CURSO: Ciência da Computação	**DATA: ____/____/ 2013**
PERÍODO: 4º.	**PROFESSOR: Andrey**
DISCIPLINA: Técnicas Alternativas de Programação	**AULA: 02**

APRESENTAÇÃO:

Apresentação; conceitos básicos da linguagem java; estrutura de um programa java; tipos de variáveis; comando de decisão if else; o comando de repetição while, seu uso combinado com o comando if e o comando de repetição for.

DESENVOLVIMENTO:

1. Histórico

A Sun Microsystems, em 1991, financiou um projeto de pesquisa interno sobre dispositivos eletrônicos inteligentes com o codinome Green que resultou em uma linguagem de programação baseada em C++, que o seu criador, James Gosling chamou de OAK. Mais tarde este nome foi mudado para JAVA pois já existia uma linguagem com o nome OAK. Com a explosão de popularidade da World Wide Web em 1993 a equipe da Sun viu o potencial da nova linguagem para adicionar conteúdo dinâmico às páginas web.
Java foi anunciado oficialmente em uma conferência em maio de 1995. Java agora é utilizado para desenvolver aplicativos corporativos de grande porte, em aplicações para web e dispositivos eletrônicos como celulares, pagers e PDAs.

Tecnologia Java:

> J2SE – Java Standard Edition – Plataforma padrão do Java. Serve para o desenvolvimento de aplicações desktop comuns.
> J2EE – Java 2 Enterprise Edition – Baseado no J2SE com a adição de serviços web, componentes, gerenciamento e bibliotecas de comunicação.
> J2ME – Java 2 Micro Edition – Plataforma Java para dispositivos móveis e sem fio.
> Java Card – Tecnologia java para smartcards.
> etc...

- Java Virtual machine - JVM
- Java RunTime Environment – JRE
- Java Development Kit – JDK – Kit de desenvolvimento que contém o JRE, o compilador e as bibliotecas.
- API – Application Programming Interface.

Http://.java.com

Ambientes de desenvolvimento:

NetBeans – http://www.netbeans.org

Eclipse – http://www.eclipse.org

Versões:

Versão	*Data*
JDK1.0	23 de janeiro de 1996[7]
JDK 1.1	19 de fevereiro de 1997
J2SE 1.2	8 de dezembro de 1998
J2SE 1.3	8 de maio de 2000

J2SE 1.4	6 de fevereiro de 2002
J2SE 5.0	30 de setembro de 2004
Java SE 6	11 de dezembro de 2006
Java SE 7	28 de julho de 2011
Java SE 8	18 de março de 2014
Java SE 9	21 de setembro de 2017
Java SE 10	20 de março de 2018
Java SE 11	September 25, 2018[8]
Java SE 12	19 de março de 2019
Java SE 13	17 de setembro de 2019
Java SE 14	17 de março de 2020

2. Estrutura de um programa Java

Por exemplo, observe o código JAVA abaixo :

```
class Programa1
{
   public static void main(String args[])
   {
      System.out.println(" Oi, mundo! ");
   }
}
```

Na linha seguinte (2), simplesmente indicamos o início do bloco principal (Não se assuste com o comprimento da expressão e com as palavras desconhecidas, nós repetiremos até você decorar!).

Nosso programa realmente é apenas uma linha (3) onde imprimimos a mensagem "Oi mundo ", na tela.

Esta linha também contém palavras separadas por pontos. Usaremos também esta expressão sempre para imprimir mensagens na tela (Você só precisa mudar a frase que está entre aspas e pronto!).

Em seguida nas linhas (4) e (5) fechamos o bloco principal e o programa respectivamente.

Para que este programa seja executado, é necessário salvá-lo em um arquivo texto chamado Programa1.java. Logo em seguida é necessário compilar este programa. Isto é feito com o comando:

```
%> javac Programa1.java
```

É necessário que o arquivo do programa tenha o mesmo nome da classe principal do programa. Uma vez compilado, é gerado um arquivo chamado Programa1.class. Este arquivo é executado pela Máquina Virtual Java com o comando:

```
%> java Programa1
```

3. Tipos de Dados primitivos

Java tem oito tipos simples: byte, short, int, long, char, float, double e boolean. Estes podem ser classificados em quatro grupos:

a) Inteiros: byte, short, int e long, que são para os números de valor inteiro.
b) Os números de ponto flutuante: float e double, que representam os números com precisão de fração.
c) Caracteres: char, que representam símbolos de um conjunto de caracteres, tais como letras e dígitos.
d) Lógicos: boolean, que é um tipo especial usado para representar valores lógicos (true e false);

Nome	Largura (bits)	Faixa
byte	8	-128 a 127
short	16	-32.768 a 32.767
int	32	-2.147.483.648 a 2.147.483.647
long	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	32	1,7e-38 a 1,7e +38
double	64	3,4e-38 a 3,4e+38

4. Identificadores

Agora daremos um "zoom" para abordar todos os aspectos gerais da sintaxe Java. Os programas Java são uma coleção de espaços em branco, comentários, palavras-chave, identificadores, operadores e separadores.

