Formabase
Aprenda passo a passo

Java
Controlo de Fluxo

Metodologia

- Interprete o documento calmamente e com atenção.
- Acompanhe a execução do exercício no seu computador.
- Não hesite em consultar o formador para o esclarecimento de qualquer questão.
- Não prossiga para o ponto seguinte sem ter compreendido totalmente o ponto anterior.
- Caso seja necessário, execute várias vezes o exercício até ter compreendido totalmente o processo.

Conteúdo programático

1. Operadores e Expressões

Em programação é muito comum falar-se em expressões. As expressões são os componentes das instruções que se destinam a ser avaliados em termos de lógica. O resultado da sua avaliação permite a execução ou não de uma determinada instrução. As expressões podem ser compostas por operadores.

Vamos trabalhar alguns exemplos de operadores de comparação:

› **Crie** uma nova classe com o nome `Comparadores.java`

› **Introduza** o seguinte código na classe:

```
class Comparadores {

   public static void main (String[] args) {
      /*
        *  Exemplo de operadores de comparação
        *  Esta é uma secção de comentários em bloco
        */

      // Operadores de comparação "igual a". Comentário de linha
      System.out.println("Expressão 2 == 1: " + (2 == 1));
      System.out.println("Expressão 2 == 2: " + (2 == 2));

      // Operador de comparação "maior que". Comentário de linha
      System.out.println("Expressão 2 > 1: " + (2 > 1));

      // Operador de comparação "menor que". Comentário de linha
      System.out.println("Expressão 2 < 1: " + (2 < 1));

      // Operadores de comparação "maior ou igual que". Comentário de linha
      System.out.println("Expressão 2 >= 1: " + (2 >= 1));
      System.out.println("Expressão 2 >= 2: " + (2 >= 2));
      System.out.println("Expressão 2 >= 3: " + (2 >= 3));

      // Operadores de comparação "menor ou igual que". Comentário de linha
      System.out.println("Expressão 2 <= 1: " + (2 <= 1));
      System.out.println("Expressão 2 <= 2: " + (2 <= 2));
      System.out.println("Expressão 2 <= 3: " + (2 <= 3));
   }
```

```
}
```

- **Compile** a classe e **execute**

- **Analise** o resultado

> De recordar que os comentários não são avaliados. Servem apenas para esclarecer secções ou instruções de código a quem consultar.

2. Controlo de fluxo

Um dos conceitos mais importantes na programação é o controlo do fluxo de execução de um programa. Este conceito permite que uma condição booleana, cuja avaliação retorne verdadeiro ou falso, condicione a execução de uma ou mais instruções.

Vamos por em prática um exemplo muito simples para analisarmos com atenção:

- **Crie** uma nova classe de nome `Condicao.java` e **introduza** o seguinte código:

```
import java.util.Scanner;

class Condicao {
   public static void main (String[] args) {

       int idade;
```

```
        System.out.println("Introduza a sua idade");
        Scanner teclado = new Scanner(System.in);
        idade = teclado.nextInt();

        if (idade < 19){
            System.out.println("Você é um jovem");
        }
        else {
            System.out.println("Você é um adulto");
        }
    }
}
```

- **Compile** a classe e **execute** com alguns valores

> Certifique-se que compreendeu a ideia do controlo do fluxo do programa.

```
if (condição) {
    bloco de instruções 1
} else {
    bloco de instruções 2
}
```

Se "condição" for verdadeira (isto verifica-se através de um resultado booleano cujo valor pode ser verdadeiro ou falso) é executado o bloco de instruções 1, caso contrário é executado o bloco de instruções 2. O bloco `else` é opcional.

Vamos agora aprofundar um pouco mais a utilização do `if`:

- **Remova** todo o bloco `if` e `else` e **introduza** nesse mesmo local o seguinte código:

```
if (idade < 19){
    System.out.println("Você é um jovem");
}
else if (idade < 65){
    System.out.println("Você é um adulto");
}
else {
    System.out.println("Você é um idoso");
}
```

- **Compile** a classe e **execute**, **testando** com os valores **10**, **40** e **80**

> Repare como foram criadas três possibilidades de resposta consoante a idade que é inserida.

A instrução de `else` que existia anteriormente foi alterada para `else if` para se poder testar mais uma condição. Isto significa que em todas as situações em que se usa `else` pode-se usar `else if` seguido da nova condição que pretendemos testar. O número de `else if`'s que se podem encadear é ilimitado.

- Volte a **trocar** o bloco `if` e respetivos `elses` pelo seguinte código:

```
if (idade < 19){
    System.out.println("Você é um jovem");
}
else if (idade < 65){
    System.out.println("Você é um adulto");
}
else if (idade < 100){
    System.out.println("Você é um idoso");
}
```

```
else {
    System.out.println("Você é um centenário!");
}
```

- **Compile** a classe, **execute** e **experimente** vários valores

  Note que pode ter vários `else ifs` encadeados mas o `else` sem `if` só pode existir no final.

