Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenahria Informática
Introdução à Programação

Guia Prático 2
Seleção, repetição e aleatoriedade

João Paulo Barros

Beja, 13 de outubro de 2016, 16 de outubro de 2016, 24 de outubro de 2017

Objectivos: Escrever programas que utilizem seleção e repetição. Conhecer e aplicar números pseudoaleatórios.

Conteúdo

1 Seleção (`if`)

A principal forma de seleção na linguagem Java™ (e muitas outras) é a instrução `if`. Esta permite a execução condicionada de um bloco de código associado.

1.1 Exemplos

Seguem-se alguns exemplos e respetivo *output*:

1. ```
   int value = 10;
   int limit = 15;
   if (value < limit)
   {
       System.out.println("low value");
   }
   System.out.println("end");
   ```

   irá escrever, no terminal, o seguinte texto

   ```
   low value
   end
   ```

2. ```
   int value = 100;
   int limit = 15;
   if (value < limit)
   {
       System.out.println("low value");
   }
   System.out.println("end");
   ```

   irá escrever, no terminal, o seguinte texto

   ```
   end
   ```

3. A instrução `if` pode, opcionalmente, ser complementada com uma palavra reservada **`else`**.

   ```
   int value = 50;
   int limit = 15;
   if (value < limit)
   {
       System.out.println("low value");
   }
   else {
       System.out.println("high value");
   }
   System.out.println("end");
   ```

   irá escrever, no terminal, o seguinte texto

   ```
   high value
   end
   ```

4. Finalmente, podemos colocar, entre o `if` e o `else`, a combinação `else if`:

```
int value = 25;
if (value < 10)
{
    System.out.println("less than 10");
}
else if (value < 20)
{
    System.out.println("less than 20");
}
else if (value < 30)
{
    System.out.println("less than 30");
}
else {
    System.out.println("high value");
}
System.out.println("end");
```

irá escrever, no terminal, o seguinte texto

```
less than 30
end
```

1.2 Exercícios

1. Qual o output produzido por cada um dos seguintes excertos de código:

   (a)

```
int a = 3;
int b = 4;
int c = 5;
if (a < 3)
{
    System.out.println(a);
}
else if (a > 3)
{
    System.out.println(b);
}
else
{
    System.out.println(c);
}
```

(b)

```
int a = 3;
int b = 4;
int c = 5;
if (a == 3)
{
    System.out.println(a);
}
else if (b == 4 )
{
    System.out.println(b);
}
else if (c == 5)
{
    System.out.println(c);
}
```

(c)

```
int a = 3;
int b = 4;
int c = 5;
if (a == 3)
{
    System.out.println(a);
}
if (b == 4 )
{
    System.out.println(b);
}
if (c == 5)
{
    System.out.println(c);
}
```

(d)

```
int a = 3;
int b = 4;
int c = 5;
if (a <= 3)
{
    System.out.println(a);
    if (a == 3)
    {
        System.out.println(a);
    }
    else
    {
        System.out.println(c);
    }
}
```

(e)

```
int a = 3;
int b = 4;
int c = 5;
if (a == 3)
{
    System.out.println(a);
}
if (3 < b)
{
    System.out.println(b);
}
else
{
    if (c == 5)
    {
        System.out.println(c);
    }
}
```

(f)

```
int a = 3;
int b = 4;
int c = 5;
if (a + b < c + 2)
{
    System.out.println(a);
}
if (a == 3)
{
    if (b == 4)
    {
        System.out.println(b);
        if (c == 5)
        {
            System.out.println(5);
        }
    }
    else
    {
        System.out.println("else");
    }
    if (a < b)
    {
        System.out.println(c);
    }
}
```

2. Altere o programa que calcula as soluções da equação de segundo grau de forma a que o mesmo escreva uma mensagem de texto diferente conforme os seguintes três casos: não há raízes; há uma raiz dupla (x1 é igual a x2); há duas raízes. Sugere-se que siga os seguintes passos: o programa pede os valores dos coeficientes (a, b e c) e calcula o $delta$ (b^2-4ac). Se o $delta$ for negativo, então termina com uma mensagem informando que não há raízes; se o delta for igual a zero é porque tem uma raiz dupla e escreve uma mensagem com essa informação; se o delta for maior, escreve uma mensagem informando que há duas raízes.

