Linguagem Orientada a Objetos

Construtores e sobrecarga

Prof. Ms. Leonardo Rocha

- Unidade de Ensino: 2
- Competência da Unidade: Compreensão de como utilizar construtores e fazer sobreposição de métodos.
- Resumo: Atributos e métodos estáticos, construtores, sobrecarga e sobreposição de métodos.
- Palavras-chave: Construstores, sobrecarga, métodos.
- Título da Teleaula: Construtores e sobrecarga
- Teleaula nº: 2

Contextualização

Atributos e métodos
Construtores
Sobrecarda e sobreposição de métodos
Estrutura de decisão, controle e repetição
Operadores e tipos de dados
Reutilização de classes

Atributos e métodos estáticos

Antes de mais nada...

As IDEs são robustas para o desenvolvimento complexo de software. Contam com:

Autocompletar - comandos aparecem na digitação
Realce de sintaxe - Cores no código para facilitar
Indicativos de erros - Erros léxicos e sintáticos
Depuração (debug) - Verificação do funcionamento
Refatoração de código - mudanças de forma automática
Geração de executável - arquivo .jar criado
Testes - Criação de testes automatizados de classes e métodos

Métodos construtores

A declaração de um construtor é obrigatória e deve sempre possuir o mesmo nome da classe. Não existe nenhum tipo de retorno, nem mesmo a palavra void pode ser especificada. É invocado no momento da criação do objeto através do **operador** new e é responsável por criá-lo em memória, ou seja, instanciar a classe que foi definida. O retorno desse **operador**, é uma referência para objeto recém-criado. É possível criar mais de um construtor dentro da mesma classe.

Em java, apenas as Interfaces não possuem construtores.

7

Exemplo

```
public class Carro {
// Construtor da classe Carro

  public Carro(){
        //Construção de objeto
  }

}
```

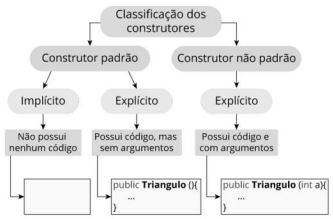
8

Invocando o Construtor

```
public class Carro {
// Construtor da classe Carro
  public Carro(){
        //Construção de objeto
  }
}
public class Venda {
  public static void main(String[] args) {
 // Construtor da classe Carro
Carro Ford = new Carro();
}
}
```

9

Tipos de construtores em Java



10

Sobrecarga de métodos

```
1  public class Matematica {
2      static int mult(int a, int b){
3          return a * b;
4      }
5      static double mult(double a, double b){
6          return a * b;
7      }
8      static double mult(double a, double b, double c){
9          return a * b * c;
10     }
11     public static void main(String[] args) {
12        System.out.println("Mult: " + Matematica.mult(5, 3));
13        System.out.println("Mult: " + Matematica.mult(3.2, 4.1));
14        System.out.println("Mult: " + Matematica.mult(1.4, 2));
15        System.out.println("Mult: " + Matematica.mult(2, 3.5));
16        System.out.println("Mult: " + Matematica.mult(4.5, 5.2,
   2));
17     }
18 }
```

11

Definição

A sobrecarga (overload) consiste num conceito de polimorfismo que é empregado na criação de variações de um mesmo método, ou seja, a criação de dois ou mais métodos com nomes totalmente iguais em uma classe. O que os difere são seus argumentos. Só dessa forma é feita a separação destes.

12

Sobreposição

Sua principal característica é possuir mais uma classe. Nesse sentido, consiste na primeira herdar a segunda.

Na sobreposição é importante fazer o uso da annotation @Override para indicar que o método foi sobreposto.

