Universidade Federal de Lavras
Departamento de Ciência da Computação
GCC – 110 – Programação Orientada a Objetos

Lista de Exercícios 1:

Assunto: Ambientação com a Linguagem Java

I – Observações:

1) O trabalho é individual. É permitido discutir os problemas e estratégias de solução com outros colegas, mas quando se tratar de escrever ou implementar computacionalmente as soluções, isto deve ser feito separadamente. O trabalho pode ser feito em Linguagem Java.

2) Forma de entrega: O trabalho deve ser entregue em formato digital por meio do ***Moodle***. Utilizar a opção ***"Lista de Exercícios 1".*** Anexe **um único arquivo .zip** contendo todos os arquivos do trabalho**.** O nome do arquivo .zip deve conter o primeiro e o último nome do aluno. Por exemplo: **Cristiano_Castro.zip. Obs:** O arquivo deve ser .zip mesmo! Não vale .tar, .tgz, .bz2, e outros formatos de compactação.

3) Trabalhos copiados receberão nota zero para todas as cópias. Trabalhos com erros de compilação não serão avaliados e receberão nota zero**.** Em qualquer erro deve-se emitir uma mensagem adequada ao erro. O programa deve ser desenvolvido seguindo as boas normas de programação.

II – Tarefas:

1. O operador vírgula (**,**) em Java tem a função de separar expressões no comando **for**. Neste caso, as expressões são avaliadas sequencialmente. Para demonstrar isso, analise o código abaixo e escreva o resultado a ser impresso após sua execução.

```
public class OperadorVirgula {
   public static void main(String[] args) {
      for(int i = 1, j = i + 10; i < 5; i++, j = i * 2) {
         System.out.println("i= " + i + " j= " + j);
      }
   }
}
```

2. O programa abaixo apresenta alguns erros gerais. Explique detalhadamente quais são os erros e como corrigi-los. O programa deve ser executado diretamente da linha de comandos.

```
public class QuestaoErro {
   public static void main (int[] args) {
      double d = 4.5;
      double d2;
      int i = 10;
      char c = (char)64;
```

```
      d2 = d * 10.5;
      i += c;
      int k = i + (int)d * i++;
      do {
         k = i / d2;
         i++;
      } while (i < 10);
      for (int j = 0; j < i; j++) {
         System.out.println("j = "+j);
      }
      System.out.println(j+i);
   }
}
```

3. O programa abaixo recebe valores numéricos passados através dos argumentos de chamada do método *main* e imprime na saída padrão o maior valor fornecido. Explique seu funcionamento e altere-o para imprimir também o menor valor, a soma e a média aritmética dos valores fornecidos.

```
public class MaiorNumero {

    public static void main (String[] args){

       if (args.length > 0) {
           double valor;
           double maior = Double.parseDouble(args[0]);

           for (int i=1; i < args.length; i++) {
               valor = Double.parseDouble(args[i]);
               if (valor > maior)
                   maior = valor;
           }
           System.out.println("Maior valor = " + maior);
       }
       else System.out.println("Entre com os valores na
forma: java MaiorNumero n1  n2 n3 ...");
    }
}
```

4. O código abaixo implementa o método *ShellSort* para ordenação de números inteiros. Ele demonstra o uso de arranjos como argumentos de um método. Lembre-se que um arranjo contém uma referência para uma estrutura, assim, o método pode mudar os elementos do arranjo (passagem de parâmetro por referência). Altere o código abaixo para implementar a ordenação de números inteiros usando o método *QuickSort*.

```
/**
   Ordena um arranjo de números inteiros usando o método
ShellSort
   @version 1.20 27/03/1998
   @author Cay Horstmann
*/

public class ShellSort
{
  public static void sort(int[] a)
   {  int n = a.length;
      int incr = n / 2;
      while (incr >= 1)
      {  for (int i = incr; i < n; i++)
         {  int temp = a[i];
            int j = i;
            while (j >= incr && temp < a[j - incr])
            {  a[j] = a[j - incr];
               j -= incr;
            }
            a[j] = temp;
         }
         incr /= 2;
      }
   }

   public static void print(int[] a)
   {  for (int i = 0; i < a.length; i++)
         System.out.print(a[i] + " ");
      System.out.println();
   }

   public static void main(String[] args)
   {  // cria um arranjo de dez números inteiros
      int[] a = new int[10];
      int i;
      // preenche o arranjo com valores aleatórios
      for (i = 0; i < a.length; i++)
         a[i] = (int)(Math.random() * 100);
      print(a);
      sort(a);
      print(a);
   }

}
```