3. Altere o programa anterior de forma a também mostrar os valores das raízes quando existentes. Procure que o programa não efectue cálculos desnecessários. Por exemplo, se o delta é zero não precisa de calcular duas raízes pois serão iguais.

2 Repetição (ciclos)

A repetição de instruções é extremamente comum. Especialmente a repetição de instruções semelhantes, mas não iguais, está na base de muitíssimos algoritmos.

É usual considerar que as repetições de instruções são de três tipos:

0 ou mais vezes As instruções podem não ser executadas ou ser executadas uma quantidade de vezes que não podemos prever quando escrevemos o código. Em Java™, tal é especificado com um ciclo **`while`** o qual tem a seguinte sintaxe:

```
while (condição)
{
        // código a repetir
}
```

É como uma instrução de selecção (`if`) que se repete enquanto for verdade.

1 ou mais vezes As instruções são sempre executadas uma vez. Depois são executadas uma quantidade de vezes adicional que não podemos prever quando escrevemos o código. Em Java™, tal é especificado com um ciclo **`do while`** que tem a seguinte sintaxe:

```
do
{
        // código a repetir
} while (condição);
```

É como uma instrução de selecção (`if`) no final que, enquanto for verdade, faz saltar para o início.

***n* vezes** As instruções são sempre executadas *n vezes*. Deve escolher este tipo de ciclo quando pretende que as instruções sejam executadas uma quantidade de vezes predeterminada. Em Java™, tal é especificado utilizando o ciclo **for**. Este é uma abreviatura do ciclo **`while`**. Tem a seguinte sintaxe:

```
for(inicialização; condição; incremento)
{
      // código a repetir
}
```

É uma instrução para repetir algo (que está num bloco de código) uma dada quantidade de vezes.

O diagrama na Fig. 1 exemplifica o funcionamento do ciclo **`while`** e **`for`**. Ambos os ciclos escrevem o dobro de cada um dos números entre `0` e `n - 1`. Note que são apenas sintaxes diferentes para uma mesma sequência de instruções.

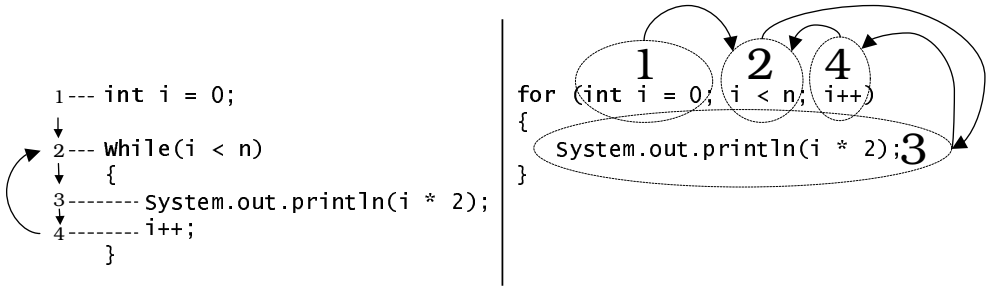

Figura 1: Funcionamento dos ciclos **while** e **for**.

saída a meio As "receitas"anteriores são as preferíveis e na maior parte dos casos devemos preferir uma delas para programa ciclos. Mas, por vezes, o código fica mais legível e/ou o algoritmo pretendido é mais fácil de codificar utilizando uma saída a meio do ciclo. Para tal utiliza-se a instrução **break**. Frequentemente a mesma é utilizada dentro de um **if**. Por exemplo:

```
do
{
    // ...
    if (condição de saída)
    {
        break;
    }
    // ...
} while (condição);
```

Esta saída no meio é muitas vezes utilizada com um ciclo que seria infinito se não se utilizasse a instrução **break**. Por exemplo:

```
while(true)
{
     // ...
     if (condição de saída)
     {
        break;
     }
     // ...
}
```

ou

```
for(;;)
{
     // ...
     if (condição de saída)
     {
        break;
     }
     // ...
}
```

Finalmente importa notar que a instrução **break**pode ser utilizada mais do que uma vez e em qualquer ciclo. Mas conforme já referido, a sua utilização deve ser vista como um último recurso, uma excepção à regra de utilizar um dos três tipos de ciclo: **0 ou mais vezes**; **1 ou mais vezes**, **n vezes**;

