5: Estruturas de Controle

5.1 Introdução às Estruturas de Controle

As estruturas de controle são usadas em programação para controlar o fluxo de execução de um programa. Em Java, existem várias estruturas de controle, como `if`, `else`, `else if`, `switch`, `for`, `while` e `do-while`, que permitem que você tome decisões e execute ações com base em condições específicas.

5.2 Estrutura Condicional `if`

A estrutura condicional `if` é usada para executar um bloco de código se uma condição especificada for verdadeira. Aqui está a sintaxe básica do `if`:

```
if (condição) {
    // bloco de código a ser executado se a condição for verdadeira
}
```
```

Por exemplo:

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

5.3 Estrutura Condicional `else`

A estrutura condicional `else` é usada em conjunto com `if` para executar um bloco de código se a condição especificada no `if` for falsa. Aqui está a sintaxe básica do `else`:

```
if (condição) {
    // bloco de código a ser executado se a condição for verdadeira
} else {
    // bloco de código a ser executado se a condição for falsa
}
```

Por exemplo:
```

```java
int idade = 16;
if (idade >= 18) {
    System.out.println("Você é maior de idade.");
} else {
    System.out.println("Você é menor de idade.");
}
```

5.4 Estrutura Condicional `else if`

A estrutura condicional `else if` é usada para verificar múltiplas condições. Ela vem após um bloco `if` e antes de um bloco `else`, e é executada se a condição do `if` anterior for falsa e sua própria condição for verdadeira. Aqui está a sintaxe básica do `else if`:

```java
if (condição1) {
    // bloco de código a ser executado se a condição1 for verdadeira
} else if (condição2) {
    // bloco de código a ser executado se a condição2 for verdadeira
} else {
    // bloco de código a ser executado se todas as condições anteriores forem falsas
}
```

Por exemplo:

```java
int nota = 75;
if (nota >= 90) {
    System.out.println("Nota A");
} else if (nota >= 80) {
    System.out.println("Nota B");
} else if (nota >= 70) {
    System.out.println("Nota C");
} else {
    System.out.println("Nota D");
}
```

5.5 Estrutura de Controle `switch-case`

A estrutura de controle `switch-case` é usada para tomar decisões com base no valor de uma expressão. Ela avalia a expressão e executa o bloco de código associado ao valor correspondente. Aqui está a sintaxe básica do `switch-case`:

```
switch (expressão) {
    case valor1:
        // bloco de código a ser executado se expressão for igual a valor1
        break;
    case valor2:
        // bloco de código a ser executado se expressão for igual a valor2
        break;
    default:
        // bloco de código a ser executado se expressão não corresponder a nenhum dos
}
```

Por exemplo:

```
char opcao = 'A';
switch (opcao) {
    case 'A':
        System.out.println("Opção A selecionada");
        break;
    case 'B':
        System.out.println("Opção B selecionada");
        break;
    default:
        System.out.println("Opção inválida");
}
```

5.6 Loops (Laços de Repetição)

Os loops são estruturas de controle usadas para repetir a execução de um bloco de código várias vezes. Em Java, existem três tipos principais de loops: `for`, `while` e `do-while`.

5.6.1 Loop `for`

O loop `for` é usado quando o número de iterações é conhecido antes do início da execução. Ele consiste em três partes: inicialização, condição e atualização. Aqui está a sintaxe básica do loop `for`:

```
for (inicialização; condição; atualização) {
    // bloco de código a ser repetido
}
```

Por exemplo:

```java
for (int i = 0; i < 5; i++) {
    System.out.println("Número: " + i);
}
```

5.6.2 Loop `while`

O loop `while` é usado quando o número de iterações não é conhecido antes do início da execução. Ele executa o bloco de código repetidamente enquanto uma condição especificada for verdadeira. Aqui está a sintaxe básica do loop `while`:

```java
while (condição) {
    // bloco de código a ser repetido
}
```

Por exemplo:

```java
int i = 0;
while (i < 5) {
    System.out.println("Número: " + i);
    i++;
}
```

5.6.3 Loop `do-while`

O loop `do-while` é semelhante ao loop `while`, mas garante que o bloco de código seja executado pelo menos uma vez, mesmo se a condição for falsa desde o início. Aqui está a sintaxe básica do loop `do-while`:

```java
do {
    // bloco de código a ser repetido
} while (condição);
```

Por exemplo:

```java
int i = 0;
do {
    System.out.println("Número: " + i);
    i++;
} while (i < 5);
```