Técnicas de Programação

Aula 11 – Programação Estruturada em Java

Objetivos de Aprendizagem

- Descrever o paradigma de programação estruturado.
- Criar algoritmos com estruturas de decisão, de repetição e vetores.

Comentários

- Os comentários são para documentar e melhorar a legibilidade dos programas.
- O compilador Java ignora os comentários fazendo com que eles não apareçam no programa executável.
- A linguagem Java tem três formas de identificar comentários.

Comentário de uma única linha

- Esse comentário inicia com `//`, ele é chamado de comentário de uma única linha, porque termina no final da linha atual.
- Observação:
  - O comentário de uma única linha pode iniciar no meio de uma linha e continuar até o final dessa linha.

Comentário de múltiplas linhas

- É um comentário que pode se estender por várias linhas. Este comentário inicia com o delimitador `/*` e termina com o delimitador `*/`.
- Observações:
  - Esquecer um dos delimitadores causa um erro de sintaxe.
  - Os comentários `/*` e `*/` não podem ser aninhados em Java, ou seja, não se pode desativar um trecho de código, pois o código que você quer desativar poderia conter um delimitador.

Comentário de documentação

- É um comentário especial para Java, que devem ser inseridos imediatamente antes de uma declaração de classe, variável ou métodos
- Indicam que o comentário deve ser incluído na documentação do programa.
- A documentação do programa é gerada pelo programa utilitário javadoc (fornecido pela Sun Microsystems com o Java 2 Software Development Kit)
- Esse programa lê esses comentários do seu programa e gera automaticamente a documentação do programa.

Exemplo de comentários

```
/**
* Descrição: Programa que demonstra o uso de comentários
* @author Prof. MSc. Renan Costa Alencar
* AGOSTO 2019
*/
public class MeuPrimeiroPrograma {
/**
* Método main da aplicação
* @param args parâmetros da linha de comando
*/
public static void main( String[] args ){
    System.out.println( "Não usamos 'Hello World!" );
} // Fim do método main
} /* Fim da classe MeuPrimeiroPrograma */
```

Identificadores

- É uma série de caracteres, alfanumérico, que consistem em letras, dígitos, sublinhados “_” e sinais de cifrão “$”, eles não podem iniciar com digito;
- Não podem conter nenhum espaço e também não podem ser palavras reservadas da linguagem Java.
- Observação:
  - Evite utilizar identificadores que contém “$”, porque estes são utilizados pelo compilador para criar nomes de identificadores.

Palavras reservadas

- As palavras reservadas são reservadas pela linguagem Java para implementar vários recursos, como as estruturas de controle.
- Elas não podem ser utilizadas como identificadores.

Palavras reservadas

abstract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
false	final	finally	float	for
goto	if	implements	import	instanceof
int	interface	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
true	try	void	volatile	while

Variáveis e constantes

- A declaração de variáveis e constantes em Java é feita colocando o identificador do tipo seguido do identificador da variável.

```
<tipo> <identificador>
```

- As regras para os nomes das variáveis e constantes em Java são as seguintes: precisa começar com uma letra e ser uma sequência de letras ou dígitos.

Variáveis e constantes

- Podem ser declaradas em qualquer lugar dentro de uma classe ou de um método.
- Em Java uma constante é definida com o uso da palavra reservada final.
- A palavra final indica que o valor atribuído fica definido para sempre, ou seja, não pode ser mais alterado.

Variáveis e constantes

- Em Java recomenda-se que a declaração de variáveis utilize nomes iniciados com letras minúsculas.
- Caso o nome seja composto de mais de uma palavra, as demais devem ser iniciadas com letras maiúsculas.
- Exemplos:
  - `contador`
  - `posicaoAbsoluta`

Variáveis e constantes

- Em Java recomenda-se que as constantes sejam utilizadas com letras maiúsculas.
- Exemplo:
  - `PI`
  - `G`

Variáveis e constantes

- É comum em Java criar uma constante que fique disponível para vários métodos de uma única classe.
- Essas são geralmente as chamadas constantes de classe que são especificadas com as palavras reservadas **`static final`**.

