Programação Orientada a Objetos

Professor: Anderson Elias

Estácio

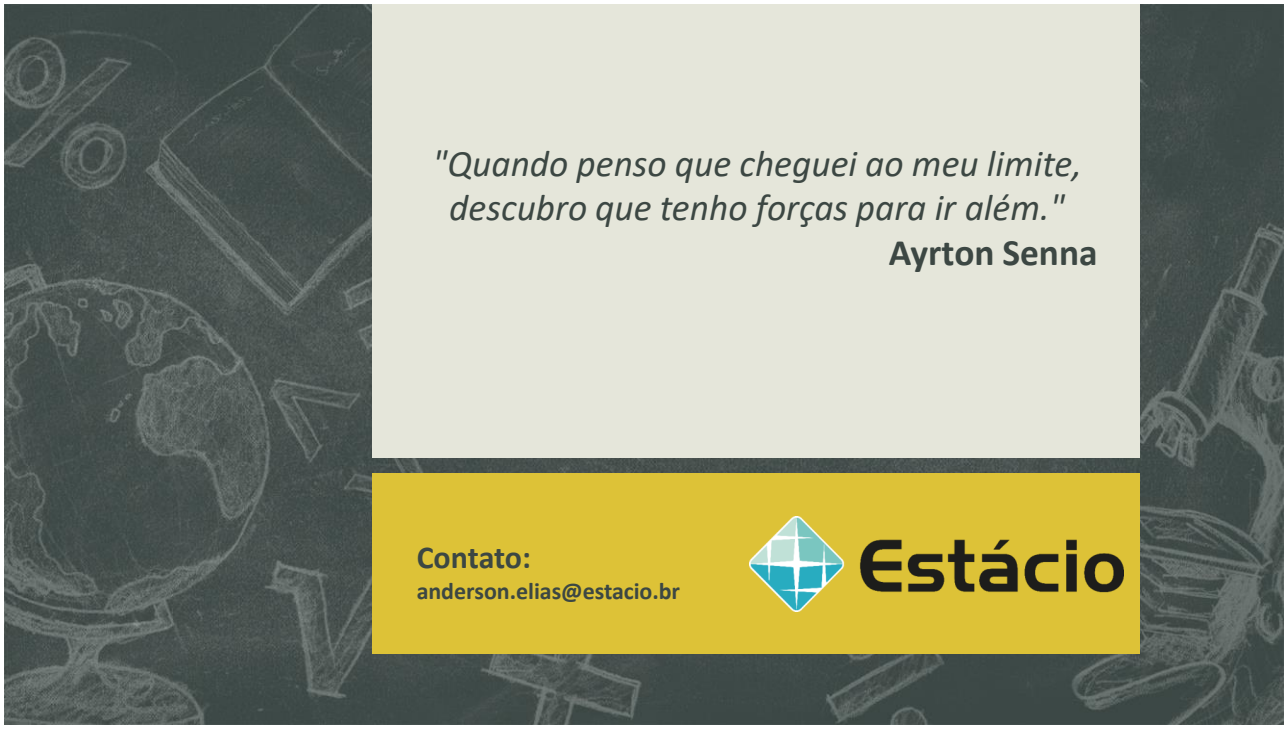
1

Conceitos Básicos

Contato:
anderson.elias@estacio.br

Estácio

2



3

Roteiro – Conceitos Básicos

- Exercícios de Revisão (PI)
- POO Conceitos
- Abstração e Princípios
- Objetos e Classes
- Interfaces e Classes Internas
- Construtores;
- Modificadores de Acesso;
- Serialização
- Sobrescrita / Sobercarga (Overriding / Overloading)
- Tratamento de Exceções
- Herança;
  - Classes, Superclasses e Subclasses;
  - Polimorfismo / Encapsulamento;
  - Vinculação dinâmica;
  - Evitando herança: Classes e métodos **Final**
  - Coerção;
  - Interfaces e Classes abstratas;
  - Acesso protegido;
- Coleções
  - List
  - Array List
  - Set
  - Map

4

Exercício - Objetos e Classes

USE MÉTODOS PARA REALIZAR AS OPERAÇÕES.

1) Escrever um programa java que receba dois números e exiba o resultado da sua soma.

2) Escrever um programa que receba dois números e ao final mostre a soma, subtração, multiplicação e a divisão dos números lidos.

3) Escrever um programa para determinar o consumo médio de um automóvel sendo fornecida a distância total percorrida pelo automóvel e o total de combustível gasto.

4) escrever um programa que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha 15% de comissão sobre suas vendas se o salário for maior ou igual a R$ 1.000,00 caso contrário a comissão será de 20%. Como saída, informar o seu nome, o salário fixo e salário no final do mês.

5

Exercício - Objetos e Classes

5) Escrever um programa que leia o nome de um aluno e as notas das três provas que ele obteve no semestre. No final informar o nome do aluno e a sua média (aritmética).

