



Prof. Dr. Fernando Almeida
proffernando.almeida@fiap.com.br

DDD (Domain Driven Design)

Java SE e Java EE

O QUE VAMOS APRENDER HOJE?

Introdução à Orientação a Objetos

1

Programação Orientada à Objetos

2

Classes e objetos

3

Atributos e métodos

4

Casting

5

Tipos de dados

6

Entrada de dados (console)

Programação Orientada a Objetos

Linguagem Java

|
+
...do caos aos objetos

Linha do tempo (*algumas* técnicas de programação)

1950 - 1960 <i>Era do caos</i>	1970 - 1980 <i>Era da estruturação</i>	1990 até agora <i>Era dos objetos</i>
saltos, goto, variáveis não estruturadas, variáveis espalhadas ao longo do programa	if-then-else, blocos, registros, laços-while	objetos, mensagens, métodos, herança...

Programação Orientada a Objetos - POO

- **Fatores importantes:** entendimento do código, fácil manutenção, reaproveitamento de código, tempo e agilidade no desenvolvimento de um sistema...
- Itens importantes: **Classes, Objetos, Atributos, Métodos, Construtores...**

Classes

Orientação à Objetos

“É um **paradigma de análise, projeto e programação** de sistemas de software baseado na *composição* e *interações* entre diversas unidades de software de software chamadas de **objetos**

Paradigma: um padrão, um modelo, um exemplo...

Pilares da Programação Orientada a Objetos - POO



Programação Procedural

“Também conhecida como **Programação Estruturada**. Voltada aos *passos* que um programa deve executar para atingir o estado desejado.

Exemplo de linguagem: Pascal, C...

Orientação à Objetos

“Com a Orientação à Objetos fica fácil *organizar* e *escrever menos.* através da *centralização* das responsabilidades e *encapsulando* a lógica do programa.



Como começar um programa em **Java**?

```
1 class MinhaClasse {  
2     //variáveis, métodos  
3 }  
4  
5 }
```

- **Classe** - abstração do mundo real
- **Estado** (variáveis e atributos) - define as características
- **Ações** (métodos) - define o comportamento da classe

```
1 class Aluno {  
2  
3     String nome;  
4     String matricula;  
5  
6     public void realizarMatricula() {}  
7  
8 }
```

Exemplo: criação de um cadastro de alunos

Meu primeiro programa em Java

Hello World!

HelloWorld.java

- O nome do arquivo deve ser o mesmo nome da classe pública e ter a extensão `.java`
- Exemplo: `HelloApp.java`

```
1 public class HelloApp {  
2     public static void main(String args[]) {  
3         System.out.println("Hello World!");  
4     }  
5 }
```

- Compilando e executando em linha de comando

```
1 cd c:/java          // entra no diretório c:/java  
2 javac HelloApp.java // compila a classe HelloApp.java e gera o arquivo HelloApp.class  
3 java HelloApp      // executa o arquivo HelloApp.class e imprime na tela: "Hello World"
```

Método main ()

Quando uma classe é executada, a máquina virtual procura pelo método `main()`. Este método é o ponto de partida de todas as aplicações Java, e deve ser assinado desta forma:

```
1 | public static void main(String args[]) {}
```

• • • • + • • Cáracterísticas das classes • + + •

- Toda classe possui um **nome**
- Possuem visibilidade (***public*, *private* e *protected***)
- Possuem membros como: **características e ações**

```
1 | public class Teste{  
2 | //ATRIBUTOS OU PROPRIEDADES  
3 | //MÉTODOS  
4 | }
```

Exemplo

```
1 public class Caes {  
2  
3     public String nome;  
4     public int peso;  
5     public String corOlhos;  
6     public int quantPatas;  
7  
8     public void falar(){  
9         //MÉTODO FALAR  
10    }  
11  
12    public void andar(){  
13        //MÉTODO ANDAR  
14    }  
15  
16    public void comer(){  
17        //MÉTODO COMER  
18    }  
19  
20    public void dormir(){  
21        //MÉTODO DORMIR  
22    }  
23 }
```