Qualquer bloco de instruções pode conter uma expressão de controlo de fluxo. A este tipo de construção em condições `if`'s chama-se `nested if`'s.

Segue-se a exemplificação deste conceito através de um programa com "if's" dentro de "if's":

- **Crie** a seguinte classe `NestedIfs.java`

- **Adicione** a essa classe o seguinte **código**:

```
import java.util.Scanner;

public class NestedIfs {
   public static void main(String[] args) {

       Scanner teclado = new Scanner(System.in);
       System.out.println("Insira um numero maior que zero");
       int numero = teclado.nextInt();
```

```
        if(numero < 100){
                if (numero > 50){
                        System.out.println("O seu numero esta entre 50 e 100");
                }
                else {
                        System.out.println("O seu numero esta entre 0 e 50");
                }
        }
        else {
                System.out.println("O seu numero e maior que 100");
        }
    }
}
```

- **Compile** e **teste** a classe com vários valores

  Repare como a condição que testa se o número é maior que 50, apenas é feita caso o número seja inferior a 100.

Não existe qualquer tipo de limites para a quantidade de `if`'s que podemos ter dentro de `if`'s.

3. Switch

Existe uma expressão de controlo de fluxo que é alternativa ao if, denominada de switch. Esta, no entanto, serve para quando temos a certeza de quais os valores exatos que a variável toma e não intervalos de valores, assim como fizemos no exemplo anterior.

De seguida, segue-se o exemplo com "if's" encadeados:

› **Crie** uma nova classe de nome `Datas.java` e **introduza** o seguinte código:

```java
import java.util.Scanner;

public class Datas {
    public static void main (String[] args) {

        Scanner teclado = new Scanner(System.in);

        System.out.println("Introduza o dia");
        int dialido = teclado.nextInt();
        System.out.println("Introduza o mes");
        int meslido = teclado.nextInt();
        System.out.println("Introduza o ano");
        int anolido = teclado.nextInt();

        String textoMes = "";

        if (meslido == 1) {
            textoMes = "Janeiro";
        }
        else if (meslido == 2) {
            textoMes = "Fevereiro";
        } else if (meslido == 3) {
            textoMes = "Marco";
        } else if (meslido == 4) {
            textoMes = "Abril";
        } else if (meslido == 5) {
            textoMes = "Maio";
        } else if (meslido == 6) {
            textoMes = "Junho";
        } else if (meslido == 7) {
            textoMes = "Julho";
        } else if (meslido == 8) {
            textoMes = "Agosto";
```

```
        } else if (meslido == 9) {
            textoMes = "Setembro";
        } else if (meslido == 10) {
            textoMes = "Outubro";
        } else if (meslido == 11) {
            textoMes = "Novembro";
        } else if (meslido == 12) {
            textoMes = "Dezembro";
        } else {
            textoMes = “Mês invalido”;
        }

        System.out.println("A sua data de nascimento foi a " +
            dialido + " de " + textoMes + " de " + anolido);

    }
}
```

- **Compile** a classe e **execute** várias vezes passando vários valores

- **Analise** o resultado

Agora passemos ao mesmo exemplo com a instrução switch:

- **Crie** uma nova classe de nome `DatasSwitch.java`

- **Insira** este **código** na nova classe:

```java
import java.util.Scanner;

public class DatasSwitch {
    public static void main (String[] args) {

        Scanner teclado = new Scanner(System.in);

        System.out.println("Introduza o dia");
        int dialido = teclado.nextInt();
        System.out.println("Introduza o mes");
        int meslido = teclado.nextInt();
        System.out.println("Introduza o ano");
        int anolido = teclado.nextInt();


        String textoMes = "";

        switch (meslido){
            case 1: textoMes = "Janeiro"; break;
            case 2: textoMes = "Fevereiro"; break;
            case 3: textoMes = "Março"; break;
            case 4: textoMes = "Abril"; break;
            case 5: textoMes = "Maio"; break;
            case 6: textoMes = "Junho"; break;
            case 7: textoMes = "Julho"; break;
            case 8: textoMes = "Agosto"; break;
            case 9: textoMes = "Setembro"; break;
            case 10: textoMes = "Outubro"; break;
            case 11: textoMes = "Novembro"; break;
            case 12: textoMes = "Dezembro"; break;
            default : textoMes = "Mês invalido";
        }

        System.out.println("A sua data de nascimento foi a " +
            dialido + " de " + textoMes + " de " + anolido);
```

```
    }
}
```

- **Compile** e **execute** a classe

- **Faça** vários **testes**

A sintaxe do switch é a seguinte:

```
switch (expressão) {
    case valor1:
        bloco de instruções
        break;
    case valor2:
        bloco de instruções
        break;
    default:
        bloco de instruções
}
```

O `default` serve para todos os valores da expressão que não foram definidos anteriormente. A palavra reservada `break` existe para dar indicação ao programa de que após executar o código referente ao caso em questão, deve sair do bloco **switch** e não executar os restantes casos.

A partir da versão 1.7 do java já é possível utilizar uma **String** na expressão do **switch**.