6) Escrever uma programa em que leia dois valores para as variáveis A e B, e efetuar as trocas dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresentar os valores trocados. Lembrando que temos que ter 3 variáveis para que uma haja como variável de armazenamento, ou seja, para evitar que o valor original de a se perca é necessário associar a uma outra variável (denominada usualmente de variável auxiliar) tal valor, estabelecer uma associação de A com o valor em B e, por último, associar B ao valor "salvo" na variável auxiliar.

7) Ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é: F=(9*C+160) / 5, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.

6

Exercício - Objetos e Classes

8) Elaborar um programa que efetue a apresentação do valor da conversão em real (R$) de um valor lido em dólar (US$). O algoritmo deverá solicitar o valor da cotação do dólar e também a quantidade de dólares disponíveis com o usuário.
9) Faça um programa que receba um valor que foi depositado e exiba o valor com rendimento após um mês. Considere fixo o juro da poupança em 0.70% a. m.
10) A Loja Mamão com Açúcar está vendendo seus produtos em 5 (cinco) prestações sem juros. Faça um programa que receba um valor de uma compra e mostre o valor das prestações.

7

Exercício - Objetos e Classes

11) Faça um programa que receba o preço de custo de um produto e mostre o valor de venda. Sabe-se que o preço de custo receberá um acréscimo de acordo com um percentual informado pelo usuário.
12) Escreva um programa para ler o raio de um circulo, calcular e escrever a sua área. (pR^2) Calculo: área= PI*(raio²)
13) Escreva um programa que entre com um número e o imprima caso seja maior do que 20.
14) Entrar com dois números e imprimir o menor número (suponha números diferentes).
15) Entrar com dois números e imprimi-los em ordem decrescente (suponha números diferentes).

8

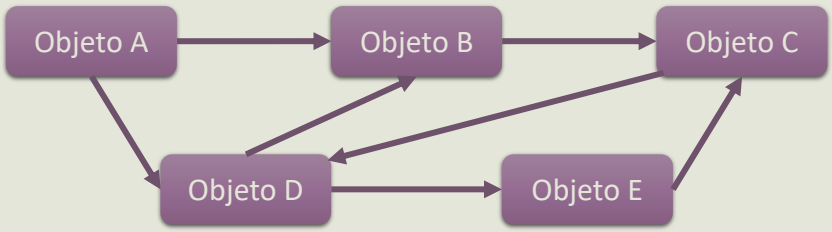
Exercício - Objetos e Classes

- Criar uma solução para alteração da folha de pagamento de uma empresa.
- Criar uma classe Empregados com os dados “Nome, função e Salário” com métodos de leitura para todos os dados e apenas o de atualização para o de salário.
- Criar uma classe EmpregadosTeste que informará os nomes dos cinco empregados, suas funções e seus salários, bem como o percentual a ser acrescido no salário atual.
- Imprimir todos os dados dos empregados com os salários atualizados.

9

Programação Orientada a Objetos (POO)

- A Programação Orientada a Objetos é um novo paradigma para desenvolver softwares;



10

(POO) Conceitos

- É uma forma especial de programar, mais próximo de como expressaríamos as coisas na vida real do que outros tipos de programação.

> "*Uma nova maneira de pensar os problemas utilizando conceitos do Mundo Real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade.*"
>
> ***Raumbaugh***

11

(POO) Conceitos

> "*Sistema orientado a objetos é uma coleção de objetos que interagem entre si.*" ***Bertrand Meyer***

- Este paradigma visualiza e representa o mundo real através de um **conjunto de objetos.**
- Estes objetos interagem entre si para que determinadas operações sejam realizadas;

12

(POO) I

- Diferenças entre os paradigmas:

Programação Estruturada		Orientação a Objeto
Processo	⟷	Objeto
Revela Dados	⟷	Oculta Dados
Projeto Monolítico	⟷	Projeto Modular
Uso Único	⟷	Reutilização
Algoritmo Ordenado	⟷	Algoritmo Desordenado

13

(POO) Abstração

Abstraindo Objetos...







14

(POO) O que é Abstração?

