Notas Práticas e Exercícios

Disciplina: Programação Orientada a Objetos

{*Anotações para uso pessoal}

1 Fundamentos

1.1 Hello World e a função `main`

Para rodar no terminal:

- compilar: `javac HelloWorldApp.java`
- rodar: `java HelloWorldApp`

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

Em Java, a função `main` é o ponto de entrada de qualquer programa. Ela é o primeiro método a ser executado quando um programa Java é iniciado. A função `main` deve ser definida em uma classe e deve ter o seguinte cabeçalho:

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

A leitura de uma string pode ser feita utilizando a classe `Scanner` ou a classe `BufferedReader`. Aqui está um exemplo de como ler uma string utilizando a classe Scanner:

```
1  import java.util.Scanner;
2  public class Exemplo {
3      public static void main(String[] args) {
4          Scanner scanner = new Scanner(System.in);
5          System.out.print("Digite uma string: ");
6          String str = scanner.nextLine();
7          System.out.println("A string digitada foi: " + str);
8          scanner.close();
9      }
10 }
```

Uma explicação linha por linha do que está acontecendo:

- A primeira linha importa a classe Scanner do pacote java.util.
- A segunda linha começa a definição da classe Exemplo, que é o nome do arquivo.

- A terceira linha define o método main que é o ponto de entrada do programa. É aqui que a execução do programa começa.
- Na quarta linha, criamos um objeto Scanner chamado scanner e o conectamos à entrada padrão do sistema (System.in). Isso nos permitirá ler a entrada do usuário através do console.
- A quinta linha exibe uma mensagem no console solicitando ao usuário que digite uma string.
- A sexta linha lê a string digitada pelo usuário e a armazena na variável str.
- A sétima linha exibe a string digitada pelo usuário no console, juntamente com a mensagem "A string digitada foi: ".
- A oitava linha fecha o objeto Scanner.

1.2 Multiplicação de Matrizes

A multiplicação de matrizes é uma operação matemática que produz uma nova matriz a partir da combinação das linhas e colunas de duas matrizes pré-existentes. Para multiplicar duas matrizes A e B, é necessário que o número de colunas da matriz A seja igual ao número de linhas da matriz B. O resultado da multiplicação é uma matriz C, onde cada elemento C(i,j) é obtido a partir do produto interno da linha i da matriz A pela coluna j da matriz B.

```
1  import java.util.Scanner;
2
3  public class MatrizMultiplicacao {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7          // Leitura do tamanho das matrizes
8          System.out.println("Digite o número de linhas da primeira matriz:");
9          int linhas1 = scanner.nextInt();
10         System.out.println("Digite o número de colunas da primeira
               matriz:");
11         int colunas1 = scanner.nextInt();
12         System.out.println("Digite o número de linhas da segunda matriz:");
13         int linhas2 = scanner.nextInt();
14         System.out.println("Digite o número de colunas da segunda matriz:");
15         int colunas2 = scanner.nextInt();
16
17         // Verifica se as matrizes são compatíveis para multiplicação
18         if (colunas1 != linhas2) {
19             System.out.println("As matrizes não são compatíveis para
                   multiplicação.");
20             return;
21         }
22
23         // Leitura dos elementos da primeira matriz
24         System.out.println("Digite os elementos da primeira matriz:");
25         int[][] matriz1 = new int[linhas1][colunas1];
26         for (int i = 0; i < linhas1; i++) {
27             for (int j = 0; j < colunas1; j++) {
28                 matriz1[i][j] = scanner.nextInt();
29             }
```

```
30          }
31
32          // Leitura dos elementos da segunda matriz
33          System.out.println("Digite os elementos da segunda matriz:");
34          int[][] matriz2 = new int[linhas2][colunas2];
35          for (int i = 0; i < linhas2; i++) {
36              for (int j = 0; j < colunas2; j++) {
37                  matriz2[i][j] = scanner.nextInt();
38              }
39          }
40
41          // Multiplicação das matrizes
42          int[][] resultado = new int[linhas1][colunas2];
43          for (int i = 0; i < linhas1; i++) {
44              for (int j = 0; j < colunas2; j++) {
45                  for (int k = 0; k < colunas1; k++) {
46                      resultado[i][j] += matriz1[i][k] * matriz2[k][j];
47                  }
48              }
49          }
50
51          // Exibição da matriz resultado
52          System.out.println("O resultado da multiplicação é:");
53          for (int i = 0; i < linhas1; i++) {
54              for (int j = 0; j < colunas2; j++) {
55                  System.out.print(resultado[i][j] + " ");
56              }
57              System.out.println();
58          }
59      }
60  }
```

1.3 Atividade `instanceof`

A hierarquia das classes Java segue a seguinte ordem:

- Object: é a classe raiz da hierarquia de classes Java e todas as outras classes herdam dela.
- Number: é a classe pai das classes que representam valores numéricos, como Integer, Long e Double.
- Integer: representa um valor inteiro.
- Long: representa um valor inteiro maior do que o Integer.
- Double: representa um valor decimal de ponto flutuante.
- Boolean: representa um valor booleano (verdadeiro ou falso).