Tipos de dados primitivos

- A linguagem Java é fortemente tipada, ou seja, cada variável do programa precisa ter um tipo declarado.
- Existem oito tipos primitivos em Java, sendo seis do tipo numérico (quatro tipos inteiros e dois de ponto flutuante), um do tipo char de caracteres, usado para caracteres com formato UNICODE, e um do tipo lógico chamado *boolean*, para valores de verdadeiro ou falso.

Tipos de dados primitivos

- Quando as variáveis de instância dos tipos de dados primitivos são declaradas em uma classe, atribuem-se valores default automaticamente a elas, a menos que se especifique.
- Às variáveis de instâncias dos tipos `char`, `byte`, `short`, `int`, `long`, `float` e `double` é atribuído o valor 0 (zero) por *default* (padrão). As variáveis do tipo *boolean* recebem por default o valor false.

Tipos inteiros

- A linguagem Java suporta os quatro tipos de inteiros. Os tipos inteiros são destinados a números sem parte fracionária.

Tipo	Tamanho em bytes	Intervalo
int	4	-2.147.483.648 a 2.147.483.647
short	2	-32.768 a 32.767
long	8	-9.223.372.036.854.775.808L a 9.223.372.036.854.775.807L
byte	1	-128 a 127

Tipos inteiros

- Por default os valores literais são tratados como inteiros simples (`int`), ou seja, valores de 32 bits.
- Não existe em Java o modificador `unsigned` disponível em outras linguagens, assim os tipos inteiros são sempre capazes de representar tanto valores positivos como negativos.

Tipos de pontos flutuantes

- Há dois tipos de pontos flutuantes.
- Os tipos de ponto flutuante denotam números com partes fracionárias.

Tipo	Tamanho em bytes	Intervalo
float	4	Aproximadamente +/- 3.40282347E+38F
double	8	Aproximadamente +/- 1.79769313486231570E+308

Tipos de pontos flutuantes

- O que difere os tipos de ponto flutuante em Java é a precisão oferecida: o tipo `float` permite representar valores reais com precisão simples (representação interna de 32 bits)
- Enquanto o tipo `double` oferece dupla precisão (representação interna de 64 bits).
- Os valore de ponto flutuante em Java estão em conformidade com o padrão IEEE 754.

Tipo caractere

- O tipo char denota caracteres segundo o esquema UNICODE de representação.
- O UNICODE foi projetado para lidar com todos os caracteres escritos do mundo, ele tem um código de 2 bytes, o que lhe permite 65.536 caracteres, bem mais que o conhecido ASCII/ANSI (255 caracteres).

Tipo caractere

- Apóstrofes, ou aspas simples é usada para denotar constantes char.
- Por exemplo, ‘H’ é um caractere e “H” é uma *string* contendo um único caractere.
- A linguagem Java não possui o tipo primitivo *string* conhecida em várias linguagens. Para manipulação de texto são utilizadas as classes `String` e `StringBuffer`.

Tipo caractere

- A linguagem Java permite usar sequências de caracteres de controle para os caracteres especiais

Seqüência	Nome
\b	Backspace (volta um espaço)
\t	Tab (tabulação)
\n	Linefeed (alimentação de linha)
\r	Carriage Return (retorno de carro)
\”	Aspas (duplas)
\’	Apóstrofe (apóstrofes)
\\	Barra Invertida (barra)

Tipo boolean

- Esse tipo tem dois valores possíveis: **falso** (**`false`**) e **verdadeiro** (**`true`**).
- Ele é empregado para efetuar **testes lógicos** usados por operadores relacionais que a linguagem Java suporta.

