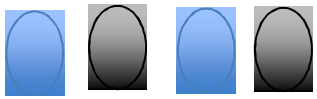
CIÊNCIAS DA COMPUTAÇÃO

FACULDADE DE CIÊNCIAS | UNIVERSIDADE AGOSTINHO NETO



1º Ano

FUNDAMENTOS DE PROGRAMAÇÃO

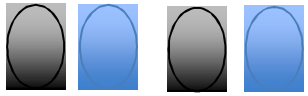
Docente:

- ✓ Mateus Padoca Calado - PhD

Monitores:

- ✓ Nsimba Kiafuka
- ✓ Anacleto Mimoso
- ✓ Mariano Calelua

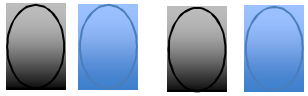
Conteúdo



Tema - 04

- Métodos
- Recursividade
- Tratamento de Excepção

Conteúdo



Cap. VII

- Métodos
- Recursividade
- Tratamento de Excepção

CAP. VII - Introdução

- Métodos são sub-rotinas que executam um conjunto de instruções específico, ou seja métodos definem o comportamento de uma class(def. OO).

- Os métodos podem ser:
  - Função: Devolvem explicitamente um resultado ao exterior.

  - Procedimentos: Não devolvem explicitamente um resultado ao exterior.

CAP. VII - Função

❑ Função

SINTAXE

```
public static tipoRetorno nomeFucao(parametros){

    <Instruções>

    return valorRetorno;
}
```

- tipoRetorno: define o tipo de dados de retorno da função(podendo ser tipo primitivo ou por referência).
- nomeFuncao: define o nome da função a ser criada.
- parametros: declaração das variaveis passada na função separadas por vírgula ex: (int a, float b, String nome).
- valorRetorno: especifica o valor a ser retorna para exterior e deve ser do mesmo tipo que o tipoRetorno.

CAP. VII - Função

- Função que recebe dois numeros inteiro e calcula a soma do dois números

Exemplo

```
import java.util.Scanner;
public class Exemplo {
    /* Declaração da função */
    public static int soma(int a, int b) {
        int c = a + b;
        return c;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Digite os dois números: ");
        int x = input.nextInt();
        int y = input.nextInt();
        int z = soma(x, y);
        System.out.println("SOMA = "+ z);
    }
}
```

CAP. VII - Procedimento

❑ Procedimento

SINTAXE

```
public static void nomeProcedimento(parametros){

    <Instruções>

}
```

- nomeProcedimento: define o nome da função a ser criada.
- parametros: declaração das variaveis passada no procedimento separadas por vírgula ex: (int a, float b, String nome).

CAP. VII - Procedimento

- Procedimento que recebe um número e informa se o número é par ou impar.

Exemplo

```
import java.util.Scanner;
public class Exemplo {
    public static void informarNumero(int numero) {
        if (numero % 2 == 0) {
            if (numero != 0) {
                System.out.println("O Número é par");
            } else {
                System.out.println("Valor Nulo.");
            }
        } else {
            System.out.println("O Número é Impar");
        }
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Digite um número: ");
        int x = input.nextInt();
        informarNumero(x);
    }}
```

CAP. VII – Array Como Parâmetro

- Para passar um argumento de array para um método especifica-se somente o nome do array sem nenhum colchetes.

```
int [] numeros = new int [5];
imprimir(numeros); /*passar um array para o método */
```

- Para um método receber um array como parâmetro, deve especificar um parâmetro de array na sua lista de parâmetro.

```
public void imprimir(int [] n){

    for(int i = 0; i < n.length; i++){
        System.out.println(n[i]);
    }
}
```

CAP. VII – Array Como Parâmetro

- Função que recebe como parâmetro um vector de nomes e uma letra e retorna o primeiro nome existente no vector que inicia com a respectiva letra.

Resolução

```
public String procurarNome(String[] nomes, char letra) {

    for (int i = 0; i < nomes.length; i++) {

        if (letra == nomes[i].charAt(0)) {
            return nomes[i];
        }
    }

    return "";
}
```

CAP. VII – ArrayList Como Parâmetro

- Função que recebe como parâmetro uma lista de nomes e uma letra e retorna o primeiro nome existente na lista que inicia com a respectiva letra.

Exemplo

```
public static String procurarNome(ArrayList<String> nomes, char letra) {

    for(String nome : nomes) {
        if (letra == nome.charAt(0)) {
            return nome;
        }
    }

    return "";
}
```

CAP. VII - Escopo

- O escopo de uma variável, representa a área do programa onde esta é visível podendo ser: global ou local.
- Em Java as variaveis locais têm maior precedência.

Exemplo

```
public class Exemplo {
    static int x = 6; /* variavel global */

    public static void imprimir() {
        int y = 0; /* variavel local */
        y = x + 2;
        System.out.println(y);
    }

    public static void main(String[] args) {
        imprimir();
    }
}
```

CAP. VII - Observações

- Em Java não há indicação explícita de passagem por referência.

- Argumentos são sempre passados por valor (ainda que o valor seja uma referência):
  - Tipos primitivos: parâmetro é uma cópia do valor do argumento
  - Outros tipos: parâmetro é uma cópia da referência (i.e. aponta para os mesmos dados que o argumento).

CAP. VII - Recursividade

- Recursividade é a possibilidade que um método tem de chamar a si próprio repetidamente até que uma condição seja satisfeita.

- Exemplo
  - Função que retorna o factorial de um número.

Exemplo

```
public static int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return factorial(n-1) * n;
    }
}
```

CAP. VII – Tratamento de Excepção

- Uma excepção é uma indicação de um problema que ocorre durante a execução de um programa.
- O tratamento de excepção garante que um programa continua com sua execução depois de lidar com o problema encontrado.

SINTAXE

```
try{

    /* Tarefas a executar */

}catch(TipoDeExcepcao variavel){
    /* Acção a tomar caso haja erros nas tarefas*/
}
```

- TipoDeExcepcao: tipo de erro que será lançado, os mais comum são:
  - ArithmeticException: Quando ocorre um erro de aritmética.(java.util)
  - InputMismatchException : Quando ocorre um erro durante a leitura de dados.(java.lang)

CAP. VII – Tratamento de Excepção

- Ex: Programa lê um número inteiro e mostra na tela

Exemplo

```
public class Exemplo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        try {
            System.out.print("Digite um número inteiro: ");
            int x = input.nextInt();
            System.out.println(x);
        } catch (java.util.InputMismatchException e) {
            System.out.println(“Valor Inválido");
        }
    }
}
```

- Um bloco try pode possuir mais de um bloco catch

CAP. VII. Métodos- Exercícios

1. Crie um procedimento que recebe como parâmetro um vectores de inteiros e imprimi na tela todos os números impar múltiplo de três existente no vector.

2. Implemente a função **public static String qtdLetra(String nome, char letra)** que retorna o números de vezes que uma letra existe no nome.

3. Crie a função **public static int getPosicao(String [] nomes, String nome)** que retorna a posição do nome no vector caso não existir retorna -1.

4. Crie o procedimento public static void regressiva(int n) que recebe como parâmetro um número inteiro e imprimi todos os números inteiro positivo menor que o mesmo.