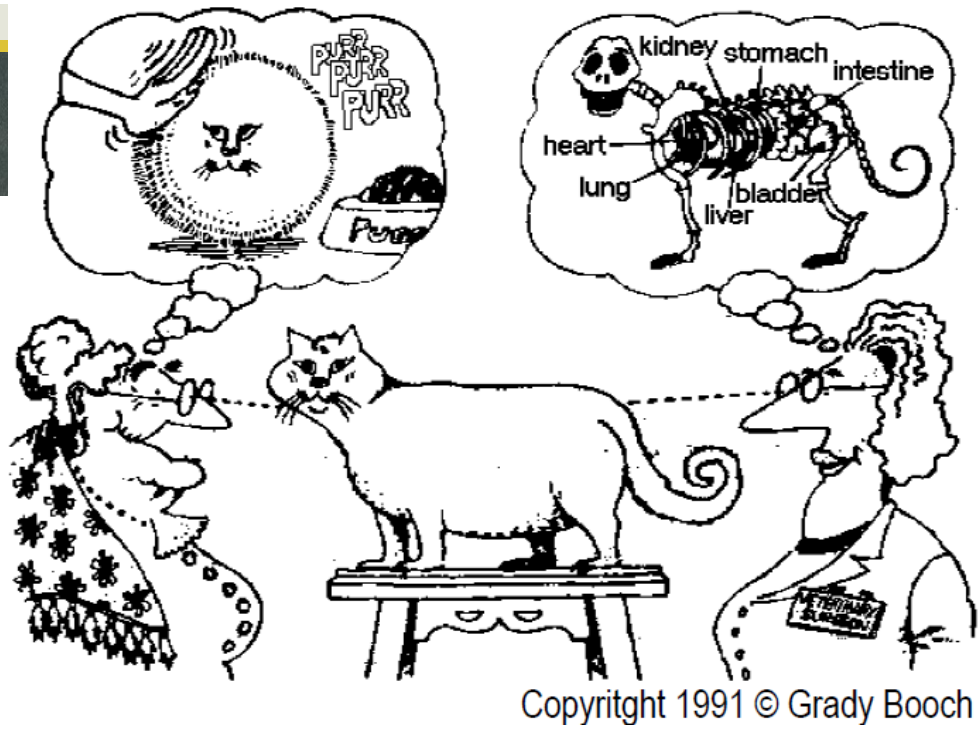
Abstração é uma visualização ou representação de uma entidade que inclui somente atributos de importância em um contexto particular.

Robert Sebesta

15

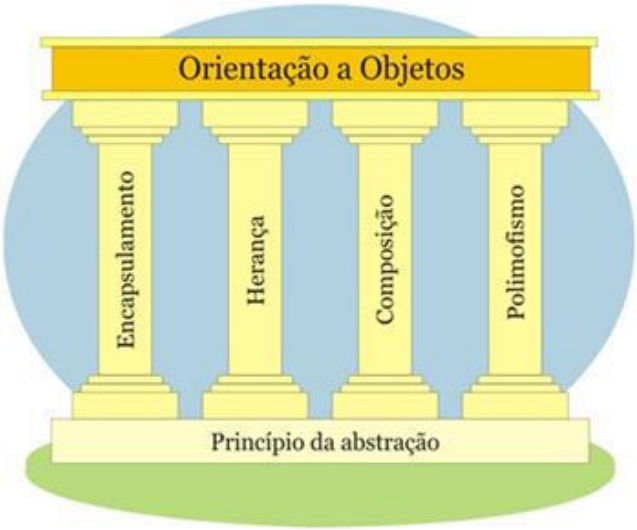
(POO)

Abstração



16

(POO) Princípios da Abstração



A programação orientada a objetos está bem sedimentada sobre os quatro pilares derivados do **princípio da abstração:**

- *Encapsulamento;*
- *Herança;*
- *Composição;*
- *Polimorfismo.*

17

(POO) Princípios da Abstração

O princípio da abstração é a capacidade de abstrair a complexidade de um sistema e se concentrar em apenas partes dele.

Exemplo:
Um médico torna-se especialista em cirurgia do coração, portanto ele abstrai sem considerar as influencias dos outros órgãos e foca apenas sua atenção no coração.

18

(POO) Princípios da Abstração

Abstração seria o ato de separar um ou mais elementos de uma totalidade complexa;

Ignorar aspectos não relevantes;

Outro exemplos do uso de abstração no nosso dia a dia:

- *Dirigir um carro.*
- *Usar um som ou TV.*

19

(POO) Princípios da Abstração

Observando ao nosso redor, veremos várias entidades ou abstrações que podem ser representadas como **objetos** em nosso programa.

Estes objetos podem ter

- **Características**: *Pelas quais os identificamos.*
- **Funcionalidades**: *Para as quais os utilizamos.*

20

(POO) Princípios da Abstração

Estas **Características** dos objetos são também chamadas de **Atributos**.

Exemplo:

- **Objeto Carro**: *Possui ano, marca, modelo, cor, peso, preço.*
- **Objeto Aluno**: *Matrícula, nome, endereço, idade, telefone.*

21

(POO) Princípios da Abstração

Os objetos podem ter **Funcionamentos/Comportamentos** associados a eles.

Exemplo:

- **Objeto Carro**: *Pode ligar, desligar, acelerar, frear.*
- **Objeto Aluno**: *Andar, correr, dirigir carro, realizar matrícula.*

22

(POO) Princípios da Abstração

Podemos dizer então que em POO os objetos possuem características e comportamentos.

Estas características chamamos de **atributos**;
Os comportamentos conhecemos como **métodos**.

