Universidade Federal de Ouro Preto

CSI032 - Programação de Computadores II

Introdução a Linguagem Orientada Objeto

Professor: Dr. Rafael Frederico Alexandre
Contato: rfalexandre@ufop.edu.br
Colaboradores: Renan Saldanha Contatos:
renansaldlinhares@gmail.com

Agenda

Agenda

Introdução

Modularização.

Conceito

Dividir o software em conjunto de instruções da linguagem que realizam alguma tarefa, constituindo um procedimento algorítmico, com uma função bem definida e o mais independente possível em relação ao restante do programa.

Vantagens

1. Legibilidade;
2. Manutenbilidade;
3. Produtividade.

Introdução

Abstração.

Abstração

Habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes.

Exemplos

1. *printf*;
2. *scanf*.

Introdução

Encapsulamento.

Conceito [Booch, 1995]

Encapsulamento é o processo de esconder todos os detalhes de um objeto que não contribuem para as suas características essenciais.

Vantagens

1. Desacoplamento;
2. Ocultação de Informações.

Exemplos

1. Bibliotecas tais como *stdio.h*, *string.h* e etc;
2. Classes em linguagens Orientada a Objeto;

Introdução

Relacionamento com Java

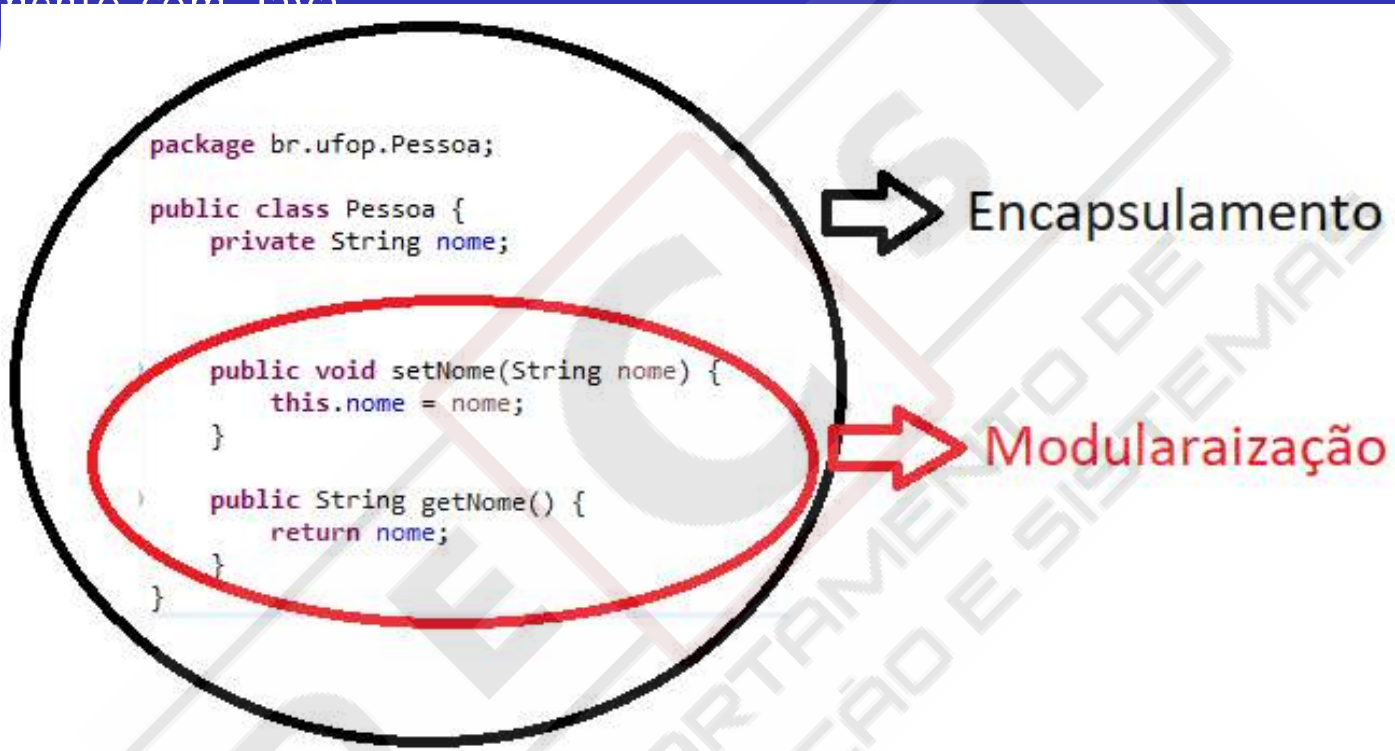


Figura: Modularização e encapsulamento em Java.

Introdução

Relacionamento com Java

Figura: Exemplo abstração.

Agenda

Características da Linguagem Java

Tipos de Variáveis

Conceito.

Espaço na memória do computador destinado a um dado que é alterado durante a execução do algoritmo. Para funcionar corretamente, as variáveis precisam ser definidas por nomes e tipos.

- Existem dois tipos de Variáveis em Java:
  - Variáveis Primitivas: Variáveis implícitas do Java com capacitade de armazenamento de um valor definido;
  - Variáveis Reference: Variáveis que fazem referência a um a uma classe.

Características da Linguagem Java

Tipos de Variáveis Primitivas

- Int: Representa um valor inteiro podendo variar entre -2.147.483.648 e 2.147.483.647, por padrão caso não seja atribuido um valor durante declaração é atribuída o valor 0. Exemplos de declaração:
  - int a;
  - int by1 = -32;
  - int by2 = 0xBB;
- Float: Representa um valor real, também conhecido como ponto flutuante, de precisão simples, ou seja, 32 bits. Por padrão caso não seja atribuido um valor durante declaração é atribuída o valor 0.0. Exemplos de declaração:
