Instruções de controlo de fluxo

UA.DETI.POO

Controlo de fluxo num programa

- A ordem de execução das instruções de um programa é normalmente linear
  - uma declaração após a outra, em sequência
- Algumas instruções permitem alterar esta ordem, decidindo:
  - se deve ou não executar uma declaração particular
  - executar uma declaração repetidamente, repetidamente
- Essas decisões são baseadas em expressões booleanas (ou condições)
  - que são avaliadas como verdadeiras ou falsas

Expressões booleanas

- Expressões booleanas retornam `true` ou `false`.
- As expressões booleanas usam operadores relacionais, de igualdade, e lógicos (AND, OR, NOT)

```
==      equal to        // Atenção!! x == y é diferente de x = y
!=      not equal to
<       less than
>       greater than
<=      less than or equal to
>=      greater than or equal to
!       NOT
&&      AND
||      OR
```

- Exemplos

```
x >= 10
(y < z) && (z > t)
```

Tabelas de verdade

- A álgebra booleana é baseada em tabelas de verdade.
- Considerando A e B, por ex: `((y < z) && (z > t))`
  - Ambos têm que ser verdadeiros para a expressão **A && B** ser verdadeira.
  - Basta um ser verdadeiro para a expressão **A || B** ser verdadeira.

a	*!a*
true	false
false	true

a	*b*	*a && b*	*a || b*
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Operador ternário

- O operador ternário (**?:**) é também conhecido como operador condicional.

```
result = testCondition ? valueIfTrue : valueIfFalse
```

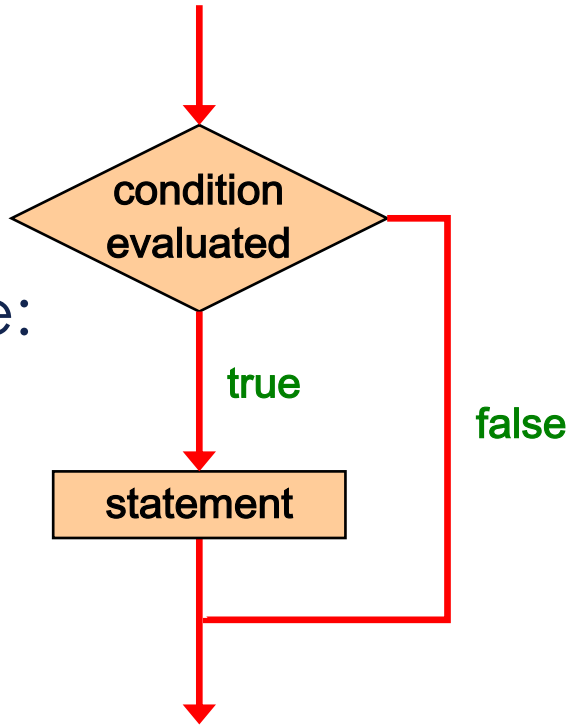
  - Avalia uma expressão (1º operando) e, caso seja true, o resultado é igual ao 2º operando, caso contrário o resultado é igual ao 3º operando.

```
char code = 'F';
boolean capitalLetter = (code >= 'A') && (code <= 'Z');
System.out.println(capitalLetter ? "sim" : "não");

minVal = (a < b) ? a : b;
```

Instruções condicionais

- Em Java existem dois tipos de instruções de decisão/seleção:
  - **if**
  - **switch**



- A instrução if tem o formato seguinte:

```
if (expressãoBooleana)
  // fazer_isto;
else  // opcional
  // fazer_aquilo;
```

Exemplo

```
Scanner sc = new Scanner(System.in);
int number = sc.nextInt();

if (number % 2 == 0)
    System.out.println("O número é par");
else
    System.out.println("O número é ímpar");

sc.close();
```

Instrução de decisão if

- Podemos encadear várias instruções if:

```
if (condição1)
    bloco1;
else if (condição2)
    bloco2;
else
    bloco3;
```

Se um bloco incluir mais que uma instrução, o bloco deve ser delimitado por { .. }.

- Exemplo

```
if (faltas <= 3)
    System.out.println("Pode ir ao exame teórico.");
else if (!regime.equals("T"))
    System.out.println("Reprovado por faltas.");
else {
    System.out.println("Aluno trabalhador sem a/c.");
    System.out.println("Deve fazer exame prático.");
}
```

Instrução de seleção switch

- A instrução switch executa um de entre vários caminhos (case), consoante o resultado de uma expressão

```
switch (expressão) {
    case valor1:
        bloco1;
        break;
    case valor2:
        bloco2;
        break;
    //...
    default:
        blocoFinal;
}
```

O resultado da expressão é pesquisado na lista de alternativas existentes em cada case, pela ordem com que são especificados.