23

(POO) Princípios da Abstração

Exemplo de objetos com suas características (atributos) e comportamentos (métodos):

Característica:(atributos):
ano, marca, modelo, cor, peso, preço

Comportamento (métodos)
ligar, desligar, acelerar, frear

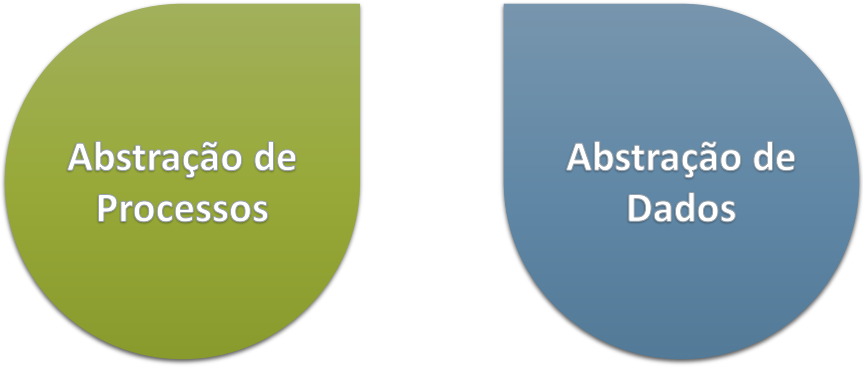
Característica:(atributos):
Matrícula, nome, endereço, idade, telefone

Comportamento (métodos)
Andar, correr, dirigir carro, realizar matrícula

24

(POO) Abstração na Programação

Como seria estas abstrações na programação?



25

(POO) Abstração de Processos

Abstração de Processos

Tem como finalidade dividir um programa em ***subprogramas*** menores para tornar mais fáceis de escrever e compreender.

(“dividir para conquistar”)

Para usar um subprograma escrito por terceiros, ***abstraímos a sua implementação*** e nos concentramos na sua ***API***.

26

(POO) Abstração de Processos

Abstração de Processos - Exemplo

```
public class Funcao {
   public static void main(String[]args){
      System.out.println(soma(2, 3));
   }
   public static int soma(int x, int y){
      int a=x, b=y, c;
      c = a + b;
      return c;
   }
}
```

Quer realizar algo

Abstração

27

(POO) Abstração de Dados

Abstração de Dados

Representação de ***entidades reais do domínio do problema numa linguagem de programação***.

- *Identificando as propriedades destas entidades que interessam ao sistema bem como suas operações*

28

(POO) Abstração de Dados

Abstração de Dados- Exemplo

```
package reg;
class Aluno {
    public String nome;
    public int idade;
    public double nota;

    public double atualizarNota(){
        return this.nota + 0.5;
    }
}
```

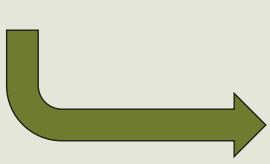
Propriedades (*Atributos*)

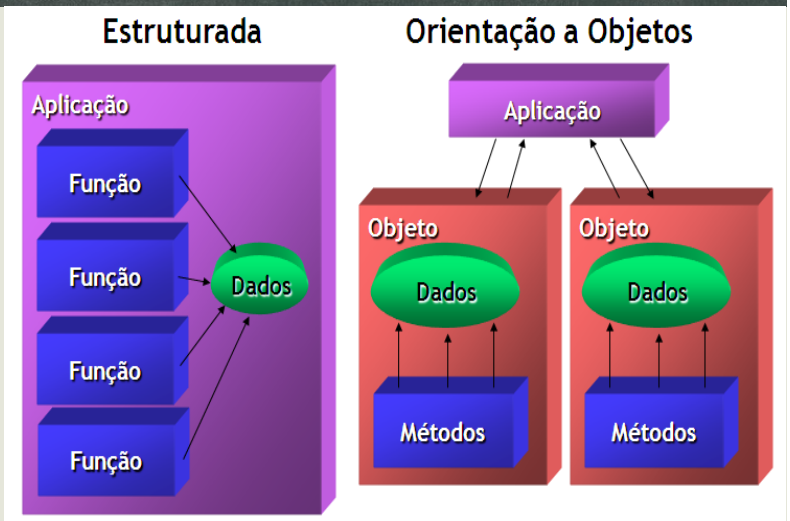
Comportamento (*Métodos*)

29

(POO) Comparativo com Procedural

- Diferenças entre os paradigmas:





30

(POO) Comparativo com Procedural

Base da Programação Estruturada:

Sequência: uma tarefa é executada após a outra, linearmente;

Decisão: a partir de um teste lógico, determinado trecho de código é executado, ou não;

Repetição: a partir de um teste lógico, determinado trecho de código é repetido por um número finito de vezes.

31

(POO) Comparativo com Procedural

Vantagens da Programação Estruturada:

Fácil de entender: Ainda muito usada em cursos introdutórios de programação;

Execução mais rápida: por não haver diversas camadas.