Demonstração da classe Cães

Classe Cães

Objetos cachorros



Anda
Fala
Come
Dorme
PegaOsso



Anda
Fala
Come
Dorme
PegaOsso



Anda
Fala
Come
Dorme
PegaOsso

a classe é sempre um molde/projeto para o objeto cachorro:

Objetos

Objetos

“Os **objetos** são características definidas pelas classes. Neles é permitido instanciar objetos da classe para inicializar os atributos e invocar métodos

- **o objeto** representa algo que existe, concreto
- **a classe** é considerada como um modelo ou projeto de um objeto

Diferença entre
objeto e classe



= **Objeto**



= **Classe**

o **objeto** é a materialização
da classe

CLASSES X OBJETOS

OBJETOS

Projeto



CLASSE



Instância



Instância



Instância

Atributos

Atributos

“Os **atributos** são propriedades de um objeto, também conhecidos como **variáveis** ou **campos**

- Definem o **estado** de um objeto (podendo sofrer alterações)

Atributos

Classe Cachorro

```
1 | public class Cachorro{  
2 |  
3 |     public String nome;  
4 |     public int peso;  
5 |     public String corOlhos;  
6 |     public int quantPatas;  
7 | }
```

Atributos

Classe Teste Cachorro

```
• • 1 public class TestaCaes {  
• • 2  
• + 3     public static void main(String[] args) {  
+ • 4         Cachorro cachorro1 = new Cachorro();  
• • 5         cachorro1.nome = "Pluto";  
• • 6         cachorro1.corOlhos = "azuis";  
• • 7         cachorro1.peso = 53;  
• • 8         cachorro1.quantPatas = 4;  
• • 9  
| 10        Cachorro cachorro2 = new Cachorro();  
+ 11        cachorro2.nome = "Rex";  
12        cachorro2.corOlhos = "amarelo";  
13        cachorro2.peso = 22;  
14        cachorro2.quantPatas = 3;  
15  
16        Cachorro cachorro3 = new Cachorro();  
17        cachorro3.nome = "Bob";  
18        cachorro3.corOlhos = "marrom";  
19        cachorro3.peso = 13;  
20        cachorro3.quantPatas = 4;  
21  
22    }  
23  
24 }
```

Métodos

Métodos

“Os métodos são **ações** ou **procedimentos**, onde podem interagir e se comunicarem com outros objetos

- A execução dessas ações se dá através de mensagens
- Envio de uma solicitação ao objeto para que seja efetuada a rotina desejada
- Boas práticas: sempre usar nomes como verbos: `voltar`, `corner`; `avançar`, `pesquisarNomes`, `resgatarValor`...

Métodos

```
1 class Cachorro{  
2     int tamanho;  
3     String nome;  
4  
5  
6     void latir(){  
7         if(tamanho > 60)  
8             System.out.println("Wooof, Wooof!");  
9         else if(tamanho > 14)  
10            System.out.println("Ruff!, Ruff!");  
11        else  
12            System.out.println("Yip!, Yip!");  
13    }  
14 }
```

Classe Cachorro com método

Classe Testadora

```
1 public class Testa_Cachorro {  
2  
3     public static void main(String[] args) {  
4  
5         Cachorro bob = new Cachorro();  
6         bob.tamanho = 70;  
7         Cachorro rex = new Cachorro();  
8         rex.tamanho = 8;  
9         Cachorro scooby = new Cachorro();  
10        scooby.tamanho = 35;  
11  
12        bob.latir();  
13        rex.latir();  
14        scooby.latir();  
15  
16    }  
17 }
```

Criando um Tipo conta bancária

Criando um tipo: Conta

Foco: *Conta bancária*

Quais informações uma conta bancária possui?

Criando um tipo: Conta

Conta bancária

- número da conta
- nome do titular da conta
- saldo da conta
- limite da conta

Criando um tipo: Conta

O que uma *conta bancária* faz?