4.1. Espaço em Branco:

Java é uma linguagem de formato livre. Você não precisa indentar nada para que ela funcione adequadamente. O recurso de indentação é utilizado apenas para uma melhor visualização do programa como um todo.

4.2. Comentários:

Existem três tipos de comentários do código-fonte: de uma linha, de várias linhas e de documentação. Os comentários de uma linha começam com // e terminam no final da linha. Este estilo de comentário é útil para explicações breves de uma única linha de código.

```
a = 42;           // se 42 é a resposta, qual é a pergunta?

/*
* Este código é um pouco complicado...
* Vou tentar explicá-lo aqui:
*/
```

4.3. Palavras-Chave Reservadas

Palavras-Chave Reservadas de Java:

Abstratct	boolean	break	byte	byvalue
case	cast	catch	char	class

Const	continue	default	do	double
Else	extendes	false	final	finally
Float	for	future	generic	goto
If	implements	import	inner	instanceof
Int	interface	long	native	new
Null	operator	outer	package	private
Protected	public	rest	return	short
Static	super	swicht	synchronized	this
Throw	throws	transient	true	try
var	void	volatile	while	

4.4. Identificadores:

Os identificadores são usados para nomes de classe, métodos e variáveis. Um identificador pose se qualquer sequência descritiva de caracteres de letras minúsculas, números, caracteres de sublinhado e símbolo de cifrão. Eles não podem começar com um número, caso contrário serão confundidos com um literal numérico descrito a seguir. Java é sensível a maiúsculas e minúsculas, o que significa que VALOR é um identificador diferente de Valor, e por sua vez diferente de valor. Alguns exemplos de identificadores válidos incluem horaDoDia, temp-val, a4, e $- (embora eu não recomendaria $). Os nomes de variáveis inválidos incluem 1more, 3$, a:b, #foo e @2.

O grupo Java seguiu a convenção de nomes de identificadores, de forma a nomear todos os métodos e variáveis públicos com letra minúscula na frente marcando o início da palavra descritiva subsequente com uma letra maiúscula, tal como poxItem, valorCorrente e obterHoraDoDia.

4.5. Operadores:

operador	***operação***
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo
++	incremento
--	decremento
+=	Atribuição aditiva
-=	Atribuição subtrativa
*=	Atribuição multiplicativa
/=	Atribuição de divisão
%=	Atribuição de módulo

5. Estrutura Condicional

Em um algoritmo, quando você precisa indicar que alguns comandos somente serão executados quando uma determinada condição for verdadeira você vai usar a estrutura condicional.

Se (<condição>)
inicio
  <comando 1>
  <comando 2>
  <comando 3>
  ...
fim

por exemplo:

```
se (a >b)
inicio
   temp <- a
   a <- b
   b <- temp
fim
```

note que os comandos:

```
   temp <- a
   a <- b
   b <- temp
```

somente serão executados quando o valor da variável a for maior que o valor da variável b
em java:

```
if (a>b)
{
   temp = a;
   a = b;
   b = temp;
}
```

Ex: Um programa que leia dois números e os imprima em ordem crescente.

```
import  java.io.*;
import java.util.Scanner;

public class Ordena
{
      public static void main(String args[ ]) throws IOException
      {
            String valor;
            int a;
            int b;
            int temp;

            Scanner teclado = new Scanner(System.in);

            System.out.println("Digite o primeiro numero");
```

```
            a = teclado.nextInt();
            System.out.println("Digite o segundo numero");
            b = teclado.nextInt();

            if (a>b)
            {
                  temp = a;
                  a = b;
                  b = temp;
            }

            System.out.println(a);
            System.out.println(b);
      }
}
```

6. Estrutura Condicional Composta

Em um algoritmo, quando você precisa indicar que alguns comandos somente serão executados quando uma determinada condição for verdadeira e outros comandos apenas serão executados quando a mesma condição for falsa, você vai usar a estrutura condicional composta.