  - float a;
  - float by1 = -32.0;
  - float bz2 = 32.2F;
  - float bze = 1.32455e4f;

Características da Linguagem Java

Tipos de Variáveis Primitivas

- Double: É o tipo de dado capaz de armazenar números reais de precisão dupla, ou seja, 64 bits de informação em forma de número real. É usado para representar valores nos quais é preciso uma precisão maior que a de float. Por padrão caso não seja atribuído um valor durante declaração é atribuída o valor 0.0. Exemplos de declaração:
  - double a;
  - double by1 = -32;
  - double by2 = 1.32455e4D;
- Char: É o tipo de dado capaz de armazenar 16 bits representando caractere. Por padrão caso não seja atribuído um valor durante declaração é atribuída o unicode "\0". Exemplos de declaração:
  - char letra = 'A';
  - char letra = '\u0045';

Características da Linguagem Java

Tipos de Variáveis Primitivas

- Boolean: É o tipo de dado que contém literal lógico. Serve para armazenar um único bit de informação. Este bit pode ser representado pelas palavras false (falso) ou true (verdadeiro). Representa estados binários. Por padrão caso não seja atribuido um valor durante declaração é atribuída o false. Exemplos de declaração:
  - boolean a;
  - boolean condicao = false;
  - boolean condicaoNova = true;
- Byte: É o tipo de dado capaz de armazenar 8 bits de informação, ou seja, um número inteiro entre -128 e 127. Por padrão caso não seja atribuído um valor durante declaração é atribuída o valor 0. Exemplos de declaração:
  - byte a;
  - byte num = '1';
  - byte numDois = 2;

Características da Linguagem Java

Tipos de Variáveis Primitivas

- Short: É o tipo de dado que é capaz de armazenar números inteiros de 16 bits, ou seja, um número inteiro entre -32.768 e 32.767. Por padrão caso não seja atribuido um valor durante declaração é atribuido o valor 0. Exemplos de declaração:
  - short a;
  - short num = -32;
  - short = 0xF;
- Long: É o tipo de dado capaz de armazenar 64 bits de informação, ou seja, que pode representar um número inteiro qualquer entre -9.223.372.036.854.775.808L e 9.223.372.036.854.775.807L. Por padrão caso não seja atribuido um valor durante declaração é atribuída o valor 0. Exemplos de decalração:
  - long a;
  - long num = 121L;