Quais *ações* podemos fazer em uma conta bancária?

Criando um tipo: Conta

Em uma conta podemos realizar as seguintes **ações**:

- **sacar** um valor
- **depositar** um valor
- **consultar o nome** do titular
- **consultar o saldo** atual da conta
- **transferir** um valor para outra conta
- **saber o tipo** da conta

Criando um tipo: Conta

Uma conta bancária tem
(características):

- número da conta
- nome do titular da conta
- saldo
- limite disponível

Uma conta bancária faz *(ações)*:

- consultar saldo
- sacar um valor
- depositar um valor
- consultar nome do titular
- transferir um valor para outra conta
- consultar o tipo da conta

Projeto da Classe Conta

Uma conta bancária tem (*características*):

- número
- titular
- saldo
- limite

Uma conta bancária faz (*ações*):

- consultar saldo
- sacar
- depositar
- consultar titular
- transferir um valor
- consultar tipo

Projeto da Classe Conta

Uma conta bancária tem
(características):

- número
- titular
- saldo
- limite

Uma conta bancária faz
(ações):

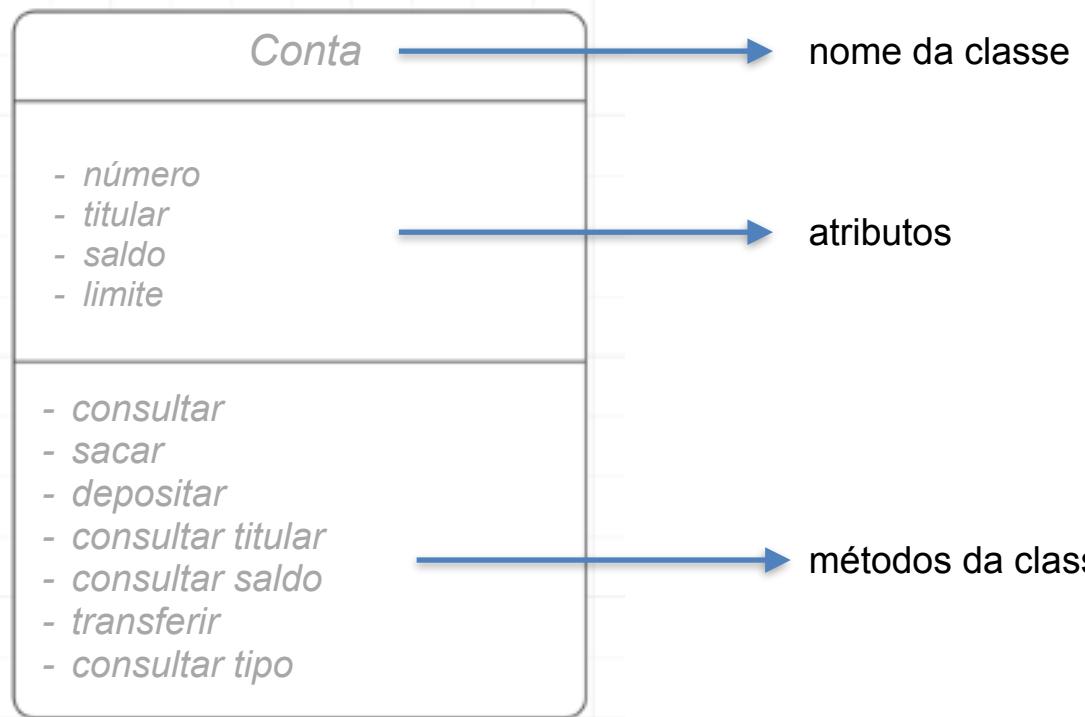
- consultar saldo
- sacar
- depositar
- consultar titular
- transferir um valor
- consultar tipo

Conta
- número - titular - saldo - limite
- consultar - sacar - depositar - consultar titular - consultar saldo - transferir - consultar tipo