Se (<condição>)
{
  <comando 1>
  <comando 2>
  <comando 3>
}
senão
{
  <comando 4>
  <comando 5>
  <comando 6>
}

No caso acima os comandos 1,2 e 3 somente serão executados quando a condição for verdadeira, e os comandos 4, 5 e 6 somente serão executados quando a condição for falsa. Por exemplo:

```
leia a
se (a >= 0)
{
   escreva "a é positivo"
}
senão
{
   escreva "a é negativo"
}
```

será escrito “a é positivo” quando a>=0 for verdadeiro e será escrito “a é negativo” quando a>=0 for falso.

em java:

```
if (a>=0)
{
```

```
   System.out.println("a é positivo");
}
else
{
   System.out.println("a é negativo");
}
```

Ex: Um programa que leia dois números e os imprima em ordem crescente.

```
import  java.io.*;
import java.util.Scanner;
public class Positivo
{
      public static void main(String args[ ]) throws IOException
      {
            String valor;
            double a;
            Scanner teclado = new Scanner(System.in);
            System.out.println("Digite o numero");
            a = teclado.nextDouble();
            if (a>=0)
            {
                  System.out.println("a é positivo");
            }
            else
            {
                  System.out.println("a é negativo");
            }
      }
}
```

7. Estrutura de repetição While

Esta estrutura é usada para indicar que uma ou mais ações ou comandos serão executadas mais de uma vez. No caso da estrutura while (enquanto), a repetição dos comandos somente será realizada quando a condição for verdadeira.

A estrutura enquanto é construída desta forma:

enquanto (<condição>)
{
      <comando>
}

por exemplo:

leia a
enquanto (a != 5)
{
      escreva você não digitou o número 5"
}
escreva " até que enfim você digitou o número 5

Note que enquanto não for digitado o número 5 será executado comando que está entre as chaves. Quando for digitado 5, a repetição termina e será executado o comando seguinte.

Em java:

```
while (a != 5)
{
      System.out.println("você não digitou o numero 5");
}
System.out.println("finalmente você digitou o numero 5");
```

Contador:

Quando eu quero que uma repetição seja feita n vezes, onde n é um número inteiro conhecido, usamos um contador. Um contador é uma variável inteira que irá contar quantas vezes foi feita a repetição.

Por exemplo:

```
cont <-1
enquanto (contador <=10)
{
      multi <- 6*cont
      escreva " 6 x "+cont+" = " +multi
}
```

em java :

```
public class Tabuada
{
      public static void main(String args[ ])
      {
            int cont;
            int mult;
            cont =1;
            while (contador <=10)
            {
                  multi = 6*cont;
                  System.out.println (" 6 x "+cont+" = " +multi);
            }
      }
}
```

8. Estrutura de repetição While em combinação com o comando if

Alguma vezes, é necessário combinarmos o comando if com a estruturas de repetição. Por exemplo, se precisarmos contar quantas vezes foi digitado um determinado valor.

Um programa que leia números inteiros enquanto não for digitado o valor -1 e conte quantas vezes foi digitado o número 5.

```
numero = 0;
cont <- 0
enquanto ( numero != -1)
{
      leia número
      se (numero = 5)
      {
            cont -<- cont +1
```

```
      }
}
escreva cont
```

note que serão lidos números enquanto não for digitado o número -1 e sempre que o valor digitado for 5 será somado 1 ao valor do contador.

Em java:

```
import java.io.*;
import java.util.Scanner;
public class Conta
{
      public static void main(String args[ ]) throws IOException
      {
            String valor;
            int numero = 0;
            int cont = 0;
            Scanner teclado = new Scanner(System.in);
            while (numero != -1)
            {
                  System.out.println("Digite o numero");
                  numero = teclado.nextInt();
                  if (numero == 5)
                  {
                        cont = cont + 1;
                  }
            }
            System.out.println(" foram digitadas "+cont+" vezes o numero 5");
      }
}
```

9. Estrutura de repetição FOR

A Estrutura de repetição FOR é a estrutura de repetição mais usada em programação. Praticamente todas as linguagens de programação possuem uma estrutura equivalente. O FOR é usado quando, no inicio da repetição, já se conhece o número de vezes que será feita a repetição. Ex.:

```
para (cont <- 1; cont <=10;cont<- cont+1)
{
      multi <- 7 * cont
      escreva multi
}
```

Neste exemplo é feito o cálculo da tabuada de 7. Note que a atribuição do valor inicial do contador, a condição de continuar repetindo e o incremento do contador é feito na declaração do comando.