13

Sobreposição de métodos

```java
public class Geom2D {
    double perimetro;
    double calcPerimetro(){
        return 0;
    }
}

public class Triangulo extends Geom2D {
    ...
    @Override
    double calcPerimetro() {
        super.perimetro = this.ladoA + this.ladoB + this.ladoC;
        return super.perimetro;
    }
    public static void main(String[] args) {
        Triangulo trian = new Triangulo(3, 25, 26);
        System.out.println("Perimetro: " +
trian.calcPerimetro());
    }
}
```

14

Sobrecarga x Sobreposição

Característica	Sobrecarga	Sobreposição
Argumentos	Devem ser trocados	Não devem ser trocados
Tipo de Retorno	Pode ser trocado	Não pode ser trocado
Tipo de Acesso	Pode ser trocado	Pode ser trocado por um modificador menos restritivo
Tipo de Exceção	Pode ser trocado	Pode ser trocado por uma exceção menos restritiva
Classe	Ocorre em uma classe	Ocorre entre duas classes
Herança	Não envolve herança	Envolve herança
Invocação	Ocorre em tempo de compilação	Ocorre em tempo de execução

Estrutras de decisão

16

Definição

Java executa o seu código de forma sequencial e de cima para baixo, linha a linha. Às vezes, é necessário romper com essa execução sequencial, e queremos executar um trecho do código somente se alguma coisa ocorrer. Essas tomadas de decisão rompem com o fluxo linear do código, criando bifurcações que ajudam a tornar o código mais dinâmico.

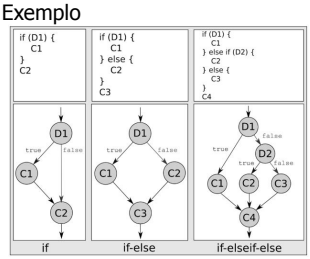
17

IF e IF-ELSE

A principal estrutura de decisão é o comando if e if-else. Esses comandos também podem ser concatenados sucessivamente por outros ifs, como em if-elseif-else, etc.

18

Exemplo



19

Operador ternário

Considerada uma forma simplificada de construção de comandos do tipo if-else. O nome deve-se ao fato de que o comando é quebrado em três partes separadas pelos símbolos de interrogação (?) e de dois pontos (:).

20

Exemplo

Comando: if-else	Comando: operador-ternário

`if (ExpLógicaA) {`
`  ComandoA;`
`} else {`
`  ComandoB;`
`}`
	`ExpLógicaA ? ComandoA : ComandoB;`

21

O código

```
import java.util.*;

/**
 *
 * @author Leonardo Rocha
 */
public class numeroPar {
    public static void main(String[] args){
        Scanner valor = new Scanner(System.in);
        System.out.println("Informe um número inteiro: ");
        int numero = valor.nextInt();

        if(numero % 2 == 0){
            System.out.println("O número digitado é par");
        }
        else{
            System.out.println("O número digitado não é par");
        }
        System.exit(0);
    }
}
```

Estrutura de decisão, controle e repetição

23

Switch-case | Switch-case-default

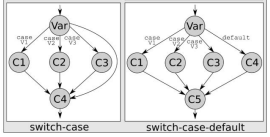
Esse comando provê o que chamamos de estrutura de seleção múltipla baseada em alguma variável de controle.

24

Exemplo

```
switch (Var) {
   case V1:
      C1; break;
   case V2:
      C2; break;
   case V3:
      C3; break;
}
C4
```

```
switch (Var) {
   case V1:
      C1; break;
   case V2:
      C2; break;
   case V3:
      C3; break;
   default:
      C4;
}
C5
```



25

Requisições

Síncrona - Nesse tipo de requisição cada operação deve ser concluída para que uma nova operação possa ser realizada. O navegador precisará recarregar a página após enviar os dados.

Assíncrona - Nesse tipo de requisição não é necessário aguardar o fim de uma operação. Não há necessidade de recarregar a página.

Ex.: À medida que os dados são digitados, o navegador envia as informações para o servidor sem a necessidade de recarregar a página (na prática, não seria necessário um botão de salvar, por exemplo).