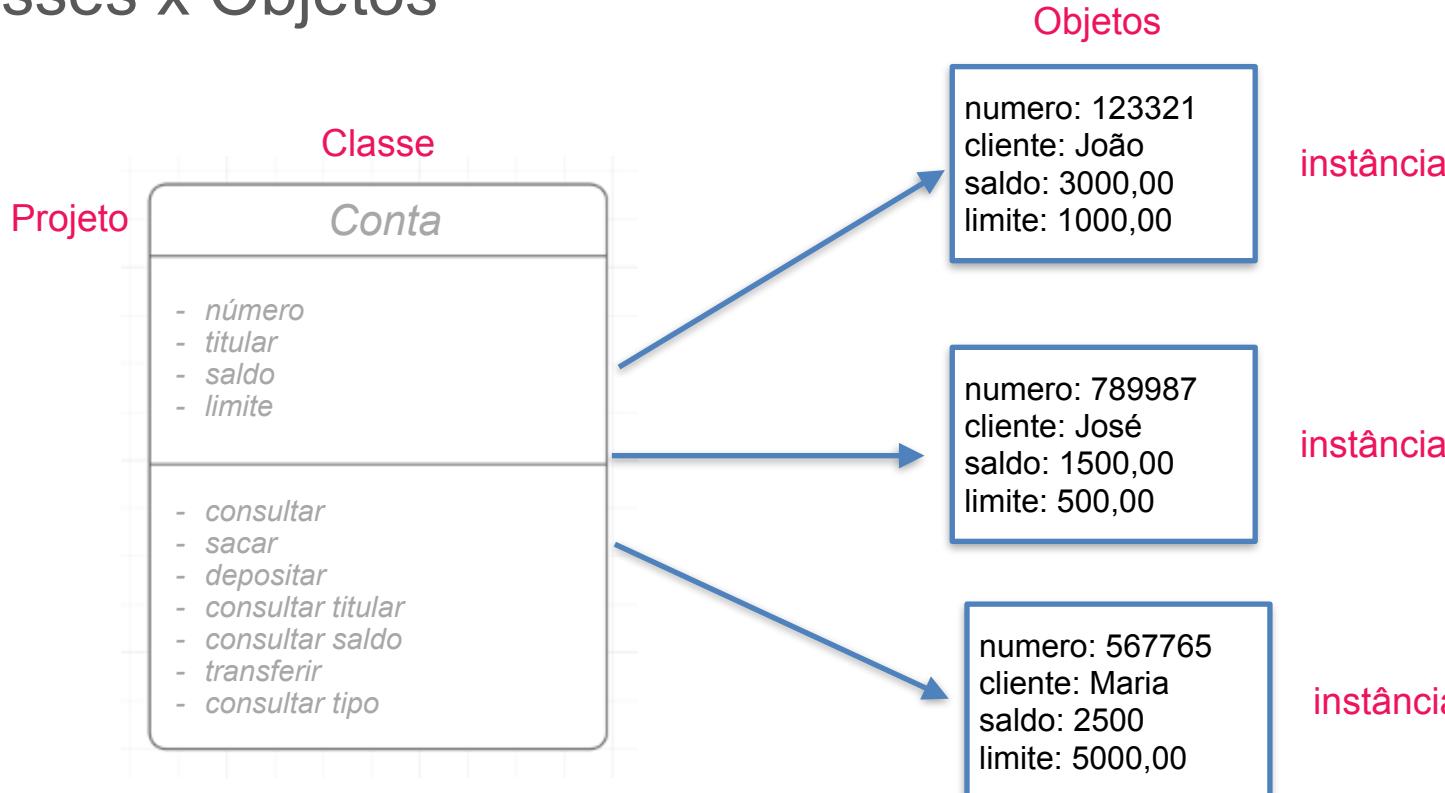
...a importância do *planejar!*

Nossa primeira classe *Conta*

Diagrama de Classe: UML



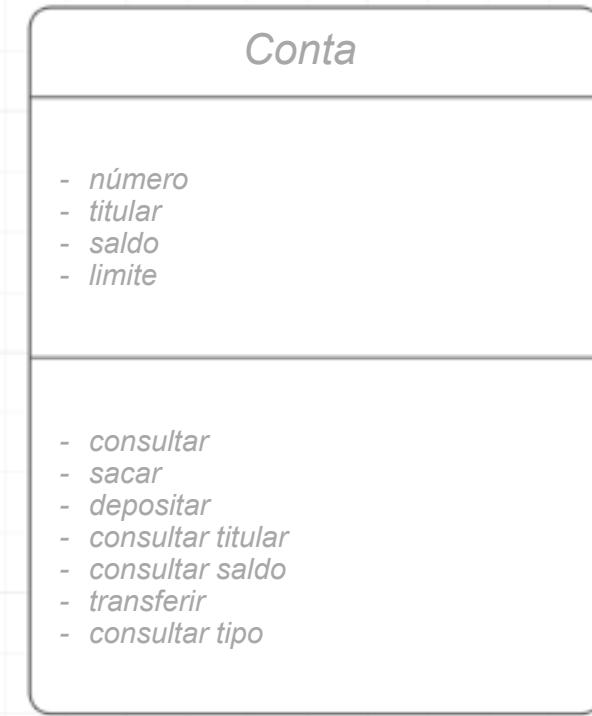
Classe x Objetos



Criando a classe *Conta* em Java

Conta.java

Classe Conta



```
class Conta {
    int numero;
    String titular;
    double saldo;
    double limite;

    //métodos...
}
```

Vamos *praticar!*

especificação da classe Conta.java

Instânciar um Objeto Conta

```
class Programa {  
    public static void main(String[] args) {  
        Conta minhaConta;  
        minhaConta = new Conta();  
    }  
}
```

Instanciar e Popular um Objeto Conta

```
class Programa {  
    public static void main(String[] args) {  
        Conta minhaConta;  
        minhaConta = new Conta();  
        minhaConta.titular = "José da Silva";  
        minhaConta.saldo = 5000.0;  
        System.out.println("Saldo atual: " + minhaConta.saldo);  
    }  
}
```

vamos *praticar!*

Tipos de Dados em Java

• Tipos de dados

- Tipos primitivos:

Tipo	Descrição