Desvantagens:

Baixa reutilização de código;

Códigos confusos: Dados misturados com comportamento;

Difícil manutenção.

32

(POO) Comparativo com Procedural

Base da Programação Orientada a Objetos:

- Classes e Objetos
- Métodos e Atributos

Permite modelar o mundo real em software:

- Cada elemento é representado em forma de objeto
- Um objeto contém características e ações, assim como vemos na realidade

33

(POO) Comparativo com Procedural

Vantagens da Programação Orientada a Objetos:

- Reuso de bibliotecas prontas (APIs) e de projetos de terceiros;
- Maior produtividade: multidesenvolvimento;
- Baixo custo;
- Melhor organização do código (modularização) Fácil leitura e manutenção;
- Permite construir sistemas mais complexos;
- Desenvolvimento em nível mais alto de abstração;
- Maior qualidade no produto final.

34

(POO) Comparativo com Procedural

Desvantagens da Programação Orientada a Objetos:

- Execução mais lenta de aplicações, mas é pouco percebida;
- Complexidade no aprendizado;
- Maior esforço na modelagem de um sistema OO do que estruturado:
  - Porém, menor esforço de programação

35

(POO

Programação orientada a objetos	Programação estruturada
Métodos	Procedimentos e funções
Instâncias de variáveis	Variáveis
Mensagens	Chamadas a procedimentos e funções
Classes	Tipos de dados definidos pelo usuário
Herança	Não disponível
Polimorfismo	Não disponível

36

(POO) Encapsulamento

Encapsulamento

"É o processo de **esconder todos os detalhes** de um objeto **que não contribuem** para suas características essenciais."

"**Nenhuma parte** de um sistema complexo **deve depender** dos **detalhes internos** das outras partes."

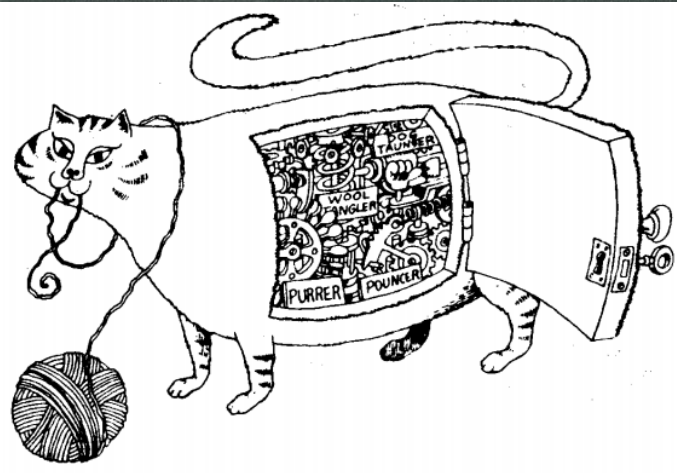
"Para uma **abstração dar certo** sua **implementação** deve estar **encapsulada**."

Grady Booch

37

(POO) Encapsulamento

Encapsulamento



Copyritght 1991 © Grady Booch

38

(POO) Encapsulamento

Encapsulamento X Abstração

- São conceitos complementares;
- A abstração representa um conceito: o comportamento observável de um objeto;
- O encapsulamento impede os clientes de verem como este conceito foi implementado

39

(POO) – Por que Encapsular?

- **Reutilizar métodos em qualquer parte do sistema**
  - Ex.: Método debitar() de uma conta usado para saque e transferência
- **Alterar métodos sem impactar o sistema**
  - Ex.: Posso alterar como o debitar() foi implementado
- **Evitar mal uso de operações do sistema**
  - Ex.: Não permitir que um aluno altere sua nota

40

(POO) – Encapsulamento (Exemplo)

- Em uma classe chamada **OperationMath** há ***dois atributos do tipo float*** que recebe os valores para realizar as operações aritméticas. Esta classe deve proteger através do encapsulamento seus atributos e a forma como realiza suas operações aritméticas, podendo ser: ***add, sub, mult e div***.
- Haverá um método público que receberá os valores e repassará para os atributos da classe. Este método fará uma checagem para qual tipo de operação deve realizar.
- Finalizando, crie uma outra classe chamada **Principal** que terá um objeto da classe anterior e fara uso do método público para realizar qualquer uma das quatro operações aritméticas.