Características da Linguagem Java

Variáveis Reference

- Variáveis que fazem referênia a uma classe utilizada para indiciar objetos, exemplo mias comum são as variaveis do tipo String. Por padrão todas as variáveis do tipo Reference recebem como valor a palavra reservada null. Exemplos de declaração:
  - String a;
  - String nome = "João da Silva";
  - String = "Rua 23";
  - classe nomeDaVariavel;

Características da Linguagem Java

Classe

Conceito[Deitel e Deitle, 2016].

Encapsulamento do código e de atribuição de dados para varios elementos do software que contém a mesma estrutura, ou seja atributos e metódos.

[Caelum, 2019].

A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma séries de atributos e comportamentos em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos.

Características da Linguagem Java

Como fazer uma Classe

Sintaxe classes

[Modificado] class Nome_da_Classe{ Atributos e métodos pertencentes a classe};

- A palavra class é reservado no Java devendo ser utilizada apenas para criação de classes;
- As chaves delimitam o bloco que representa as clases, todos os atributos e metódos dessa classe devem ser declarados dentro das chaves;
- Toda classe deve ser seguir o padrão camelcase;

Características da Linguagem Java

Exemplo classe

```
public class Carro {

}
```

Figura: Exemplo de uma classe do tipo Carro

Características da Linguagem Java

Atributos

- Características que definem a classe, ou seja são as própriedades que dizem respeito a classe, de metódo mais direto as variáveis que determinada classe possui;
- Todo atributo deve ser declarado com a primeira letra minuscula, caso se uma palavra composta é escrita tudo junto e e com a primeira letra a partir da segunda palavra em maiúscula. ex.: numeroDaConta;
- Palavra chave TEM;

Exemplo

Conta tem número, conta tem agência, tem titular, data de abertura, saldo e etc, porém conta não tem gênero, logo é possível perceber que os atributos de uma conta seriam número da conta, agência, titular, data de abertura e saldo

Atributos

Exemplo

```
package br.ufop.FabricaDeCarro;

public class Carro {
    private String modelo;
    private int numeroDoModelo;
    byte numeroDePassageiros;


}
```

Figura: Exemplo de uma classe do tipo Carro.

Características da Linguagem Java

Métodos

- Um método em Java é equivalente a uma função, subrotina ou procedimento em outras linguagens de programação;
- Não existe em Java o conceito de métodos globais. Todos os métodos devem sempre ser definidos dentro de uma classe;
- Todo método, salvo exceções, deve começar com letra _ ou $ caso comece com letra a primeira letra minuscula, caso se uma palavra composta é escrita tudo junto e e com a primeira letra a partir da segunda palavra em maiuscula. ex.: numeroDaConta;
- Podem ser considerados como ações ou decisões que uma classe tem que fazer/tomar.

Características da Linguagem Java

Métodos

Sintaxe métodos

A sintaxe para a construção de um método é a seguinte: [modificador] tipo_de_retorno nomeDoMetodo (atributos que o método recebe){ Código }

- Tipos de Retorno:
  - Variáveis Primitivas;
  - Variáveis Reference;
  - Palavra reservada VOID, sem retorno;

Métodos

Exemplo

```
public int getNumeroDoModelo() {
    return numeroDoModelo;
}
```

Figura: Exemplo de um método da classe Carro.

Características da Linguagem Java

Modificadores de Acesso

- Java controla o acesso a atributos e métodos através do uso dos modificadores de acesso;
- Mais segurança para dados, garantido a restrição desses dados a outras classes;
- Tipos de Modificadores:
  - Public;
  - Protect;
  - Default;
  - Private.