<code>boolean</code>	Pode assumir os valores <code>true</code> ou <code>false</code>
<code>char</code>	Representa um caractere Unicode de 16 bits para armazenar dados alfanuméricos
<code>byte</code>	Inteiro de 8 bits. Pode assumir valores entre -2^7 e 2^7-1 (de -128 a 127)
<code>short</code>	Inteiro de 16 bits. Pode assumir valores entre -2^{15} e $2^{15}-1$ (de -32.768 a 32.767)
<code>int</code>	Inteiro de 32 bits. Pode assumir valores entre -2^{31} e $2^{31}-1$ (de -2.147.483.648 a 2.147.483.647)
<code>long</code>	Inteiro de 64 bits. Pode assumir valores entre -2^{63} e $2^{63}-1$
<code>float</code>	Número de ponto flutuante de 32 bits. Pode assumir valores entre $\pm 1.40129846432481707e-45$ e $\pm 3.40282346638528860e+38$
<code>double</code>	Número de ponto flutuante de 64 bits. Pode assumir valores entre $\pm 4.94065645841246544e-324$ e $\pm 1.79769313486231570e+308$

- String:

Tipo	Descrição
<code>String</code>	Cadeia de caracteres que usam 2 bytes por caractere. Strings podem ser vazias (zero caractere) e conter qualquer tipo de caractere.

