UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

AULA 12

LINGUAGEM DE PROGRAMAÇÃO 2
JAVA

PROF. JANIHERYSON FELIPE

CONTEÚDO DESSA AULA

- CONHECER A IMPRESSÃO FORMATADA
- CONHECER AS CLASSES EXTERNAS E INTERNAS;
- ENTENDER O CONCEITO DE ANNOTATIONS;
- CONHECER AS CLASSES E MÉTODOS GENERICAS;
- DISCUSSÕES E DÚVIDAS GERAIS.

IMPRESSÃO FORMATADA (PRINTF)

- O método printf() em Java é usado para imprimir uma **string formatada** no console ou em qualquer fluxo de saída especificado. Ele permite formatar a string de saída com base em um padrão específico, semelhante ao que é feito com a função printf() em linguagens como C.

```
System.out.printf("formato string", arg1, arg2, arg3);
```

```
System.out.printf("Nome: %s, Idade: %d, Altura: %.2f metros%n", nome, idade, altura);
```

IMPRESSÃO FORMATADA (PRINTF)

- “%s” e “%S” - Recebe uma string
- “%c” e “%C” - Recebe um character
- “%d” - Recebe um inteiro
- “%.2f” - Recebe um float com duas casas decimais
- “%n” - Quebra uma linha
- “%20s” - Faz que a string ocupe 20 espaços a direita
- “%-20s” - Faz que a string ocupe 20 espaços a esquerda

IMPRESSÃO FORMATADA (PRINTF)

- “%+d” e “% d” Imprime um inteiro com seu sinal
- “%015d” - Insere zeros na frente do numero para que ele fique com 15 elementos de tamanho;
- “%.d” - Separa as unidade de centena, dezena, etc. (ponto/vírgula)
- “R$%10.2f” - Dez espaços, com duas casas decimais.

CLASSES EXTERNAS E INTERNAS

```
class Externa{
    String texto = "Classe Externa";

    class Interna{
        String texto = "Classe Interna";

        public void imprimemensagem(){
            System.out.println(texto);
        }
    }
}
```

CLASSES EXTERNAS E INTERNAS

```
class Main{
    Run | Debug
    public static void main(String[] args) {
        Externa externa = new Externa();
        Externa.Interna interna = externa.new Interna();

        interna.imprimemensagem();
    }
}
```

ANNOTATIONS - ANOTAÇÕES EM JAVA

Annotations são marcadores que fornecem metadados sobre classes, métodos, variáveis e outros elementos do código-fonte. Elas não afetam diretamente o comportamento do código, mas são utilizadas pelo compilador, pelo ambiente de execução ou por frameworks para fornecer informações adicionais ou instruções sobre como o código deve ser processado.

ANNOTATIONS - ANOTAÇÕES EM JAVA

As annotations são declaradas precedendo o elemento que elas estão marcando com o símbolo "@" seguido do nome da annotation. Elas podem ou não ter atributos associados, dependendo da annotation específica.

Java fornece algumas annotations embutidas, como **@Override**, **@Deprecated** e **@SuppressWarnings**, que são amplamente utilizadas.

ANNOTATIONS - ANOTAÇÕES EM JAVA

```java
@interface MinhaAnnotation {
    String nome() default "Indefinido";
    String cpf() default "Indefinido";
    String email();
}
```

```java
@MinhaAnnotation(email = "felipe@gmail", nome = "Felipe")
Run | Debug
public static void main(String[] args) {
```

CLASSES E MÉTODOS GENÉRICOS

Classes genéricas em Java são classes que permitem a **especificação de tipos** de dados variáveis, conhecidos como parâmetros de tipo, que são fornecidos no momento da criação de instâncias da classe. Isso permite que você crie classes e métodos que possam funcionar com **qualquer tipo** de dados, fornecendo flexibilidade e reutilização de código.

```java
class Generica<T>{
    private T valor;


    public T getValor() {
        return valor;
    }

    public void setValor(T valor) {
        this.valor = valor;
    }


}
```

CLASSES E MÉTODOS GENÉRICOS

```
public class Main {
    Run | Debug
    public static void main(String[] args) {

        Generica<String> gene = new Generica<>();
        gene.setValor(valor:"Qualquer valor");
        System.out.println(gene.getValor());
    }
}
```

CLASSES E MÉTODOS GENÉRICOS

```java
class Generica<T>{
    private T valor;

    public T getValor() {
        return valor;
    }

    public void setValor(T valor) {
        this.valor = valor;
    }

     public Double operar() {
        if (valor instanceof Integer) {
            Integer valorInteiro = (Integer) valor;
            return Math.pow(valorInteiro, b:2);
        } else if (valor instanceof Double) {
            Double valorDouble = (Double) valor;
            return Math.pow(valorDouble, b:2);
        }
        return null;
    }
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Generica<Integer> genericaInteiro = new Generica<>();
        genericaInteiro.setValor(valor:5);
        System.out.println("Integer: " + genericaInteiro.operar());

        Generica<Double> genericaDouble = new Generica<>();
        genericaDouble.setValor(valor:3.5);
        System.out.println("Double: " + genericaDouble.operar());
    }
}
```

IMPORTAÇÃO ESTÁTICA

Permite que você acesse membros estáticos de uma classe diretamente, sem precisar qualificar seu nome com o nome da classe. Isso é especialmente útil quando você precisa usar constantes ou métodos estáticos de uma classe frequentemente e quer tornar o código mais limpo e legível.

```java
import static java.lang.Math.*;
public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.println(pow(a:10, b:2));
        System.out.println(sqrt(a:144));
        System.out.println(PI);

    }
}
```

GARBAGE COLLECTOR

O coletor de lixo (Garbage Collector) em Java é um sistema de gerenciamento automático de memória que tem como objetivo liberar a memória de objetos que não são mais acessíveis ou necessários pelo programa.

FASES:

- Fase de Marcação (Marking)
- Fase de Limpeza (Sweeping
- Fase de Compactação (Compacting)

```
Runtime.getRuntime().gc();
```