LTP 1 – Java Básico

Aula 7

Tópicos

1 - Solucionando os exercícios da aula passada

1.1 - Exercício do Número Perfeito

Dizemos que um número é perfeito se a soma de seus fatores, incluindo o 1 (mas não o próprio número) é igual ao número. Por exemplo, 6 é um número perfeito porque 6=1+2+3. Escreva um método chamado perfeito que determina se o parâmetro numero é um número perfeito. Exiba os fatores de cada número perfeito para confirmar que o número é de fato perfeito. Teste todos os números de 1 a 1000.

```
import javax.swing.*;
public class Ex1
{
   public static void main(String[] args)
   {
      Numeros var = new Numeros();
      String x = JOptionPane.showInputDialog(null,"Digite um número");
      int valor = Integer.parseInt(x);
      var.perfeito(valor);
   }
}
```

```
public class Numeros
{
   public void perfeito(int numero)
   {
      int i=1,soma=0;
      for(i=1;i<numero;i++)
      {
```

```
            if(numero%i==0)
            {
               System.out.println(i);
               soma=soma+i;
            }
         }
         System.out.println("soma = "+soma);
         if(soma==numero)
            System.out.println(numero+" é um número perfeito");
         else
            System.out.println(numero+" não é um número perfeito");
      }
}
```

1.2 - Exercício do Cálculo da Distância

Escreva um método que calcula a distância (variável de instância) entre dois pontos (x1,y1) e (x2,y2), com a leitura de parâmetros no próprio método e retorno do valor da distância. Escreva outro método com o mesmo nome, mas que passa os valores de x1, x2, y1 e y2, e que também calcula a distância entre dois pontos e também retorna o valor da distância.

```
public class Ex1
{
   public static void main(String[] args)
   {
      DistanciaEuclidiana var = new DistanciaEuclidiana();
      System.out.println("distância = "+var.distancia());
      System.out.println("distância = "+var.distancia(1,1,3,3));
   }
}
```

```
import javax.swing.*;
public class DistanciaEuclidiana
{
   private double distancia;
   public void setDistancia(double distancia) {
      this.distancia = distancia;
   }
   public double getDistancia() {
      return distancia;
   }
   public double distancia() {
      String x = JOptionPane.showInputDialog(null,"Digite o valor de x1");
      int x1=Integer.parseInt(x);
      x = JOptionPane.showInputDialog(null,"Digite o valor de y1");
      int y1=Integer.parseInt(x);
      x = JOptionPane.showInputDialog(null,"Digite o valor de x2");
      int x2=Integer.parseInt(x);
      x = JOptionPane.showInputDialog(null,"Digite o valor de y2");
      int y2=Integer.parseInt(x);
      double valor=Math.sqrt(Math.pow((x1-x2),2)+Math.pow((x1-x2),2));
      setDistancia(valor);
      return getDistancia();
   }
```

```
   public double distancia(int x1,int y1, int x2, int y2)
   {
      double valor=Math.sqrt(Math.pow((x1-x2),2)+Math.pow((x1-x2),2));
      setDistancia(valor);
      return getDistancia();
   }
}
```

1.3 - Exercício sobre Aprender a Contar

Os computadores estão desempenhando um crescente papel na educação. Escreva um programa que ajudará um estudante do ensino fundamental a aprender multiplicação. Utilize algum gerador de números aleatórios para produzir dois números inteiros positivos (entre 0 e 10). O Programa deve então fazer ao usuário o seguinte tipo de pergunta: Quanto é 6 vezes 7? Depois o programa deve ler a resposta e informar se o usuário acertou ou não.

```
public class Ex1
{
   public static void main(String[] args)
   {
      Tabuada var = new Tabuada();
      var.ajudaAprender();
   }
}
```

```
import java.util.*;
import javax.swing.*;
public class Tabuada
{
   public void ajudaAprender()
   {
      Random var = new Random();
      int x=var.nextInt(10);
      int y=var.nextInt(10);
      String texto =String.format("Quanto é "+x+" vezes "+y+"?");
      String resp=JOptionPane.showInputDialog(null,texto);
      int valor=Integer.parseInt(resp);
      if(valor==(x*y))
         JOptionPane.showMessageDialog(null,"Parabéns, você acertou!!!");
      else
         JOptionPane.showMessageDialog(null,"Você errou!!!");
   }
```

2 - Arrays

Na linguagem C vimos uma estrutura chamada matriz, que em Java se chama ARRAY, que são estruturas de dados com valores do mesmo tipo. Ou seja, array é um grupo de variáveis que contém valores que são todos do mesmo tipo (tipos primitivos ou por referência). Em Java os arrays são objetos, portanto são considerados tipos por referência.

Para referenciar um elemento particular de um array, especificamos o nome da referência para o array e o número da posição do elemento do array, esse número de posição também é chamado de índice ou subscrito do elemento.

2.1 - Declarando e criando arrays

Os objetos de array ocupam espaço na memória, e como os outros objetos, são criados com a palavra-chave new. Para criar um objeto de array, o programador especifica o tipo e número de elementos do array, como parte de uma expressão de criação do array.