• Declaração de variáveis

- + .
- + •

tipo do atributo

|
+ **int pesoVeiculo;**

identificador
(nome do atributo)

Exemplos:

float precoProduto;
int idadeAluno;
char conceito;
String nome_Aluno;
boolean maiorDade;

Exemplo de identificadores

- O identificador (nome da variável) é formado por caracteres Unicode. Eles podem ser
- formados por letras, cifrão (\$), *underline*(_) e números (**não pode ser iniciado com número**)

```
1 public class Variaveis {  
2  
3     public static void main(String args[]){  
4         String nome;      // válido  
5         int $idade;     // válido  
6         double 1preco;   // inválido  
7         double preco1;   // válido  
8         int ____$;      // válido  
9         String :nome;    // inválido  
10    }  
11  
12 }
```

Atribuição

pesoVeiculo = 1500

identificador
(nome do atributo)

valor do atributo

Exemplos:

```
float precoProduto = 3.5;  
int idadeAluno = 18;  
char conceito = 'A';  
String nome_Aluno = "João";  
boolean maiorIdade = false;
```

Exemplo de atribuição

• + ..

- Uma vez declarada, a variável deve ser inicializado, e após isso, ser modificada e utilizada.
Elas podem ser declaradas e inicializadas em uma mesma linha de código.

```
| 1 | String nome;           // declara uma variável do tipo String
+ 2 | nome = "Frederico Maia"; // inicializa com um valor
| 3 | int idade = 21;        // declara e inicializa na mesma linha
| 4 | System.out.println(nome+" "+idade); // imprime o valor das variáveis na tela
```

Operadores Aritméticos

- + •
- + •

media = somaNotas / 2;

operator
+
Expressão

Exemplos:

Operação	Operador	Expressão Algébrica	Expressão Java
Adição	+	X + 1	X + 1
Subtração	-	Y - 2	Y - 2
Multiplicação	*	K . X	K * X
Divisão	/	C / 2	C / 2
Resto	%	X mod Y	X % Y

Precedência de Operadores Aritméticos

Operador

Operação

Expressão Algébrica

* / %

Multiplicação, Divisão
e Resto

É o primeiro a ser avaliado. A ordem de avaliação é da esquerda para a direita.

- +

Subtração e soma

É avaliado posteriormente. A ordem de avaliação também é da esquerda para a direita.

Operadores Relacionais

operador
(media >= 6)
Respostas
true ou **false**

Exemplos:

Operação	Operador Matemático	Operador Java	Exemplo	Significado
Igual	=	==	X == Y	X é igual a Y
Diferente	≠	!=	X != Y	X é diferente de Y
Maior	>	>	X > Y	X é Maior que Y
Menor	<	<	X < Y	X é menor que Y
Maior ou Igual	≥	>=	X >= Y	X é maior ou igual a Y
Menor ou Igual	≤	<=	X <= Y	X é menor ou igual a Y

Permite saber a relação existente entre seus dois operandos

Operadores Lógicos

• + •
+ •

resulta em true

$$(2 > 1) \text{ || } (3 < 7)$$

resulta em false

$$(3 > 2) \text{ && } (2 == 2)$$

resulta em true

$$(5 != 0) \text{ || } (1 < 2)$$

resulta em false

!true

Exemplos:

Operação	Operador Matemático	Operador Java	Exemplo
OU	\vee	<code> </code>	<code>(notaEnem > 6) (notaRedacao == 10)</code>
E	\wedge	<code>&&</code>	<code>(mediaFinal >= 6) && (totalFaltas < 25%)</code>
Negação	\sim	<code>!</code>	<code>!pendenciaDocumento</code>

conectam duas ou mais expressões
relacionais



Convenções de Código Java

Java é Case Sensitive!

significa que Java diferencia letras maiúsculas e minúsculas

Convenções de Código em Java

- “80% do tempo e dos custos gastos com software estão relacionados com atividades de manutenção” (*Sun Microsystem*)
- Boas práticas de programação
- Possibilidade de desenvolver códigos mais legíveis
- Maior qualidade
- Melhor entendimento do código
- Aplicáveis em todo desenvolvimento: classes, interfaces, métodos variáveis e constantes...