41

```
public class OperationMath {
  private float a, b;
  public void realizarOperacao(float x, float y, String op) {
    a = x;
    b = y;
    switch (op) {
      case "+":
        System.out.println(add(a,b));
        break;
      case "-":
        System.out.println(sub(a,b));
        break;
      case "*":
        System.out.println(mult(a,b));
        break;
      case "/":
        System.out.println(div(a,b));
        break;
      default:
        System.out.println("Operação realizada.");
    }
  }
```

CONTINUAÇÃO...

```
  private float add(float z, float w) {
    return z + w;
  }
  private float sub(float z, float w) {
    return z - w;
  }
  private float mult(float z, float w) {
    return z * w;
  }
  private float div(float z, float w) {
    return z / w;
  }
} //Fecha a Classe
```

42

(POO) – Encapsulamento

TESTANDO...

```
public class Principal {
  public static void main(String[]args){
    OperationMath op = new OperationMath();
    op.realizarOperacao(2, 3, "*");
  }
}
```

43

(POO) – Encapsulamento (Exemplo)

- Digamos que desejamos obter os valores dos atributos da classe OperationMath que estão invisíveis(***encapsulados***) para qualquer outra classe?

Podemos nos utilizar dos Get e Set além do toString para isto.

- Portanto, na classe OperationMath, devemos criar estes mecanismos de acesso e escrita para os atributos privados da classe.
- O método ***Get*** e ***toString*** Obtém os valores dos atributos, já o ***Set*** altera o valor deles.

Para usar, basta Source/Generate Getters and Setters. Depois toString

44

(POO) – Modularização

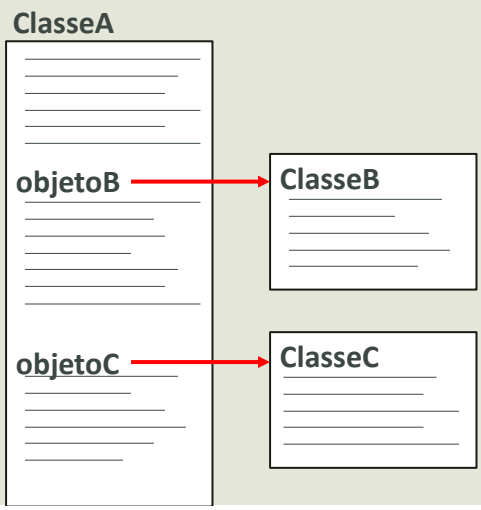
- Processo de dividir um todo em partes bem definidas;
  - *Que podem ser construídas e examinadas separadamente.*
- Essas partes interagem entre si, fazendo com que o sistema funcione de forma adequada;
- Particionar um programa em componentes individuais pode reduzir a complexidade.

45

Modularização

Premissas da Decomposição:

- Dividir o problema em suas partes principais;
- Analisar a divisão obtida para garantir coerência;
- Se alguma parte ainda permanecer complexa, decompô-la também;
- Analisar o resultado para garantir entendimento e coerência.



46

(POO) – Modularização

Vantagens:

- Cada divisão possui um código mais simplificado;
- Facilita o entendimento: divisões independentes;
- Códigos menores são mais fáceis de ser modificados
- Desenvolvimento do sistema através de uma equipe de programadores;
- Reutilização de trechos de códigos

47

(POO) – Elementos Básicos

- Objetos
- Instâncias
- Classes

48

(POO) – Objetos

Na programação orientada a objetos, os Objetos são usados para representar entidades do mundo real ou computacional.

Observando ao nosso redor, percebemos várias entidades ou abstrações as quais podem ser representadas como objetos no nosso programa.

- Pessoas, carros e casas podem ser vistos como objetos.

49

(POO) – Objetos

Um objeto é a entidade que realiza um tipo ou uma coleção de tipos (tipo composto) através de suas propriedades e comportamento.

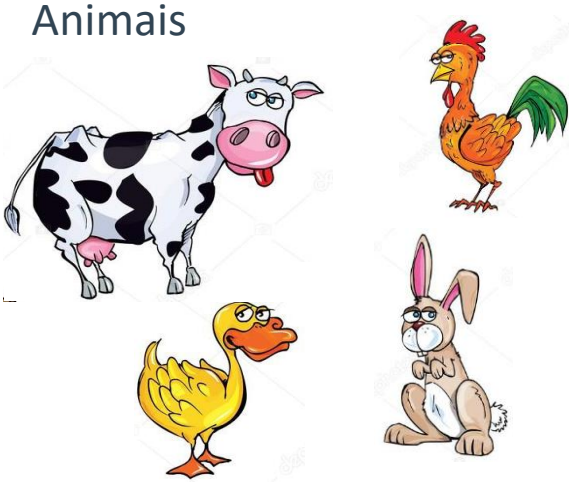
Um objeto representa um elemento do problema real modelado a partir dos tipos que implementa.

Um objeto é uma variável do tipo de uma classe definida pelo usuário. Ou seja, **um objeto é uma instância de uma classe**.

50

(POO) – Classificando os Objetos

Animais



Seres Humanos

51

(POO) – Objetos

Um objeto tem estado que é um elemento particular e personificado da classe.

Podemos dizer que um objeto possui três partes:

- Estado
- Comportamento
- Identidade

52

(POO) – Estado do Objeto

O estado de um objeto é uma das condições em que ele pode existir.

É uma característica transitória.