Atribuições e inicializações

- Após declarar uma variável, deve-se inicializá-la explicitamente por meio de uma instrução de atribuição.
- Faz-se uma atribuição a uma variável usando o sinal de igual (=).
- Um recurso interessante do Java é a possibilidade de declara e inicializar uma variável na mesma linha.

Atribuições e inicializações

```
public class AtribuicaoInicializacao {
    public static void main(String[] args) {
        int x, y;
        x = 123;
        float f = 123.45f;
        double d = 123.45;
        char c = '\n';
        boolean b = true;
        // atribuições ilegais
        y = 1.23;
        b = 1; //comum na programação C++
        f = 123.45;
    }
}
```

Conversão entre tipos primitivos

- É possível em Java transforma um tipo primitivo em outro, através de uma operação chamada ***typecast***.
- Para fazer essa operação basta colocando o tipo destino entre parêntesis, antes da expressão que será convertida (conversão explícita).
- Também existe a conversão implícita que é realizada pela máquina virtual.

Conversão entre tipos primitivos

- Java permite fazer conversões implícitas, atribuindo o valor de uma variável de um tipo para outra.
- As conversões permitidas são:
  - `byte → short → int → long → float → double → char → int`

Operadores aritméticos

- Os operadores aritméticos habituais da linguagem Java são: adição (+), subtração (-), multiplicação (*) e divisão (/).
- O operador de divisão denota divisão inteira se os dois argumentos forem inteiros e divisão de ponto flutuante caso contrário.
- A função resto da linguagem Java é denotada por %.

Exponenciação

- Java não tem operador para elevar uma quantidade a uma potência; é necessário usar o método ***pow***.
- O método ***pow*** faz parte da classe ***Math*** de ***java.lang***.
- O método ***pow*** precisa de argumentos que são do tipo ***double*** e retorna um ***double*** também.

Operadores aritméticos de atribuição

- A linguagem Java oferece vários operadores de atribuições para abreviar expressões de atribuição.

Operador	Expressão	Equivalência
+=	x += 7	x = x + 7
-=	x -= 5	x = x – 5
*=	x *= 3	x = x * 3
/=	x /= 2	x = x / 2
%=	x %= 4	x = x % 4

Exercícios

1. Faça um programa para mostrar o resultado da divisão de dois números.
2. Faça um algoritmo para ler três números inteiros e escrever a média dos números lidos.
3. Faça um algoritmo para ler os coeficientes de uma equação do segundo grau e escrever o valor de seu delta.
4. Faça um algoritmo para ler os catetos de um triângulo retângulo e escrever a sua hipotenusa.
5. Faça um algoritmo para ler duas variáveis inteiras e trocar o seu conteúdo.

Operadores relacionais

- A linguagem Java tem um conjunto completo de operadores relacionais

Operador Relacional	Descrição
==	Igualdade
!=	Desigualdade
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

Operadores lógico

- A linguagem Java tem um conjunto completo de operadores lógicos.

Operador Lógico	Descrição
&&	AND com curto-circuito
||	OR com curto-circuito
!	NOT
&	AND sem curto-circuito
|	OR sem curto-circuito

Operadores de incremento

- A linguagem Java fornece o operador de incremento unário (++).

Op	Chamado de	Expressão	Explicação
++	Pré-incremento	++x	Incrementa x por 1, depois utiliza o novo valor de x na expressão em que x reside.
++	Pós-incremento	x++	Utiliza o valor atual de x na expressão em que x reside, depois incrmenta x por 1.

Operadores de decremento

- A linguagem Java fornece o operador de decremento unário (--).

Op	Chamado de	Expressão	Explicação
--	Pré-decremento	--y	Decrementa y por 1, depois utiliza o novo valor de y na expressão em que y reside.
--	Pós-decremento	y--	Utiliza o valor atual de y na expressão em que y reside, depois decrementa y por 1.