Comentários no código única linha, várias linhas, documentação

Comentários em Java

- São textos ignorados pelo compilador, mas que podem ser útil para nós humanos
- Utilizados para explicar funcionalidade, melhorar a compreensão do que foi implementado

```
1 | // texto ignorado pelo compilador
```

Comentários de apenas uma linha (//)

```
1 | /* texto que pode conter  
2 |     várias linhas */
```

Comentários de várias linhas (/* */)

```
1 | /** comentários de documentação,  
2 |     começam com barra e dois asteriscos */
```

Comentários de documentação (/** */)

Conversões de Variáveis implícitas ou explícitas

Conversões de variáveis

Conversões

IMPLICITAS

Nenhuma sintaxe especial é necessária porque a conversão é de tipo seguro e nenhum dado será perdido. Exemplo:

- byte para int ou long para float → byte b = 10; int i = b;

Conversões

EXPLICITAS

As conversões explícitas exigem um operador cast. A conversão é necessária quando as informações podem ser perdidas na conversão ou quando a conversão pode não funcionar por outros motivo. Exemplo:

- Float para long ou int para byte → int i = 10; byte b = (byte)i;

Conversões

OUTRAS

O restante dos tipos de conversão disponíveis ou criados no programa.

Exemplo:

- String para int → String s = "123" ; int i = Integer.parseInt(s);

Conversões de variáveis

PARA:	byte	short	char	int	long	float	double
DE:							
byte	----	<i>Impl.</i>	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
short	(byte)	----	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
char	(byte)	(short)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
int	(byte)	(short)	(char)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
long	(byte)	(short)	(char)	(int)	----	<i>Impl.</i>	<i>Impl.</i>
float	(byte)	(short)	(char)	(int)	(long)	----	<i>Impl.</i>
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

Entrada de Dados (console)

Classe Scanner

Recebendo dados do teclado

- A forma mais comum de receber dados informados pelo usuário é através do teclado
- Java possui em sua API uma classe para nos auxiliar - **Scanner**
- A classe Scanner possui métodos (funcionalidades) que permitem ao usuário informar valores do tipo int, long, float, double e String
- A classe Scanner pode ser utilizada para entrada de dados através de um arquivo de texto ou através do teclado

Entrada de dados (console) em JAVA

- Captura a entrada de dados via console
- Possui diversos métodos para facilitar a leitura de dados pelo teclado

```
1 package Aula1;  
2  
3 import java.util.Scanner;  
4  
5 /**  
6 *  
7 * @author Fernando  
8 */  
9 public class IntroducaoScanner {  
10     public static void main(String[] args) {  
11         Scanner input = new Scanner(System.in);  
12         String nome;  
13         System.out.println("Nome: ");  
14         nome = input.next();  
15         System.out.println("Olá, " + nome);  
16     }  
17 }
```

Métodos e tipos de retorno da classe Scanner

Método	Tipo de Retorno
<code>int nextInt()</code>	Retorna o valor lido como um int. Se o valor não for um inteiro, ou estiver fora de sua faixa, lança uma exceção.
<code>long nextLong()</code>	Retorna o valor lido como um long. Se o valor não for um long, ou estiver fora de sua faixa, lança uma exceção.
<code>float nextFloat()</code>	Retorna o valor lido como um float. Se não for um float ou estiver fora de sua faixa, lança uma exceção.
<code>double nextDouble()</code>	Retorna o valor lido como um double. Se não for um double ou estiver fora de sua faixa, lança uma exceção.
<code>String next()</code>	Retorna o valor lido como uma String . A função termina ao encontrar um espaço em branco. Caso entre com um valor como “Sou Java”, apenas a palavra “Sou” será capturada pelo método e retornada.
<code>String nextLine()</code>	Retorna a String informada, mesmo com espaços. Neste caso, se entrar com um valor como “Sou Java”, este mesmo texto será retornado.
<code>void close()</code>	Fechá o Scanner.



```
• • • • .  
• • • . . . .  
• + .  
+ .  
  
1 import java.util.Scanner;  
2  
3 public class Media {  
4  
5     public static void main(String[] args) {  
6         String nome = "";  
7         double nota1 = 0.0;  
8         double nota2 = 0.0;  
9         double nota3 = 0.0;  
10        double media = 0.0;  
11  
12        Scanner entrada = new Scanner(System.in);  
13  
14        System.out.println("Informe seu nome: ");  
15        nome = entrada.nextLine();  
16        System.out.println("Informe o valor da nota 1: ");  
17        nota1 = entrada.nextInt();  
18        System.out.println("Informe o valor da nota 2: ");  
19        nota2 = entrada.nextInt();  
20        System.out.println("Informe o valor da nota 3: ");  
21        nota3 = entrada.nextInt();  
22  
23        media = (nota1+nota2+nota3)/3;  
24  
25        System.out.println("Olá "+nome+" sua média é: "+media);  
26    }  
27  
28 }
```