5. Mostre o resultado impresso pelo programa abaixo, considerando sua chamada de execução com os seguintes parâmetros: **java Impressao 10 2.5 12 20.4**

```
public class Impressao {
   public static void main (String[] args) {
      String[] args2 = args;
      args2[3] = "30.8";
      double[] d = {2, 4, 4.5};
      double[] d2 = new double[d.length];
      int i;
      for (i = 0; i < d.length; i += (int)d[i]) {
         d[i] = Double.parseDouble(args[i+1]);
      }
      i = 0;
      while (i < d.length-1) {
         d2[i] = d[i] + Double.parseDouble(args[i]);
         i++;
      }
      d2[d.length-1] = d[d.length-1];
      for (i = 0; i < args2.length; i++)
        System.out.print(args2[i]+" ");
      System.out.println();
      for (i = 0; i < d.length; i++)
        System.out.print(d[i]+" ");
      System.out.println();
      for (i = 0; i < d2.length; i++)
        System.out.print(d2[i]+" ");
      System.out.println();
      switch ((int)d[0]) {
         case 3:  System.out.println("string 3"); break;
         case 4:  System.out.println("string 4"); break;
         case 20: System.out.println("string 20"); break;
         default: System.out.println("default"); break;
      }
   }
}
```

6. Desenvolva um método estático em Java que recebe como parâmetros um arranjo de números inteiros e um número inteiro a ser pesquisado dentro do arranjo. O método deve retornar “True” se o número pesquisado se encontra no arranjo e “False”, caso contrário. Para testá-lo, crie um método main que execute os seguintes passos:
   - Cria e inicializa um arranjo de 100 números inteiros aleatórios variando de 100 a 199.
   - Leia da entrada padrão vários números a serem pesquisados. Para cada um deles imprima um resultado dizendo se o número se encontra ou não no arranjo.
   - Imprima o arranjo de números inteiros criado.

7. Faça um programa em Java que inicialize um vetor de 10 elementos inteiros com valores aleatórios entre 12 e 21. Em seguida, elimine todos os elementos repetidos e imprima apenas os elementos restantes.

- Exemplo:

Vetor gerado

12	14	18	14	12	13	14	14	21	17

Vetor resultante

12	14	18	13	21	17	?	?	?	?

O programa deve possuir três métodos estáticos:

- **`public static int eliminaRepetidos (int[] vet)`** – elimina os elementos repetidos do vetor *vet*. Retorna o número de elementos restantes no vetor após a eliminação.
- **`public static void imprime (int[] vet, int numElem)`** – imprime na saída padrão os elementos do vetor de inteiros *vet*. *NumElem* é o número de elementos do vetor.
- **`public static void main (String[] args)`** – cria e inicializa o vetor, elimina os elementos repetidos através da chamada do método **`eliminaRepetidos`** e imprime o vetor resultante usando o método **`imprime`**.

8. Faça um programa em Java que inicialize uma matriz de 10 linhas e 10 colunas de elementos inteiros com valores aleatórios entre 0 e 4. Em seguida, imprima os elementos da matriz e uma mensagem informando se a matriz é ou não simétrica. Uma matriz é simétrica se for quadrada (o número de linhas é igual ao número de colunas) e todos os elementos das posições **i**, **j** são iguais aos elementos das posições **j**, **i**, onde **i** representa o número da linha e **j** representa o número da coluna.

   O programa deve possuir três métodos estáticos:

   - Método main, para criar, inicializar e imprimir a matriz, e imprimir uma mensagem informando se a matriz é ou não simétrica. O método main deve usar os dois métodos definidos abaixo.
   - Método para imprimir uma matriz. Este método deve receber como parâmetro a matriz a ser impressa.
   - Método para verificar se uma matriz é simétrica. Este método deve receber como parâmetro a matriz a ser verificada e retornar *true* se a matriz é simétrica e *false*, caso contrário. O método não deve imprimir nenhuma mensagem.

9. Mostre qual o resultado será impresso pelo programa se ele for executado.

```
public class Parametro {
   public static void main(String[] args) {
      int[] v = new int[4];
      v[0] = (int)5.25;
      v[1] = 1;
      v[2] = 2;
      v[3] = 3;
      troca1(v[0],v[1]);
      imprime(v);
      troca2(v);
      imprime(v);
   }
   public static void troca1(int a, int b) {
      int aux = a;
      a = b;
      b = aux;
   }
   public static void troca2(int[] v) {
       int aux;
       for (int i=0; i<v.length/2; i++) {
          aux = v[i];
          v[i] = v[v.length-i-1];
          v[v.length-i-1] = aux;
       }
   }
   public static void imprime(int[] v) {
      for(int i=0; i<v.length; i++)
          System.out.print(v[i]+" ");
   }
}
```