Vale ressaltar que existem muitas outras classes na hierarquia de classes Java que não foram mencionadas nesta lista, incluindo classes para lidar com strings, datas, coleções, entre outras. Obs. usar Integer.valueOf(123); para instanciar.

```java
public class Exemplo {

    public static void main(String[] args) {
        Integer intObj = new Integer(123);
        Integer intObj1 = Integer.valueOf(123);
        Long longObj = new Long(1234567890);
        Double doubleObj = new Double(12.345);
        Boolean boolObj = new Boolean(true);
        Object[] objArray = {intObj, longObj, doubleObj, boolObj};
        for (int i = 0; i < objArray.length; i++) {
            if (objArray[i] instanceof Number) {
                System.out.println(objArray[i].toString() + " é um
                    Number.");
            } else {
                System.out.println(objArray[i].toString() + " NÃO é um
                    Number.");
            }
        }
    }
}
```

1.4 Palindromo

Um palíndromo é uma palavra, frase, número ou qualquer sequência de caracteres que pode ser lida da mesma maneira da esquerda para a direita ou da direita para a esquerda. Por exemplo, 'arara'.

No código abaixo, a entrada do usuário é lida como uma string e os espaços em branco são removidos e as letras são convertidas para minúsculas. Em seguida, a string original é invertida e comparada com a string original para verificar se é um palíndromo. A saída do programa informa se a string é um palíndromo ou não.

```java
import java.util.Scanner; // Importa a classe Scanner para ler a entrada do
    usuário

public class Palindromo {
    public static void main(String[] args) {
        // Cria um objeto Scanner para ler a entrada do usuário
        Scanner sc = new Scanner(System.in);

        // Pede para o usuário digitar uma string
        System.out.print("Digite uma string: ");
        String str = sc.nextLine();

        // Remove os espaços em branco da string e converte para letras
            minúsculas
        str = str.replaceAll("\\s", "").toLowerCase();

        // Cria uma string vazia para armazenar a string invertida
        String invertida = "";

        // Inverte a string original
        for (int i = str.length() - 1; i >= 0; i--) {
```

```
            invertida += str.charAt(i);
        }

        // Verifica se a string original é igual à string invertida
        if (str.equals(invertida)) {
            System.out.println("A string é um palíndromo.");
        } else {
            System.out.println("A string não é um palíndromo.");
        }
    }
}
```

1.5 Criando uma classe Retangulo

Uma classe é uma estrutura/modelo básico para definir objetos e seus comportamentos. Uma classe é composta por campos (variáveis) e métodos (funções), que podem ser acessados e usados para criar instâncias da classe. As instâncias de uma classe em Java são objetos criados a partir dessa classe. Cada instância representa uma 'cópia' da classe, com seus próprios valores para os campos da classe. Você pode criar várias instâncias de uma mesma classe, cada uma com seus próprios valores para os campos. Por exemplo, se você tiver uma classe Pessoa com campos para nome e idade, você pode criar várias instâncias dessa classe para representar diferentes pessoas, cada uma com seu próprio nome e idade. Para criar uma instância de uma classe em Java, você precisa usar a palavra-chave new, seguida pelo nome da classe e parênteses (com ou sem argumentos dependendo do construtuor).

O código abaixo define uma classe `Retangulo` (que estará em um arquivo `Retangulo.java`) em Java que tem duas variáveis de instância privadas (`largura e altura`) e métodos públicos para calcular a área e o perímetro do retângulo, bem como métodos `get e set` para acessar e modificar as variáveis de instância.

O construtor da classe `Retangulo` recebe dois argumentos (`largura e altura`) e usa o operador this para atribuir esses valores às variáveis de instância correspondentes. O método `calcularArea()` simplesmente retorna o produto da largura e altura do retângulo. O método `calcularPerimetro()` retorna o perímetro do retângulo, que é o dobro da soma da largura e altura.

