Lógica

Agenda

- Condições
- Laços
- Exceções

Condições

- Uma estrutura de seleção simples permite a escolha de um grupo de ações e estruturas a ser executado quando determinadas condições, representadas por expressões lógicas, são ou não satisfeitas

```
int idade = leInteiro("Informe sua Idade");

if(idade >= 18) {
    escrevaL("Você é maior de idade");
}
```

Condições

- Na seleção composta, caso o resultado da condição seja falsa, teremos a execução de uma outra seqüência de comandos.

```
int idade = leInteiro("Informe sua Idade");

if(idade >= 18) {
    escrevaL("Você é maior de idade");
} else {
    escrevaL("Você é menor de idade");
}
```

Condições

- Na seleção composta também permite que seja encadeado uma seqüência de testes

```
int idade = leInteiro("Informe sua Idade");

if(idade < 14) {
    escrevaL("Você ainda é uma criança");
} else if(idade < 18) {
    escrevaL("Você é quase maior de idade");
} else {
    escrevaL("Você é maior de idade");
}
```

Condições

- Uma outra forma de tratar um problema de lógica com seleções encadeadas, quando a ação a ser executada depende do valor de uma variável

```
int codigo = leInteiro("Informe o código de acesso");

switch (codigo) {
case 1:
    escrevaL("Vá para o quinto andar");
    break;
case 3:
    escrevaL("Vá para o nono andar");
    break;
default:
    escrevaL("Vá para o primeiro andar");
    break;
}
```

Laço

- Consiste numa estrutura de controle do fluxo lógico que permite executar diversas vezes um mesmo trecho do algoritmo, porém, sempre verificando antes de cada execução se é “permitido” repetir o mesmo trecho.

```
int quantidade = leInteiro("Informe a quantidade de valores");

double total = 0;
int contador = 1;
while (contador <= quantidade) {
    double valor = leReal("Informe o ", contador, "° valor");
    total = total + valor;
    contador = contador + 1;
}
escreval("O valor Total é de R$", total);
```

Laço

- Para realizar a repetição com teste no final, utilizamos a estrutura “repita”, que permite que um bloco ou ação primitiva seja repetida até que uma determinada condição seja verdadeira.

```
double total = 0;
int contador = 1;
char continua = 'N';
do {
    double valor = leReal("Informe o ", contador, "° valor");
    total = total + valor;
    contador = contador + 1;
    continua = leCaracter("Deseja continuar? (informe S ou N");
} while (continua == 'S');
escrevaL("O valor Total é de R$", total);
```

Laço

- A estrutura “para” repete a execução do bloco um número definido de vezes, pois ela possui limites fixos.

```
int quantidade = leInteiro("Informe a quantidade de valores");

double total = 0;
for(int contador = 1; contador <= quantidade; contador++) {
    double valor = leReal("Informe o ", contador, "° valor");
    total = total + valor;
}
escrevaL("O valor Total é de R$", total);
```

Exceções

try, catch e finally

- O termo **try** é utilizado para demarcar um bloco de código que pode gerar algum tipo de exceção
- O termo **catch** oferece um caminho alternativo a ser percorrido no case de ocorrer efetivamente uma exceção
- O termo **finally** delimita um bloco de código que será executado em quaisquer circunstâncias (ocorrendo ou não uma exceção)

Exceções

throw e throws

- A instrução **throw** é utilizada para gerar intencionalmente uma exceção
- O termo **throws** é utilizado para declarar um método que pode gerar uma exceção com a qual não consegue lidar

Exceções

A sintaxe geral de tratamento de exceções

```
try {
    // bloco
} catch(Exception1 var) {
    // bloco
    throw new Exception();
}
// ...
catch (Exception2 var) {
    // bloco
    throw new Exception();
} finally {
    // bloco
}
```

Exceções

ao lançar a **exceção**, é executado o **catch**

```
public static void main(String[] args) {
    int num = 0;
    while (num == 0) {
        try {
            String temp = JOptionPane.showInputDialog("Informe um N°");
            num = Integer.parseInt(temp);
            System.out.println(num);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null,"O n° informado é inválido!");
        }
    }
    System.out.println("Fim");
}
```