26

Operadores aritméticos

Nome Operador	Símbolo	Exemplo em Java	Exemplo Numérico
Soma	+	x + y	5 + 3 (= 8)
Subtração	-	x - y	5 - 3 (= 2)
Multiplicação	*	x * y	5 * 3 (= 15)
Divisão	/	x / y	5 / 3 (= 1)
Resto da divisão	%	x % y	5 % 3 (= 2)

27

Operadores relacionais

Nome Operador	Símbolo	Exemplo em Java	Exemplo Numérico
Igualdade	==	x == y	5 == 3 (falso)
Diferente	!=	x != y	5 != 3 (verdade)
Maior que	>	x > y	5 > 3 (verdade)
Maior ou igual a	>=	x >= y	5 >= 3 (verdade)
Menor que	<	x < y	5 < 3 (falso)
Menor ou igual a	<=	x <= y	5 <= 3 (falso)

28

Operadores de atribuição

Nome do Operador	Símbolo	Exemplo Encurtado	Significado Exemplo
Atribuição de adição	+=	x += 3	x = x + 3
Atribuição de subtração	-=	x -= 3	x = x - 3
Atribuição de multiplicação	*=	x *= 3	x = x * 3
Atribuição de divisão	/=	x /= 3	x = x / 3
Atribuição de resto	%=	x %= 3	x = x % 3

29

Operadores de incremento e decremento

Nome Operador	Símbolo	Exemplo em Java	Explicação do Exemplo
Pré-incremento	++	++x	Incrementa x em 1 e, então, utiliza x na expressão atual.
Pós-incremento	++	x++	Utiliza x na expressão atual e, então, incrementa x em 1.
Pré-decremento	--	--x	Decrementa x em 1 e, então, utiliza x na expressão atual.
Pós-decremento	--	x--	Utiliza x na expressão atual e, então, decrementa x em 1.

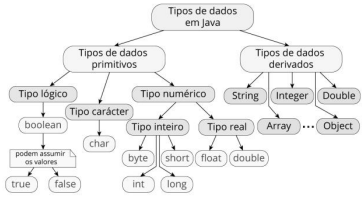
30

Operadores lógicos

Nome Operador	Símbolo	Exemplo em Java	Exemplo com Valores
E Lógico	&&	exp1 && exp2	true && false (falso)
Ou Lógico	||	exp1 || exp2	true || false (verdade)
Negação Lógico	!	!exp	!true (falso)

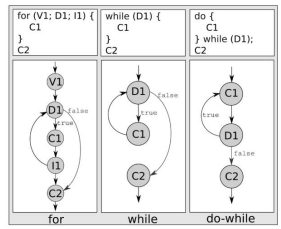
31

Tipos de dados



32

Repetição com while / for / do while



33

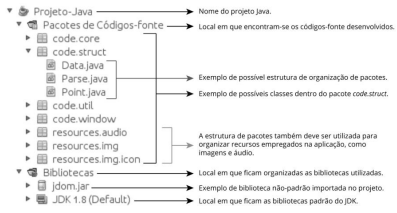
Break e Continue

O programador pode querer interromper todo o fluxo de um laço de repetição a qualquer momento ou, em vez disso, desejar interromper parte de um trecho de código dentro de um loop e ir diretamente para a estrutura de decisão; para tanto, a palavra reservada break e continue deverão ser utilizadas, respectivamente.

Reutilização de classes

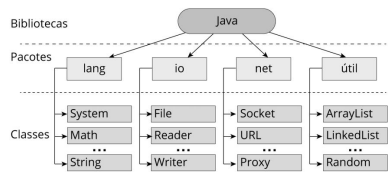
35

Estrutura



36

Principais pacotes nativos Java



37

Exemplo de importação

Exemplo de Código	Descrição sobre as Importações
`import java.io.File;`	Importação da classe File que está dentro pacote *java.io.*
`import java.net.Socket;`	Importação da classe Socket do pacote *java.net.*
`import java.util.Random;`	Importação da classe Random do pacote *java.util.*
`import java.lang.System;`	Importação da classe System do pacote *java.lang.* **OBS:** esta importação é a única opcional, pois é feita de forma automática.
`import java.io.*;`	Importação de todas as classes do pacote *java.io.*

38

Exemplo de importação

```
import java.util.*;

/**
 *
 * @author Leonardo Rocha
 */
public class numeroPar {
    public static void main(String[] args){
        Scanner valor = new Scanner(System.in);
        System.out.println("Informe um número inteiro: ");
        int numero = valor.nextInt();

        if(numero % 2 == 0){
            System.out.println("O número digitado é par");
        }
        else{
            System.out.println("O número digitado não é par");
        }
        System.exit(0);
    }
}
```

39

Modificadores de acesso

Por meio deles, podemos restringir ou permitir que um atributo, método, construtor e/ou classe seja acessado ou não.