Ele normalmente muda com o decorrer do tempo e é caracterizado pelos valores instantâneos dos seus atributos e de suas ligações e relacionamentos com outros objetos.

53

(POO) – Estado do Objeto

Exemplo:

Um objeto lâmpada pode ter basicamente dois estados:

acesa e apagada

E que só podem ser alterados através das ações:

acender ou apagar

Estas ações fariam com que o valor do atributo de nome aceso variasse entre verdadeiro, quando a lâmpada estivesse acesa e falso, quando estivesse apagada.

54

(POO) – Comportamento do Objeto

É definido pelo conjunto de seus métodos, ou seja, pelo conjunto das ações que este objeto pode executar e pela forma que ele responde às chamadas de outros objetos.

Determina como o objeto age e reage às requisições de outros objetos.

55

(POO) – Comportamento do Objeto

Exemplo:

Um objeto pessoa possui alguns comportamentos padrões:
comer, beber, andar, etc.

Ao enviar uma mensagem para um objeto pessoa do tipo *comer*, ele saberá como responder e executar a ação. Mas se você mandar uma mensagem do tipo *voar*, esse objeto não irá responder, pois *voar* não faz parte de seu comportamento.

56

(POO) – Instâncias

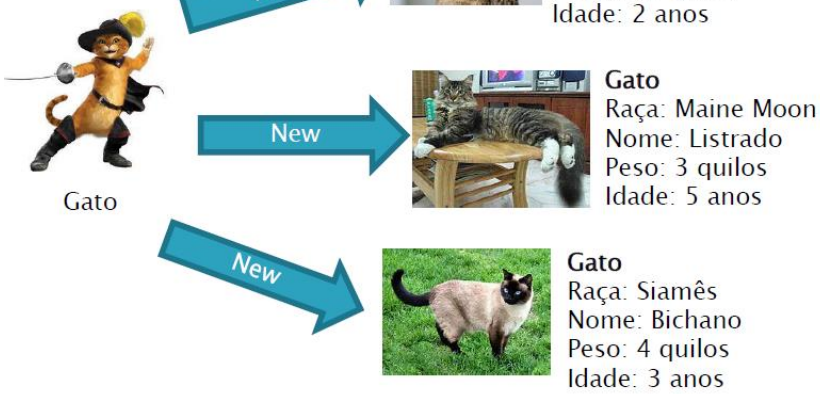
A instância em POO é o momento em que um objeto é criado a partir de uma classe.

A classe é uma espécie de forma e ao criar um ou mais objetos desta mesma classe, diz-se que há instâncias desta classe na forma de objetos. Portanto estes objetos tem características e comportamentos comuns.

57

(POO) – Instâncias

Exemplo:



58

(POO) – Classes

Em POO o conceitos de **Classe**, se dá na modelagem de programação de um conjunto de objetos que possuem características (atributos) e comportamentos (métodos) comuns.

- Cada classe funciona no fundo como um molde para a criação de um dado objeto ("*fábrica de objetos*");
- Os objetos são vistos como representações concretas (instâncias) das classes.

59

(POO) – Classes

- Uma classe determina um conjunto de objetos com:
  - Propriedades semelhantes
  - Comportamentos semelhantes
  - Relacionamentos comuns com outros objetos
- Em uma classe encontramos duas divisões:
  - Estrutura: as informações inerentes à classe.
  - Comportamento: as operações realizadas pela classe.

60

(POO) – Classes

- A classe implementa um ou mais tipos , estabelecendo propriedades e comportamento.
- Define o aspecto genérico de um objeto. Todo objeto pertence a uma classe.
- A classe é a ideia a partir do qual o objeto se concretiza.
- Uma classe pode especializar e agregar outras classes para formalizar uma ideia.

61

(POO) – Classes

A classe define que objetos devem ter *ano, marca modelo, cor, peso e preço* mas não indica explicitamente quais são seus valores.

Ano = ?
Marca = ?
Modelo = ?
Cor = ?
Peso = ?
Preço = ?

62

(POO) – Classes

Dois diferentes carros foram criados tomando como base a estrutura da classe:

Ano = 2015
Marca = VW
Modelo = Novo Fox
Cor = Azul
Peso = 1.5Ton
Preço = 45Mil

Ano = 2017
Marca = VW
Modelo = Fox Pepper
Cor = Vermelha
Peso = 1.75ton
Preço = 51.5Mil