Modificadores de Acesso

Tipos de Modificadores de Acesso

- Public: Todos podem acessar aquilo que for definido como public. Classes, atributos, construtores e métodos podem ser public;
- Protected: Pode ser acessado por todas as classes do mesmo pacote e por todas as classes que o estendam, mesmo que essas não estejam no mesmo pacote. Somente atributos, construtores e métodos podem ser protected

Modificadores de Acesso

Tipos de Modificadores de Acesso

- Private: A única classe capaz de acessar os atributos, construtores e métodos privados é a própria classe. Classes, como conhecemos, não podem ser private, mas atributos, construtores e métodos sim
  - Todo atributo em java é declarado como private;
- Default: Todas as classes do mesmo pacote têm acesso ao atributo, construtor, método ou classe.

Modificadores de Acesso

Exemplo

```
public class Carro {
    public String modelo = "carro";
    private int numeroDoModelo;
    private byte numeroDePassageiros;


    @Override
    public String toString() {
        return "modelo=" + modelo + ", numeroDoModelo=" + numeroDoModelo + ",
                + numeroDePassageiros;
    }
}
```

Figura: Classe Carro com modificadores de Acesso.

Getters e Setters

- O modificador private faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão.
  1. Solução:Utilização de métodos para leitura e modificação dos atributos.

Getters e Setters

- O modificador private faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão.
  1. Solução:Utilização de métodos para leitura e modificação dos atributos.
  2. Método Getter para leitura do valor de um atributo;

Getters e Setters

- O modificador private faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão.
  1. Solução:Utilização de métodos para leitura e modificação dos atributos.
  2. Método Getter para leitura do valor de um atributo;
  3. Método setter para modificar o valor de um atributo;

Getters e Setters

- O modificador private faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão.
  1. Solução:Utilização de métodos para leitura e modificação dos atributos.
  2. Método Getter para leitura do valor de um atributo;
  3. Método setter para modificar o valor de um atributo;
  4. A convenção para esses métodos é de colocar a palavra get ou set seguida do nome do atributo.

Getters

- O método do tipo get deve ter como retono o mesmo tipo do atributo que o mesmo vai retor;
- Não pode receber nenhum parâmetro, uma vez que o mesmo apenas deve retornar um valor;
- Seu modificador de acesso deve ser do tipo Public;

Getters

Exemplo

```
public int getNumeroDoModelo() {
    return numeroDoModelo;
}
```

Figura: Getter do atributo numeroDoModelo.

Setters

- O método do tipo set deve ter como retorno void uma vez que o mesmo não deve retornar nenhum valor;
- Deve receber o parâmetro do mesmo tipo ou inferior que o tipo do atributo;
- Seu modificador de acesso deve ser do tipo Public;

Setters

Exemplo

```
public void setNumeroDoModelo(int numeroDoModelo) {
    this.numeroDoModelo = numeroDoModelo;
}
```

Figura: Setter do atributo numeroDoModelo.

Construtores

- Os construtores são os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida;
- Por padrão, o Java já cria esse construtor sem parâmetros para todas as classes;
- A partir do momento que você declara um construtor, o construtor default não é mais fornecido.
- Podem ser definidos quantos construtores forem preciso.

Sintaxe Construtores

[Modificador de Acesso] Nome_da_Classe(Atributos){Bloco}

Construtores

Por que usar?

- Instânciar atributos que devem obrigatória conter um argumento.

Exemplo

Como exemplo, vamos pegar a nossa Classe carro, existe carro sem chassis, modelo e tipo de combustível? Não, logo para evitar erros de atribuição de valores, deve-se fazer um construtor que receba esses argumentos como paramentros e instânciem esses atributos.

Construtores

Exemplo

```
public Carro(String modelo, int numeroDoModelo) {
    super();
    this.modelo = modelo;
    this.numeroDoModelo = numeroDoModelo;
}
```

Figura: Construtor [Caelum, 2019].