Modificador	Palavra-Reservada	Descrição da Visibilidade
Público	*public*	Aplicável à própria classe e a qualquer outra.
Protegido	*protected*	Aplicável à própria classe, outras classes dentro do próprio pacote e dentro de subclasses em outros pacotes.
Default	-	Aplicável à própria classe e a outras classes dentro do mesmo pacote.
Privado	*private*	Aplicável à própria classe somente.

40

Encapsulamento

Como exemplo, assumimos como regra que um objeto nunca deve manipular diretamente os atributos de outro objeto.

Em muitos casos, a manipulação direta pode ocasionar erros e gerar muitas inconsistências no programa desenvolvido. Dessa forma, devemos bloquear os

atributos utilizando os modificadores de acesso private, default ou, ainda, protected, porém, após o bloqueio dos atributos, como acessá-los e modificá-los?

41

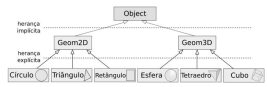
Métodos getters e setters

São conhecidos como gets e sests. Os métodos gets servem para pegar alguma informação, já os métodos sets servem para definir alguma informação.

```
1  public class Pessoa {
2      private int idade;
3      public int getIdade() {
4          return idade;
5      }
6      public void setIdade(int idade) {
7          this.idade = idade;
8      }
9  }
```

42

Herança



A operação de herança envolve duas classes, em que uma das classes é chamada de subclasse e a outra é chamada de superclasse.

Nessa operação, a subclasse herda todas as características definidas na superclasse.

```
1  class SuperClasse {
2      ...  //código associado a superclasse
3  }
4  class SubClasse extends SuperClasse {  //operação de herança
5      ...  //código associado a subclasse
6  }
```

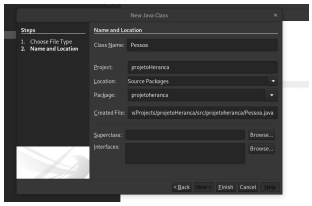
43

Polimorfismo

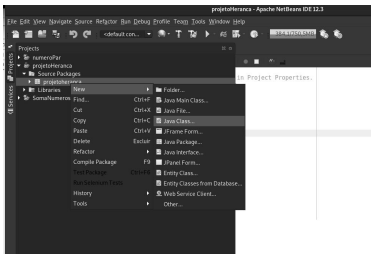
Diz respeito à capacidade da linguagem de programação de processar objetos de formas diferentes dependendo do seu tipo de dado ou classe. A vantagem é ter métodos com o mesmo nome mas que são implementados de formas diferentes. Exemplo: Classe conexão com método abrirConexao() será o mesmo para as classes filhas ConexãoOracle e ConexãoPostgree mas a implementação será diferente.

44

Criar projeto com Apache Ant - Herança



45



46

```java
public class Pessoa {

    String sexo;
    String corPele;
    int  peso;

    //Cria método 1
    public void comer()
    {
        System.out.println("Estou com fome!");
    }

    //Cria método 2
    public void sono()
    {
        System.out.println("Estou com sono!");
    }

    //Cria método estudar
    public void estudar()
    {
        System.out.println("O/a estudante vai estudar");
    }

    //Exemplo de método privado. Este não pode ser herdado
    private void correr()
    {
        System.out.println("Necessário correr para exercitar");
    }

}
```

Recapitulando

48

- Atributos e métodos
- Construtores
- Sobrecarda e sobreposição de métodos
- Estrutura de decisão, controle e repetição
- Operadores e tipos de dados
- Reutilização de classes