63

(POO) – Classes

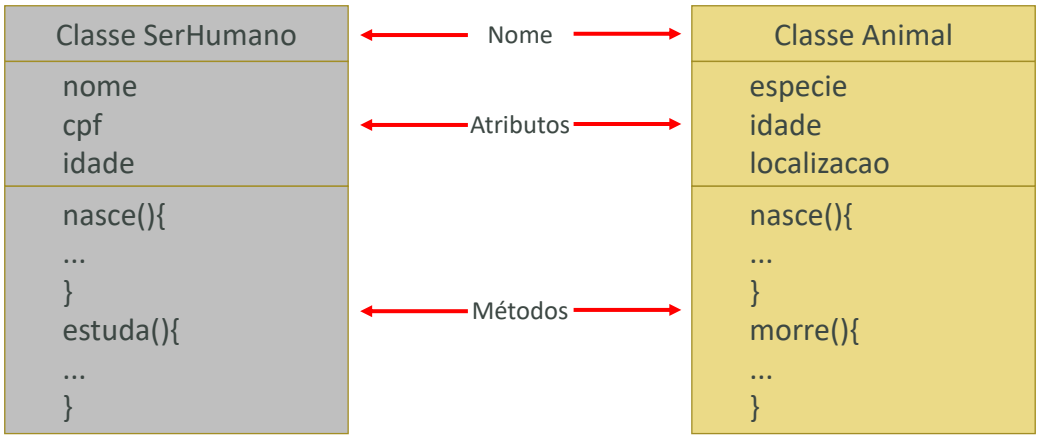
Uma classe é formada por:

- Por um nome
- Por atributos
- Por métodos

64

(POO) – Classes

Exemplos:



65

(POO) – Referências para Objetos

A maioria dos objetos em um programa Java são acessados por variáveis chamadas referências

- Como Java é fortemente tipada, estas variáveis devem ser declaradas e tipificadas em tempo de compilação
- Exemplos:

```
String s;       //s é do tipo String
Veiculo v;      //v é do tipo Veiculo
Triangulo t;    //t é um Triângulo
```

66

(POO) – Referências para Objetos

A declaração indica o tipo de uma referência

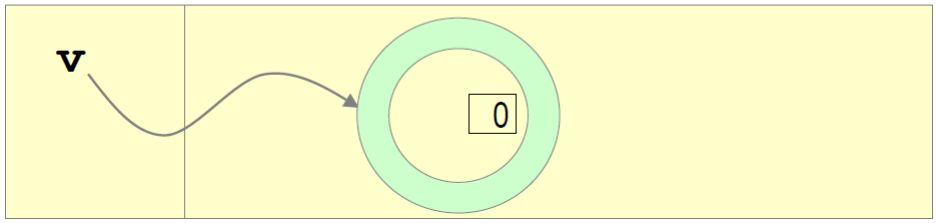
```
Veiculo v;                  //declaração
v = new Veiculo();          //instanciação
v.setNumLugares(50);        //uso
v = null;                   //desprezo
```

67

(POO) – Referências para Objetos

A instanciação cria o objeto na memória.

```
Veiculo v;                  //declaração
v = new Veiculo();          //instanciação
v.setNumLugares(50);        //uso
v = null;                   //desprezo
```



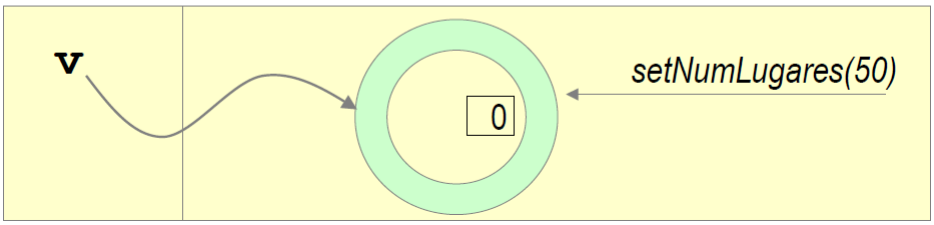
68

(POO) – Referências para Objetos

Uma mensagem é enviada ao objeto.

E faz o método correspondente ser executado.

```
Veiculo v;                  //declaração
v = new Veiculo();          //instanciação
v.setNumLugares(50);        //uso
v = null;                   //desprezo
```

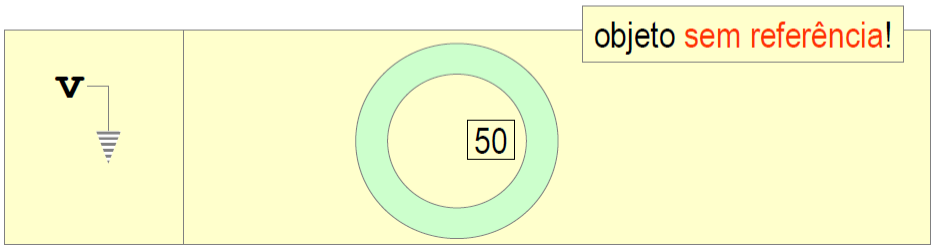


69

(POO) – Referências para Objetos

A referência pode deixar de apontar p/ o objeto.

```
Veiculo v;                  //declaração
v = new Veiculo();          //instanciação
v.setNumLugares(50);        //uso
v = null;                   //desprezo
```



70

(POO) – Exercícios

Lista 2

71

Dúvidas?

72