Instânciando Objetos

1. Uma classe não é um objeto.

Instânciando Objetos

1. Uma classe não é um objeto.
2. Um objeto é criado a partir do momento que uma classe é instânciada

Instânciando Objetos

1. Uma classe não é um objeto.
2. Um objeto é criado a partir do momento que uma classe é instânciada
3. Palavra reservada "new";

Instânciando Objetos

1. Uma classe não é um objeto.
2. Um objeto é criado a partir do momento que uma classe é instânciada
3. Palavra reservada "new";

Sintaxe

```
Nome_da_Classe nome_da_Variavel = new
Nome_da_Classe(Atributos do construtor);
```

Vetores

1. Vetores são objetos em java.

Vetores

1. Vetores são objetos em java.
2. Obrigatório a passagem do tamanho do vetor;

Vetores

1. Vetores são objetos em java.
2. Obrigatório a passagem do tamanho do vetor;
3. Posições vão de zero a n-1;

Vetores

1. Vetores são objetos em java.
2. Obrigatório a passagem do tamanho do vetor;
3. Posições vão de zero a n-1;
4. Tentativas de escrita ou leituras de posições menores que 0 ou maiores que o tamanho do vetor-1 geraram exceções.

Vetores

1. Vetores são objetos em java.
2. Obrigatório a passagem do tamanho do vetor;
3. Posições vão de zero a n-1;
4. Tentativas de escrita ou leituras de posições menores que 0 ou maiores que o tamanho do vetor-1 geraram exceções.

Sintaxe

```
Tipo[] nome_da_Variavel = new
Tipo[quantidad_de_epaço_a_ser_alocado]
```

Agenda

Exercícios

1. Definas os quatro tipos de modificadores de acesso, especificando a visibilidade de cada um.

Exercícios

1. Definas os quatro tipos de modificadores de acesso, especificando a visibilidade de cada um.
2. Crie uma classe Carro e defina todos os atributos necessários para essa classe e seu consttrutor e seus metódos;

Exercícios

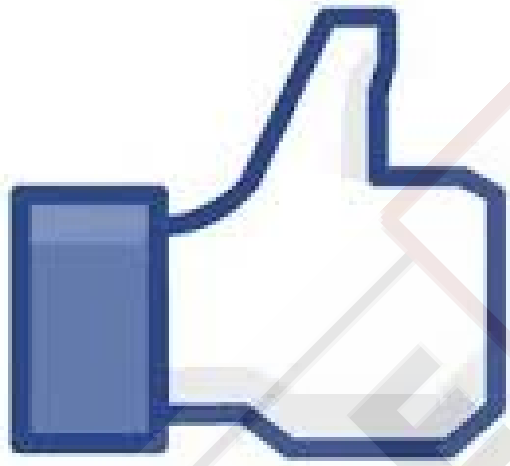
1. Definas os quatro tipos de modificadores de acesso, especificando a visibilidade de cada um.
2. Crie uma classe Carro e defina todos os atributos necessários para essa classe e seu consttrutor e seus metódos;
3. Crie um que receba variáveis do tipo Carro (Classe criada no exercicio anterio) e faca com esses carros sejam impressos.

Exercícios

1. Suponha que vc foi contratado para fazer o sistem de uma conssecionaria de veículos. Quais classes seu projeto teria? Não levar em consideração herança.

Exercícios

1. Suponha que vc foi contratado para fazer o sistem de uma conssecionaria de veículos. Quais classes seu projeto teria? Não levar em consideração herança.
2. Implemente as classes anteriores e monte o sistema da conssecionaria



MUITO OBRIGADO!!!

Referências Bibliográficas I

Booch, G. (1995).
Unified Method for Object-Oriented Development.
Rational Software Corporation.

Caelum (2019).
Java e orientação a objeto.
https://www.caelum.com.br/download/caelum-java-objetos-fj11.pdf.

Deitel, P. e Deitle, H. (2016).
Java: como programar.
Pearson Education do Brasil„ São Paulo :, 10. ed. edition.