Formato Geral:
tipo nome[ ] = new tipo[tamanho];

Ou pode ser feita em dois passos:
tipo nome[ ];//declara a variável do tipo array
nome = new tipo[tamanho];//cria o array. Cria a referência ao objeto.

Exemplo:
int c[ ]=new int[10];

Observação:
Em uma declaração da variável do tipo array, não pode especificar o número de elementos entre os colchetes, que resultaria em um erro de sintaxe. int c[12]; //não pode

2.2 - Criando vários arrays em uma única declaração

Exemplo:
String b[ ] = new String[100], x[ ] = new String[27];

Ou fazer , para ficar mais legível:

String b[ ] = new String[100];
String x[ ] = new String[27];

2.3 - Declarar múltiplas variáveis array em uma única declaração

Quando um array é declarado, o tipo do array e os colchetes podem ser combinados no começo da declaração, para identificar que todos os identificadores são variáveis array.

Exemplo:
double[ ] valor1,valor2;//neste caso as variáveis valor1 e valor2 seriam ambas do tipo array.

2.4 - Exemplo completo de uso de array

```
import java.util.*;
public class ExemploArray
{
   public void testaArray()
   {
      int i;
      Random var = new Random();
      int valor[]=new int[10];
      for(i=0;i<10;i++)
      {
         valor[i]=var.nextInt(100);
         System.out.println("valor["+i+"] = "+valor[i]);
      }
   }
```

```
}
```

```
public class Main {
   public static void main(String[] args)
   {
     ExemploArray var = new ExemploArray();
     var.testaArray();
   }
}
```

Na tela:

```
valor[0] = 39
valor[1] = 69
valor[2] = 52
valor[3] = 11
valor[4] = 41
valor[5] = 50
valor[6] = 45
valor[7] = 87
valor[8] = 17
valor[9] = 15
```

2.5 - Inicializando um array

Um programa pode criar um array e inicializar seus elementos com um inicializador de array, que é uma lista de expressões separadas por vírgula, colocadas entre chaves. Neste caso o comprimento do array é determinado pelo número de elementos da referida lista. Essa declaração não requer o new para criar o objeto array.

Exemplo:

```
int n[ ] = {1,2,3,4,5,6};
```

Observação:

Para saber a quantidade de elementos de um array você deve usar nomedoarray.length.

Exemplo:

```
int n[ ] = {1,2,3,4,5,6};
int tamanho;
tamanho = n.length;
```

Nesse caso o número 6 seria armazenado na variável tamanho.

2.6 - Inicializando arrays com uma constante

Para utilizar uma constante, basta acrescentar a palavra final antes do tipo da variável e inicializá-la (obrigatoriamente). O valor armazenado numa constante não pode ser alterado ao longo da execução do programa.

Exemplo:

```
final int tamanho=10;//declara a constante
int valor[ ] = new int[tamanho];//cria o array
```

Observações:

- Se uma tentativa de acessar o valor de uma variável final for feita antes dela ser inicializada, o compilador emite uma mensagem de erro.

- Atribuir um valor a uma constante depois de a variável ter sido inicializada é um erro de compilação.

3 - Passagem de argumentos para métodos

Em muitas linguagens de programação podemos a passagem de parâmetro por valor ou por referência (também conhecidas como chamada por valor e chamada por referência).

Quando um argumento é passado por valor, apenas uma cópia do valor original é passado, ou seja, as alterações na cópia não alteram o valor original.

Quando um argumento é passado por referência, o método que recebe o argumento pode alterar o valor original, se necessário. Passar por referência aprimora o desempenho eliminando a necessidade de copiar quantidades de dados possivelmente grandes.

Ao contrário de outras linguagens, o Java não permite ao programador passar por parâmetro ou por referência – todos os argumentos são passados por valor.

Uma chamada de método pode passar dois tipos de valores:

- Valores primitivos, por exemplo do tipo int, double, etc.: quando um método modifica um parâmetro do tipo primitivo, essas alterações não tem efeito algum no valor original.
- Cópias de referências para objetos (inclusive arrays): como o valor passado referencia o mesmo objeto original, assim qualquer alteração no método que recebeu uma cópia da referência pode afetar o valor original e utilizar os métodos do objeto original.

3.1 - Passando arrays para métodos

Para passar um array como argumento de um método, você deve especificar o nome do array sem colchetes, pois todo array conhece seu próprio comprimento. Para um método receber um array como parâmetro, o mesmo deve especificar um parâmetro do tipo array, que deve ser do mesmo tipo que o array que foi passado como argumento.