Operadores

```
public static void main(String[] args) {
    int x = 1;
    int y = x++;
    System.out.println(x+" "+y); // 2 1
    x = 1;
    y = ++x;
    System.out.println(x+" "+y); // 2 2
    x = 2;
    y += x; // y = y + x
    System.out.println(x+" "+y); // 2 4
    x = 1;
    y -= x;
    System.out.println(x+" "+y); // 1 3
}
```

Operador ternário

- O Java possui um operador ternário que é utilizado baseado na seguinte sintaxe:

<expressão1> ? <expressão2> : <expressão3>;

```
public static void main(String[] args) {
    int x = 1, y = 2, maximo;
    maximo = (x > y) ? x : y;
    System.out.println(maximo);
}
```

Operadores de manipulação de bits

- A Java possui dois operadores de manipulação de bits, geralmente conhecidos como operadores de shift.

Operador	Descrição
>>	Deslocamento para direita
<<	Deslocamento para esquerda
>>>	Deslocamento para direita (*unsigned*)

Operadores de manipulação de bits

```
public static void main(String[] args) {
    int x = 4;
    int y;
    y = x >> 1;
    System.out.println(y); // 2
    y = x << 1;
    System.out.println(y); // 8
    x = -8;
    y = x >> 1;
    System.out.println(y); // -4
    x = -4;
    y = x >>> 1;
    System.out.println(y); // 2147483646
}
```

Exercício

- Altere a função que lê as notas do aluno e imprime a média para que a mensagem impressa se comporte da seguinte maneira:
  - Se a média foi maior ou igual a 7.0 a mensagem deve ser “Aprovado”. Caso contrário a mensagem deve ser “Estude mais para a final”.

Estrutura de seleção if/else

- A sintaxe de comando ***if*** em Java é da seguinte forma:

```
if (expressão lógica) {
    // bloco de comandos
}

if (expressão lógica) {
    // bloco de comandos
} else {
    // bloco de comandos
}
```

Estrutura de seleção if/else

```
public static void main(String[] args) {
    int x = 1, y = 2;
    if ( x > y ) {
        System.out.println("X é maior do que Y");
    } else if ( x < y ) {
        System.out.println("X é menor do que Y");
    } else {
        System.out.println("X é igual a Y");
    }
}
```

Exercícios

1. Escreva um programa que lê a idade de um usuário e em seguida diz se o usuário é ou não maior de idade.
2. Escreva um programa que lê um número inteiro e diz se o número é par ou ímpar.
3. Escreva um programa que lê três números e em seguida imprime quantos deles são iguais.
4. Escreva um programa que lê três número s inteiros e em seguida imprime os números em ordem crescente.

Estrutura de seleção switch

- Em Java, além do if, temos um outro comando de seleção switch, que possui a seguinte sintaxe:

```
switch (expressão) {
    case <constante1> :
        //bloco de comandos
    break;
    case <constante2> :
        //bloco de comandos
    break;
    ...
    default :
        //bloco de comandos
}
```

Estrutura de seleção switch

- A expressão a ser avaliada pode ser do tipo ***byte***, ***short***, ***int*** ou ***char***.
- O comando ***break*** entre um case e outro é para indicar que não deve ser feito nenhum outro teste.
- Caso não seja colocado todos os comandos serão efetuados sem efetuar o teste.

Estrutura de seleção switch

```
public static void main(String[] args) {
    int x = 1, y = 1;
    switch ( x ) {
        case 1 :
            y = y + 1;
        break;
        case 2 :
            y = y + 2;
```

Estrutura de seleção switch

```
            break;
        case 3 :
            y = y + 3;
        break;
        default :
        y = y + 4;
    }
    System.out.println(y); // 2
}
```

Estrutura de seleção switch

```
public static void main(String[] args) {
    int x = 1, y = 1;
    switch ( x ) {
        case 1 :
            y = y + 1;
        case 2 :
            y = y + 2;
        case 3 :
            y = y + 3;
        default :
            y = y + 4;
    }
    System.out.println(y); // 11
}
```

Exercício

- Escreva um programa que funcione como uma calculadora fazendo as quatro operações (adição, subtração, multiplicação e divisão). Segue menu:

```
### Calculadora ####
Digite dois números: 3 4
Digite a operação: +
Resultado de 3 + 4 = 7
```

- O programa deve receber dois números (operando e operador) e a operação (‘+’, ‘-’, ‘*’, ‘/’).
- Seu programa precisa checar se o operador é zero para a operação de divisão.

