessoa2.getIdade());

        // Modificando idade da pessoa1 usando o setter
        pessoa1.setIdade(26);
        System.out.println("\nDepois de alterar a idade de Pessoa 1:");
        System.out.println("Nome: " + pessoa1.getNome());
        System.out.println("Nova Idade: " + pessoa1.getIdade());
    }
```

Comandos

Gerando automaticamente construtores, getters e setters com Eclipse

- Botão direito -> Source -> Generate Constructor using Fields
- Botão direito -> Source -> Generate Getters and Setters
- Botão direito -> Source -> Generate toString

Desafios

Desafio 1

Fazer um programa para ler um número inteiro N e depois os dados (id, nome e salario) de N funcionários. Não deve haver repetição de id. Em seguida, efetuar o aumento de X por cento no salário de um determinado funcionário. Para isso, o programa deve ler um id e o valor X. Se o id informado não existir, mostrar uma mensagem e abortar a operação. Ao final, mostrar a listagem atualizada dos funcionários, conforme exemplos. Lembre-se de aplicar a técnica de encapsulamento para não permitir que o salário possa ser mudado livremente. Um salário só pode ser aumentado com base em uma operação de aumento por porcentagem dada.

```
Quantos funcionários serão cadastrados? 3

Funcionário #1:
Id: 333
Nome: Maria Brown
Salário: 4000.00
Funcionário #2:
Id: 536
Nome: Alex Grey
Salário: 3000.00
Funcionário #3:
Id: 772
Nome: Bob Green
Salário: 5000.00
Informe o Id do funcionário que terá aumento salarial : 536
Digite a porcentagem:  10.0
Lista de funcionários:
333, Maria Brown, 4000.00
536, Alex Grey, 3300.00
772, Bob Green, 5000.00
```

Desafio 2

Fazer um programa para ler nome, idade e altura de N pessoas, conforme exemplo. Depois, mostrar na tela a altura média das pessoas, e mostrar também a porcentagem de pessoas com menos de 16 anos, bem como os nomes dessas pessoas caso houver.

```
Quantas pessoas serao digitadas? 5
Dados da 1a pessoa:
Nome: Joao
Idade: 15
Altura: 1.82

Dados da 2a pessoa:
Nome: Maria
Idade: 16
Altura: 1.60

Dados da 3a pessoa:
Nome: Teresa
Idade: 14
Altura: 1.58

Dados da 4a pessoa:
Nome: Carlos
Idade: 21
Altura: 1.65

Dados da 5a pessoa:
Nome: Paulo
Idade: 17
Altura: 1.78

Altura média: 1.69
Pessoas com menos de 16 anos: 40.0%
Joao
Teresa
```

Desafio 3

A dona de um pensionato possui dez quartos para alugar para estudantes, sendo esses quartos identificados pelos números 0 a 9. Fazer um programa que inicie com todos os dez quartos vazios, e depois leia uma quantidade N representando o número de estudantes que vão alugar quartos (N pode ser de 1 a 10). Em seguida, registre o aluguel dos N estudantes. Para cada registro de aluguel, informar o nome e email do estudante, bem como qual dos quartos ele escolheu (de 0 a 9). **Um quarto ocupado não pode ser alugado!** Ao final, seu programa deve imprimir um relatório de todas ocupações do pensionato, por ordem de quarto, conforme exemplo.

```
Quantos quartos serão alugados?  3

Aluguel  #1:
Quarto: 5
Nome: Maria Green
Email: maria@gmail.com

Aluguel  #2:
Quarto: 1
Nome: Marco Antonio
Email: marco@gmail.com

Aluguel  #3:
Quarto: 8
Nome: Alex Brown
Email: alex@gmail.com

Quartos ocupados:
1: Marco Antonio, marco@gmail.com
5: Maria Green, maria@gmail.com
8: Alex Brown, alex@gmail.com
```