Os métodos `get` e `set` permitem que outros objetos acessem e modifiquem as variáveis de instância privadas largura e altura. O método `getLargura()` retorna o valor atual da largura e o método setLargura() permite que a largura seja modificada. O mesmo vale para o método `getAltura()` e `setAltura()`.

```
public class Retangulo {
    private double largura;
    private double altura;

    public Retangulo(double largura, double altura) {
        this.largura = largura;
        this.altura = altura;
    }

    public double calcularArea() {
        return largura * altura;
    }

    public double calcularPerimetro() {
        return 2 * (largura + altura);
    }

```

```
18      public double getLargura() {
19          return largura;
20      }
21
22      public void setLargura(double largura) {
23          this.largura = largura;
24      }
25
26      public double getAltura() {
27          return altura;
28      }
29
30      public void setAltura(double altura) {
31          this.altura = altura;
32      }
33  }
```

Agora você pode criar objetos do tipo `Retangulo` e usar os métodos do objeto para calcular a área e o perímetro. Também é possível acessar e modificar os atributos do objeto usando os métodos get e set.

```
1   public class Exemplo {
2
3       public static void main(String[] args) {
4           Retangulo retangulo = new Retangulo(10, 5);
5           double area = retangulo.calcularArea();
6           double perimetro = retangulo.calcularPerimetro();
7           double largura = retangulo.getLargura();
8           retangulo.setLargura(20);
9
10      }
11
12  }
```

2 Exercícios Práticos

1. Escreva um programa que leia três inteiros e imprima a soma, a média, o produto e o maior e o menor elemento lido.

2. Escreva um programa que leia o raio de um círculo e imprima o diâmetro, área e circunferência do círculo. Os resultados devem ser escritos com duas casas decimais.

3. Crie um programa que leia dois números inteiros e determine se um é múltiplo do outro.

4. Escreva um programa que lê um caractere e retorna seu código na tabela ASCII (um número inteiro).

5. Escreva um programa que lê 15 números double e através de uma função determine quais são os dois maiores deles, que não devem ser repetidos.

6. Uma grande companhia paga seus vendedores com salários fixos mais comissões sobre as vendas. Um vendedor recebe R$200,00 mais 9% de comissão sobre todas as suas vendas. Escreva um programa que lê o valor total das vendas de um vendedor e calcule seu salário. Seu programa deve computar o salário de diversos vendedores, e parar quando o valor informado for -1.

7. Escreva um programa que estima o valor da constante e pela fórmula (o usuário deve informar o número de termos deste somatório, que define a precisão do resultado):

$$e = (1/1!) + (1/2!) + (1/3!) + ... \tag{1}$$

8. Crie um programa que leia 12 números inteiros em qualquer ordem e imprima uma listagem contendo o número lido e uma mensagem "PAR" ou "ÍMPAR" conforme o número.

9. Crie um programa que armazene 12 números em um array e determine qual a porcentagem de números menores que 8 e qual a porcentagem de números maiores que 10.

10. **Quadrado Mágico.** Um Quadrado Mágico é uma tabela quadrada de lado N, onde a soma dos números das linhas, das colunas e das diagonais principal e secundária é constante, sendo que nenhum destes números se repete. Crie um programa para verificar (não confunda com resolver) se uma matriz realmente é um quadrado mágico. Para isso primeiro o usuário entra com o tamanho da matriz e depois digita os valores. Um exemplo de quadrado mágico:

8	1	6
3	5	7
4	9	2

11. Escreva um programa que leia uma string do usuário e verifique se ela contém apenas caracteres únicos.

12. Após implementar e estudar o exemplo da Seção 1.5,faça:

    - Escreva uma classe Quadrado que tenha uma variável de instância para representar o lado do quadrado. Adicione métodos para calcular a área e o perímetro do quadrado.
    - Escreva uma classe Círculo que tenha uma variável de instância para representar o raio do círculo. Adicione métodos para calcular a área e a circunferência do círculo.
    - Escreva uma classe Triângulo que tenha três variáveis de instância para representar os lados do triângulo. Adicione métodos para calcular a área e o perímetro do triângulo.
    - Escreva um programa que crie objetos do tipo Quadrado, Círculo e Triângulo e imprima suas áreas e perímetros.

13. Crie uma classe Conta que tenha os seguintes campos privados: Nome; CPF (pode usar long); Saldo R$; Número da conta; Agência. Para cada um dos campos disponibilize um método set e get. Faça um método para depósito (que recebe um valor) e outro método para saque da conta (que recebe um valor e retorna ele).