Estrutura de repetição for

- A estrutura de repetição ***for*** tem a seguinte sintaxe:

```
for (<expressãoInicial>; <teste>; <expressaoDeAlteração>) {
    // bloco de comandos
}
```

Estrutura de repetição for

- Neste exemplo as variáveis ***i*** e ***j*** foram declaradas no próprio ***for*** e o seu escopo será apenas dentro do bloco de comandos.

```
public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
        System.out.println(i);
    }
    for (int i = 0, j = 0; i < 5; i++, j++) {
        System.out.println(i+" "+j);
    }
}
```

Exercício

1. Como ficaria o algoritmo para calcular a media dos 5 alunos usando repetição fixa?
2. Escreva um algoritmo que lê 5 números inteiros e em seguida mostra a soma de todos os ímpares lidos.
3. Altere o algoritmo anterior para que ele considere apenas a soma dos ímpares que estejam entre 100 e 200.
4. Construa um algoritmo que leia um conjunto de 20 números inteiros e mostre qual foi o maior e o menor valor fornecido.

Estrutura de repetição while

- O comando de repetição ***while*** tem a seguinte sintaxe:

```
while (expressão lógica) {
    // bloco de comandos
}
```

Estrutura de repetição while

```
public static void main(String[] args) {
    int i = 1;
    int fat = 1;
    while (i <= 5) {
        fat = fat * i;
        i++;
    }
    System.out.println(fat); // 120
}
```

Exercício

1. Como seria um programa para calcular a média de 50 alunos da uma turma?
2. Escreva um programa que calcula o produto de dois números lidos sem usar o operador de multiplicação (‘*’).
3. Construa um algoritmo que fica lendo indefinidamente números positivos. Caso o numero lido seja igual a 0 o algoritmo pára de ler números e imprime a média dos números pares lidos anteriormente.

Exercício Desafio

- Escreva um programa que lê um número e em seguida calcula e imprime seu fatorial.
  - 4! = 4 * 3 * 2 * 1
  - n! = n * (n – 1)!

Estrutura de repetição do/while

- A terceira estrutura de repetição do Java e o ***do***, cuja a sintaxe é a seguinte:

```
do {
    // bloco de comandos
} while (expressão lógica);
```

Estrutura de repetição do/while

```
public static void main(String[] args) {
    int ant = 1, meio = 1, i = 1;
    int fibo;
    do {
        fibo = ant + meio;
        ant = meio;
        meio = fibo;
        i++;
    } while (i <= 5) ;
    System.out.println(fibo); // 13
}
```

Comandos break e continue

- Temos dois comandos especiais para as estruturas de repetição: ***break*** e ***continue***.
- O comando ***break*** deixa uma estrutura de repetição incondicionalmente.
- O comando ***continue*** executa a próxima iteração do laço, independente dos comandos que seriam executados na sequência.

Comandos break e continue

```
public static void main(String[] args) {
    int i = 1;
    while ( i <= 10 ) {
        if ( i == 5 ) {
            break;
        }
        System.out.println(i);
        i++;
    } // vai imprimir 1,2,3 e 4
```

Comandos break e continue

```
        i = 0;
        while ( i < 10 ) {
            i++;
            if ( i % 2 == 0 ) {
                continue;
            }
            System.out.println(i);
        } // vai imprimir 1,3,5,7 e 9
}
```

Exercício

- Como ficaria o algoritmo para calcular a média dos 50 alunos com teste no final usando o comando do-while?