Exemplo:

```
public class Ex1
{
   public static void main(String[] args)
   {
      int vetor[]={3,5,7,9};
      ExemploArray var = new ExemploArray();
      var.mostraArray(vetor);
   }
}
```

```
public class ExemploArray
{
   public void mostraArray(int valor[])
   {
      int i;
      for(i=0;i<valor.length;i++)
         System.out.println(valor[i]);
      //ou
      System.out.println("Ou ainda");
      for(int n:valor)
         System.out.println(n);
```

```
   }
}
```

Observação:

- Quando passamos um array como argumento, passamos uma cópia da referência do array original. Então o que alterar no método, **altera o array original**. Veja exemplo abaixo:

Exemplo:

```
public class Ex1
{
   public static void main(String[] args)
   {
      int i,vetor[]={3,5,7,9};
      ExemploArray var = new ExemploArray();
      System.out.println("Cópia do vetor");
      var.mostraArray(vetor);
      System.out.println("Vetor original");
      for(int n:vetor)
         System.out.printf("%4d\t",n);
      System.out.println("");
      System.out.println("vetor[3] = "+vetor[3]);
   }
}
```

```
public class ExemploArray
{
   public void mostraArray(int valor[])
   {
      int i;
      for(i=0;i<valor.length;i++)
      {
         valor[i]=valor[i]*2;
         System.out.printf("%4d\t",valor[i]);
      }
      System.out.println("");
   }
}
```

Na Tela:

```
Cópia do vetor
   6        10        14        18
Vetor original
   6        10        14        18
vetor[3] = 18
```

3.2 - Passando um elemento específico do array

Quando você quer passar para um método o valor armazenado numa posição específica do array, deverá especificar essa posição. E o método que recebe esse argumento, deverá especificar um parâmetro comum, sem ser array.

Exemplo:

```
public class Ex1
```

```
{
   public static void main(String[] args)
   {
      int vetor[]={3,5,7,9};
      ExemploArray var = new ExemploArray();
      var.mostraArray(vetor[3]);
   }
}
```

```
public class ExemploArray
{
   public void mostraArray(int aux)
   {
      System.out.println("dobro de "+aux+" = "+aux*2);
   }
}
```

Observação:

- Nesse caso, o valor alterado no método **não altera o valor original** do elemento do array. Veja exemplo a seguir:

Exemplo:

```
public class Ex1
{
   public static void main(String[] args)
   {
      int i,vetor[]={3,5,7,9};
      ExemploArray var = new ExemploArray();
      var.mostraArray(vetor[3]);
      System.out.println("vetor[3] = "+vetor[3]);
   }
}
```

```
public class ExemploArray
{
   public void mostraArray(int aux)
   {
      System.out.println("valor passado como argumento = "+aux);
      aux = aux*2;
      System.out.println("em dobro = "+aux);
   }
}
```

4 - Exercícios

a) Uma empresa quer transmitir dados por telefone, mas quer que os mesmos sejam criptografados. Cada mensagem transmitida é composta por 4 dígitos de números inteiros.

Crie uma classe para criptografar a mensagem da seguinte forma:

- Adicione 7 a cada dígito
- Obtenha o resto da divisão de cada dígito por 10
- Troque o primeiro dígito pelo terceiro
- Troque o segundo dígito pelo quarto

- Mostre a nova mensagem com os novos 4 dígitos.

Crie outra classe para descriptografar para os dígitos originais.

5 - Solução dos exercícios

5.1 - Exercício a

```
public class Ex1
{
   public static void main(String[] args)
   {
      Criptografa var1=new Criptografa();
      Descriptografa var2=new Descriptografa();
      int vetor[]={5,7,9,11};
      System.out.printf("valor atual do vetor\n");
      for(int i=0;i<vetor.length;i++)
        System.out.printf("%d  ",vetor[i]);
      var1.recebe(vetor);
      System.out.printf("\nvalor atual do vetor\n");
      for(int i=0;i<vetor.length;i++)
        System.out.printf("%d  ",vetor[i]);
      var2.valorOrginal(vetor);
   }
}
public class Criptografa
{
   public void recebe(int digitos[])
   {
      for(int i=0;i<digitos.length;i++)
         digitos[i]=digitos[i]+7;
      int aux;
      aux=digitos[0];
      digitos[0]=digitos[2];
      digitos[2]=aux;
      aux=digitos[1];
      digitos[1]=digitos[3];
      digitos[3]=aux;
      System.out.printf("\nNumeros criptografados\n");
      for(int i=0;i<digitos.length;i++)
        System.out.printf("%d  ",digitos[i]);
   }
}
public class Descriptografa
{
  public void valorOrginal(int digitos[])
  {
      for(int i=0;i<digitos.length;i++)
         digitos[i]=digitos[i]-7;
      int aux;
      aux=digitos[0];
```

```
        digitos[0]=digitos[2];
        digitos[2]=aux;
        aux=digitos[1];
        digitos[1]=digitos[3];
        digitos[3]=aux;
        System.out.printf("\nNumeros descriptografados\n");
        for(int i=0;i<digitos.length;i++)
          System.out.printf("%d  ",digitos[i]);
        System.out.printf("\n");
    }
}
```

Na tela:

valor atual do vetor
5 7 9 11
Numeros criptografados
16 18 12 14
valor atual do vetor
16 18 12 14
Numeros descriptografados
5 7 9 11