Se a pesquisa for bem sucedida, o bloco de código correspondente é executado. Se houver a instrução break, a execução do switch termina. Caso contrário serão executadas todas as opções seguintes até que apareça break ou seja atingido fim do switch.

Se a pesquisa não for bem sucedida e se o default existir, o bloco de código correspondente (blocoFinal) é executado.

Exemplo

```
switch (category) {
      case 10:
         System.out.println ("a perfect score. Well done.");
         break;
      case 9:
         System.out.println ("well above average. Great.");
         break;
      case 8:
         System.out.println ("above average. Nice job.");
         break;
      case 7:
         System.out.println ("average.");
         break;
      case 6:
         System.out.println ("below average.");
         System.out.println ("See the instructor.");
         break;
      default:
         System.out.println ("not passing.");
   }
```

Exemplo

```
Scanner sc = new Scanner(System.in);
int mes = sc.nextInt();
int dias;
switch (mes)
{  case 4:
   case 6:
   case 9:
   case 11: dias = 30; break;
   case 2:  dias = 28; break;
   default: dias = 31;
}
System.out.println("Mês tem " + dias + " dias");
sc.close();
```

Ciclos

- Por vezes existe a necessidade de executar instruções repetidamente.
  - A um conjunto de instruções que são executadas repetidamente designamos por ciclo.

- Um ciclo pode ser do tipo condicional (**while** e **do…while**) ou do tipo contador (**for**).
  - Normalmente utilizamos ciclos condicionais quando o número de iterações é desconhecido e ciclos do tipo contador quando sabemos à partida o número de iterações.

Ciclo while

- O ciclo **while** executa enquanto a condição do ciclo esteja verdadeira.
  - A condição é avaliada antes de cada iteração do ciclo.

```
while (condição)
    bloco_a_executar;
```

  - Exemplo:

```
Scanner sc = new Scanner(System.in);
int nota = -1;
while ( (nota > 20) || (nota < 0) ) {
    System.out.println("Insira a nota do aluno.");
    nota = sc.nextInt();
}
sc.close();
```

Ciclo do while

- O ciclo **do...while** executa uma primeira vez e só depois verifica se é necessário repetir.
  - A condição é avaliada no fim de cada iteração do ciclo.

```
do
    bloco_a_executar;
while (condição);
```

  - Exemplo:

```
Scanner sc = new Scanner(System.in);
int nota;
do {
    System.out.println("Insira a nota do aluno.");
    nota = sc.nextInt();
} while ( (nota > 20) || (nota < 0) );
sc.close();
```

Ciclo for

- O ciclo **for** é mais geral pois suporta todas as situações de execução repetida.

```
for (inicialização; condição; atualização)
    bloco_a_executar;
```

1. Antes da 1ª iteração, faz a **inicialização** (só uma vez)
2. Depois realiza o teste da **condição**.
   Se for *true* executa o bloco, se for *false* termina
3. No fim de cada iteração, executa a parte de **atualização** e retoma no ponto 2 anterior.

Exemplos

Exemplo 1

```
for (int i = 1 ; i <= 10 ; i++)
    System.out.println(i+" * "+ i +" = " + i*i);
```

Exemplo 2

```
int[] tb = new int[10];
for (int i = 0 ; i < tb.length ; i++)
    tb[i] = i * 2 ;
for (int i = 0 ; i < tb.length ; i++)
    System.out.print(tb[i] + ", ");
```

```
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
10 * 10 = 100
```

```
0, 2, 4, 6, 8, 10, 12, 14, 16, 18,
```

Ciclo for (sintaxe foreach)

- O ciclo for, quando usado com vetores, pode ter uma forma mais sucinta (foreach).

```
Scanner sc = new Scanner(System.in);
double[] a = new double[5];
for (int i = 0; i < a.length; i++)
    a[i] = sc.nextDouble();

for (double el : a)
    System.out.println(el);

sc.close();
```

Instruções break e continue

- Podemos terminar a execução dum bloco de instruções com duas instruções especiais:
  - **break** e **continue**.

- A instrução **break** permite a saída imediata do bloco de código que está a ser executado.
  - É usada normalmente em switch mas também pode ser usada em ciclos.
- A instrução **continue** permite terminar a execução da iteração corrente, forçando a passagem para a iteração seguinte (i.e. não termina o ciclo).

Instruções break e continue

- Exemplo:

```
public class Testes {
    public static void main(String[] args) {
        int[] numbers = { 10, 20, 30, 40, 50 };
        for (int x : numbers) {
            if (x == 30) {
                break;
            }
            System.out.println(x);
        }
    }
}
```

```
10
20
```

Sumário

- Instruções condicionais
  - if
  - if .. else
  - switch
- Instruções de ciclos
  - while
  - do … while
  - for