Vamos
praticar!

Métodos da classe Scanner

Método	Finalidade
next()	Aguarda uma entrada em formato String
nextInt()	Aguarda uma entrada em formato Inteiro
nextByte()	Aguarda uma entrada em formato Inteiro
nextLong()	Aguarda uma entrada em formato Inteiro Longo
nextFloat()	Aguarda uma entrada em formato Número Fracionário
nextDouble()	Aguarda uma entrada em formato Número Fracionário



```
package Aula1;  
• • • .  
• • • .  
• + . .  
+ •  
  
    import java.util.Scanner;  
    /**  
     * @author Fernando  
     */  
    public class Exemplo1ClasseScanner {  
        public static void main(String[] args) {  
  
            Scanner input = new Scanner(System.in);  
            int valorA, valorB, valorC;  
            double media;  
  
            System.out.println("Valor A: ");  
            valorA = input.nextInt();  
            System.out.println("Valor B: ");  
            valorB = input.nextInt();  
            System.out.println("Valor C: ");  
            valorC = input.nextInt();  
  
            media = (valorA + valorB + valorC) / 3;  
  
            System.out.println("Média: " + media);  
        }  
    }
```

Atividades

Entrada, Processamento e Saída

Atividades

- 1) Escreva um programa em Java para calcular o pagamento de comissão para vendedores de peças automotivas. Considerando que para cada peça vendida, o vendedor terá 5% de comissão. O programa deve ter os seguintes dados:
 - 1) Identificação do vendedor
 - 2) Código da peça
 - 3) Preço unitário da peça
 - 4) Quantidade vendida



Atividades

2) Na empresa onde trabalhamos há uma tabela com o quanto foi gasto em cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total, sabendo que em Janeiro, foram gastos R\$ 15000, em Fevereiro, R\$ 23000 e em Março, R\$ 17000. Faça um programa que calcule e imprima o gasto total no trimestre. Siga os passos abaixo:

- 1) Crie uma classe chamada `BalancoTrimestral` com um bloco main (exemplo anterior)
- 2) Dentro do main, declare uma variável inteira chamada `gastosJaneiro` e inicialize-a com R\$ 15000
- 3) Crie também as variáveis `gastosFevereiro` e `gastosMarco`, inicializando-as com R\$ 23000 e R\$ 17000, respectivamente (utilize uma linha para cada declaração)
- 4) Crie uma variável chamada `gastosTrimestre` e inicialize-a com a soma das outras 3 variáveis:
`int gastosTrimestre = gastosJaneiro + gastosFevereiro + gastosMarco`
- 5) Imprima a variável `gastosTrimestre`
- 6) Adicione um código (sem alterar as linhas que já existem) para imprimir a média mensal de gastos, criando uma variável `mediaMensal` junto com uma mensagem. Para isso, concatene a String com valor utilizando a expressão “**Valor da média mensal = “ + media mensal**



Obrigado e até a próxima aula!



Copyright © 2022 | Professor Fernando Luiz de Almeida

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.