em Java

```
public class Tabuada
{
      public static void main(String args[ ])
      {
            int multi;
            int cont;
```

```
            for (cont = 1; cont <=10;cont++)
            {
                  multi = 7 * cont;
                  System.out.println(multi);
            }
      }
}
```

comparando com o while:

```
                  for                                     while
      for (cont = 1; cont <=10;cont++)    |     cont =1;
      {                                   |     while(cont<=10)
            multi = 7 * cont;             |     {
            System.out.println(multi);    |           multi = 7 * cont;
      }                                   |           System.out.println(multi);
                                          |           cont ++
                                          |     }
```

A repetição pode ser controlada por uma variável. Por exemplo: um programa que lê um número e a seguir uma sequência de n números e imprime o maior deles.

```
import java.io.*;
import java.util.Scanner;
public class Tabuada
{
      public static void main(String args[ ]) throws IOException
      {
            int numero;
            int cont;
            int n;
            Scanner teclado = new Scanner(System.in);
            System.out.println("Digite a quantidade de números");
            n = teclado.nextInt();
            System.out.println("Digite um numero");
            numero = teclado.nextInt();
            maior = numero;
            for (cont = 1; cont <=n;cont++)
            {
                  System.out.println("Digite um numero");
                  numero = teclado.nextInt();
                  if (numero>maior)
                        maior = numero;
                  System.out.println("o maior numero é = "+maior);
            }
      }
}
```

As estruturas de repetição podem ser usadas em combinação umas com as outras. Este uso é muito comum, principalmente quando necessitamos realizar operações sobre conjuntos de conjuntos. Por exemplo conjuntos de turmas de alunos, matrizes, planilhas, etc.

Neste exemplo faremos um programa que leia as 4 notas de 30 alunos da turma e escreve a média de cada aluno e a media da turma.

```
import java.io.*;
import java.util.Scanner;

public class Medias
{
      public static void main (String args[]) throws IOException
      {
            double nota;
            double somaNotas;
```

```
		double somaMedias;
		double media;
		double mediaTurma;
		String nome;
		int contNotas,contAlunos, n = 4, alunos = 1;
		Scanner teclado = new Scanner(System.in);
		somaMedias = 0;
		for (contAlunos=1;contAlunos<=alunos;contAlunos++)
		{
			System.out.println("digite o nome do aluno");
			nome = teclado.nextLine();
			somaNotas = 0;
			for (contNotas=1;contNotas<=n;contNotas++)
			{
				System.out.println("digite a nota "+contNotas);
				nota = teclado.nextDouble();
				somaNotas = somaNotas + nota;
			}
			media = somaNotas/n;
			somaMedias = somaMedias + media;
			System.out.println("a media de "+nome+" é "+media);
		}
		mediaTurma = somaMedias/alunos;
		System.out.println("a media da turma é "+mediaTurma);

	}
}
```

INTEGRAÇÃO:

1) fazer um programa em java que calcule e imprima a tabuada de 6
2) fazer um programa em java que calcule e imprima as raízes da equação x^2 +x -6 =0
3) fazer um programa em java que mostre o seu nome completo, o nome do time de futebol de sua preferência e o bairro onde você mora.
4) Construa um programa em Java que leia um número e diga se ele é positivo.
5) Construa um programa em Java que leia um numero inteiro e diga se ele é par ou ímpar.
6) Construa um programa em Java que leia um número x, calcule e escreva o valor da função f(x), dada por:

   0<=x<5, f(x) = x
   5<=x<10; f(x) = 2x+1
   x>=10; f(x) = x-3

7) Construa um programa em Java que leia três números e diga se eles podem ser os lados de um triângulo isósceles, equilátero ou escaleno.
8) Fazer um programa em java que leia um número inteiro e calcule e imprima a tabuada deste número de 1 a 10.
9) Fazer um programa em java que leia números inteiros enquanto não for digitado o número -1, e calcule e imprima a soma destes números.
10) Fazer um programa que leia os valores do peso e da altura de pessoas, enquanto não for digitado o número -1, conte e escreva quantas pessoas estão acima do peso. A condição (peso /(altura*altura)) <= 25 diz que a pessoa está no peso normal.
11) Fazer um programa em java que calcule e imprima a soma dos 10 primeiros múltiplos de 3.

12) Fazer um programa em java que leia as 4 notas de 30 alunos da turma e escreva a maior nota de cada aluno e a maior nota da turma.

BIBLIOGRAFIA:

FURGERI, SÉRGIO. *Java 2 Ensino Didático. Desenvolvendo e Implementando Aplicações*. ed. Érica. São Paulo, 2002.
DEITEL, H. M. e DEITEL, P. J.. Java, como Programar. Ed. Bookman. Porto Alegre. 2001.
ARNOLD Ken, GOSLING James: "The Java Programming Language Second Edition", Addison-Wesley, 1997.