Programação Orientada a Objetos I

CÁSSIO CAPUCHO PEÇANHA – 03

Impressão/Formatação

Impressão/Formatação

- No Java, você pode exibir informações no console usando diferentes métodos da classe System.out.
- System.out.println()
- System.out.print()
- System.out.printf()
- System.err.println()

Impressão/Formatação

- System.out.println()
  - Imprime um texto e pula para a próxima linha.

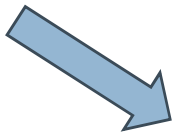
```
System.out.println("Olá, mundo!");
System.out.println("Outra linha.");
```

```
Olá, mundo!
Outra linha.
```

Impressão/Formatação

- System.out.print()
  - Imprime o texto, mas não pula para a próxima linha.

```
System.out.print("Olá, ");
System.out.print("mundo!");
```

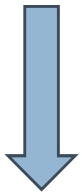


```
Olá, mundo!
```

Impressão/Formatação

- System.out.printf()
  - Permite **formatar** a saída com placeholders (%d, %s, %.2f, etc.).

```
System.out.printf("O valor de PI é aproximadamente %.2f%n", 3.14159);
```



```
O valor de PI é aproximadamente 3.14
```

Impressão/Formatação

- System.err.println()
  - Exibe mensagens de erro no console (em vermelho em alguns terminais).

```
System.err.println("Erro: Algo deu errado!");
```

```
Erro: Algo deu errado!
```

Placeholders

Placeholder	Tipo de Dado	Descrição	Exemplo	Saída
%d	Inteiro	Representa números inteiros	System.out.printf("Idade: %d anos", 25);	Idade: 25 anos
%f	Ponto flutuante	Representa números decimais (float/double)	System.out.printf("Altura: %f metros", 1.75);	Altura: 1.750000 metros
%.2f	Ponto flutuante	Limita o número de casas decimais	System.out.printf("Preço: %.2f", 9.99);	Preço: 9.99
%s	String	Insere texto	System.out.printf("Nome: %s", "Carlos");	Nome: Carlos
%c	Caractere	Representa um único caractere	System.out.printf("Letra: %c", 'A');	Letra: A
%b	Booleano	Insere valores true ou false	System.out.printf("Status: %b", true);	Status: true
%n	Nova linha	Quebra de linha (equivalente a \n)	System.out.printf("Linha 1%nLinha 2");	Linha 1 Linha 2
%%	Porcentagem	Insere o caractere %	System.out.printf("Desconto: 10%%");	Desconto: 10%

Entrada de Dados

Entrada de Dados – Tipos Primitivos

Método	Retorno	Descrição
nextByte()	byte	Lê um número do tipo byte (-128 a 127).
nextShort()	short	Lê um número do tipo short (-32.768 a 32.767).
nextInt()	int	Lê um número inteiro (int).
nextLong()	long	Lê um número inteiro grande (long).
nextFloat()	float	Lê um número decimal (float).
nextDouble()	double	Lê um número decimal maior (double).
nextBoolean()	boolean	Lê um valor true ou false.

Entrada de Dados – Leitura de Texto

Método	Retorno	Descrição
next()	String	Lê **uma única palavra** (até encontrar espaço ou quebra de linha).
nextLine()	String	Lê **uma linha inteira** (até encontrar um \n).
next().charAt(0)	char	Lê **o primeiro caractere** digitado (não existe nextChar()).

Limpeza do buffer → Após nextInt(), nextDouble(), etc., **use scanner.nextLine()** antes de ler uma String, para evitar problemas de buffer.

Fluxos de Controle

Fluxos de Controle

- Fluxos de controle em Java são estruturas que permitem ao programador controlar a execução do programa, determinando a sequência de instruções que serão executadas de acordo com certas condições.
- Essas estruturas permitem que o programa tome decisões com base em condições específicas ou execute repetidamente um bloco de código enquanto uma determinada condição for verdadeira.

Fluxos de Controle

- Repetições (loops):
  - while, do-while , for
- Condicionais:
  - if-else, switch-case
- Desvios:
  - break, continue, label:, return
- Manipulação de exceções:
  - try-catch-finally, throw

Fluxos de Controle - Condicionais

If-else

- A estrutura if-else permite tomar decisões com base em uma condição.
- Se a condição especificada for verdadeira, o bloco de código associado ao if é executado.
- Caso contrário, o bloco associado ao else é executado.

java

```java
int idade = 25;
if (idade >= 18) {
    System.out.println("Você é maior de idade.");
} else {
    System.out.println("Você é menor de idade.");
}
```

Fluxos de Controle - Condicionais

switch-case

- O switch-case é uma estrutura de decisão que permite selecionar uma ação com base no valor de uma expressão.
- Se o valor da expressão corresponder a um dos casos definidos, o bloco de código associado a esse caso é executado.
- É uma alternativa mais limpa e legível para if-else quando há muitas opções a serem avaliadas.

java

```java
int diaSemana = 3;
switch (diaSemana) {
    case 1:
        System.out.println("Domingo");
        break;
    case 2:
        System.out.println("Segunda-feira");
        break;
    // ...
    default:
        System.out.println("Dia inválido");
        break;
}
```

Fluxos de Controle - Repetições

while

- O loop while é uma estrutura de repetição que executa um bloco de código enquanto uma condição especificada for verdadeira.
- Antes de cada iteração, a condição é verificada.
- Se for verdadeira, o bloco de código é executado; caso contrário, o loop é encerrado.

```java
int contador = 1;
while (contador <= 5) {
    System.out.println("Contador: " + contador);
    contador++;
}
```

Fluxos de Controle - Repetições

do-while

- O loop do-while é semelhante ao while, mas a verificação da condição é feita após a execução do bloco de código.
- Isso significa que o bloco de código é executado pelo menos uma vez, mesmo que a condição seja falsa desde o início.

```java
int contador = 1;
do {
    System.out.println("Contador: " + contador);
    contador++;
} while (contador <= 5);
```

Fluxos de Controle - Repetições

for

- O loop for é uma estrutura de repetição mais compacta, que combina a inicialização de uma variável de controle, a verificação da condição e o incremento da variável em uma única linha.

```java
for (int i = 1; i <= 5; i++) {
    System.out.println("Contador: " + i);
}
```

Fluxos de Controle - Desvios

- break: A palavra-chave break é usada para sair de uma estrutura de controle, como um loop for, while, do-while ou um switch-case. Quando break é executado, o controle é transferido para fora do bloco de controle.
- continue: A palavra-chave continue é usada dentro de um loop para pular o restante do bloco de código na iteração atual e passar para a próxima iteração.
- label: Labels são marcadores usados para identificar loops ou blocos de código específicos. Eles são usados em conjunto com break e continue para sair ou pular iterações em loops aninhados.
- return: A palavra-chave return é usada em métodos para retornar um valor e terminar a execução do método. Ela também pode ser usada para encerrar a execução de um loop ou método antes do esperado.

Fluxos de Controle - Exceções

- try-catch-finally:

- A estrutura try-catch-finally é usada para capturar e manipular exceções (erros) que podem ocorrer durante a execução do programa.
- O bloco try contém o código que pode gerar uma exceção. Se uma exceção é lançada, o bloco catch captura a exceção e trata o erro.
- O bloco finally é opcional e é usado para executar código que sempre deve ser executado, independentemente de ter ocorrido uma exceção ou não.

java

```java
try {
    int resultado = 10 / 0; // Lança uma ArithmeticException
} catch (ArithmeticException e) {
    System.out.println("Erro de divisão por zero: " + e.getMessage());
} finally {
    System.out.println("Fim do programa.");
}
```