2.1 Exercícios

1. Qual o output produzido por cada um dos seguintes excertos de código:

   (a)

```
int a = 3;
int b = 6;
while (a < b)
{
    System.out.println(a);
    a++;
    System.out.println(a);
}
```

   (b)

```
int a = 3;
int b = 6;
do
{
    System.out.println(a);
    a++;
    System.out.println(a);
} while (a < b);
```

   (c)

```
int a = 3;
int b = 6;
do
{
    System.out.println(a * 3);
    a++;
    System.out.println(a);
} while (a < b);
```

   (d)

```
int a = 3;
int b = 6;
do
{
    System.out.println(a + b);
    a = a + b;
    System.out.println(a);
} while (a < b * b);
```

   (e)

```
int a = 1;
for(int i = 0; i < 5; i++)
{
    for(int j = 0; j < i; j++)
    {
        System.out.println(i + ", " + j);
    }
}
```

3 Pergunta ao utilizador – leitura de texto e repetição

Para perguntar ao utilizador se pretende que um código seja executado novamente podemos utilizar o seguinte idioma:

```
String ans = "";
do
{

   // here add code too eventually repeat

   System.out.println("Repeat? (y/n)");
   ans = scanner.next();
} while (ans.equals("y"));
```

3.1 Exercícios

1. Altere o programa do exercício 3 da Secção 1.2 de forma a que o mesmo pergunte ao utilizador se pretende indicar novo polinómio. Se a resposta for "s" então o programa deve voltar para o início. Considere o código da Secção 3.
2. Adicione a mesma possibilidade de repetição ao programa para cálculo do índice de massa corporal já relatado na aulas.

4 Números aleatórios

Os números verdadeiramente aleatórios têm de resultar de um processo físico, por exemplo atirar uma moeda ao ar. Num computador é mais fácil gerar números pseudo-aleatórios. Estes parecem aleatórios e até passam em testes estatísticos mas são calculados por fórmulas que tipicamente têm como base o último número gerado. Para tal, essas fórmulas partem de um valor inicial a que se chama "semente" (*seed*).

4.1 Gerar um número pseudoaleatório utilizando Java™

Em Java™, a função `Math.random()` devolve um número pseudo-aleatório entre 0.0 e 1.0, excluindo o 1.0, ou seja, no intervalo $[0,1[$. Se pretendermos outro intervalo podemos fazer uma conta. Por exemplo, para o intervalo $[3,15]$ podemos utiliza o seguinte código:

```
double rand = Math.random();
int between3And15 = (int)(3 + 13 * rand);
```

É quase sempre boa ideia utilizar uma fórmula geral:

```
int min = 3;
int max = 15;
double rand = Math.random();
int between3And15 = (int)(min + (max - min  + 1) * rand);
```

4.2 Exercícios

1. Escreva um programa que pede um valor mínimo e um valor máximo ao utilizador. Depois, o programa entra num ciclo que repete 10 vezes. Em cada iteração desse ciclo, o programa gera um número aleatório e dá 5 tentativas para o jogador adivinhar o número. De cada vez o programa indica se o utilizador acertou ou não. No final, mostra quantas vezes o utilizador acertou e quantas errou. Considere as seguintes etapas para construir o seu programa:

   (a) O programa pede ao utilizando um número mínimo e um número máximo. Depois gera um número aleatório (*vide* Secção 4) nesse intervalo. Seguidamente, pede ao utilizador para indicar um número nesse intervalo e informa o utilizador se acertou ou não no número gerado aleatoriamente.

   (b) Agora o programa pergunta cinco vezes ao utilizador. Se o utilizador acertar no número ou se esgotar as cinco tentativas, o programa termina.

   (c) Finalmente, o programa da alínea anterior é agora executado dez vezes. Para tal utilize um ciclo `for`, tal como exemplificado na Secção 2. O programa mostra quantas vezes o utilizador acertou e quantas errou, quer em número absoluto em quer em percentagens. Por exemplo:

   ```
   Acertou  2 vezes (20%)
   Errou    8 vezes (80%)
   ```

   Note que o output deve respeitar o formato indicado. Para tal utilize o printf com os formatos adequado. Veja, por exemplo, **esta página**, em partícular a secção "Controlling integer width with printf".

2. Escreva um programa que dado um numero inteiro positivo, escreve todo os números pares inteiros positivos menores do que esse número. Resolva de três formas, utilizando um ciclo **`for`**, um ciclo **`while`** e um ciclo **`do while`**. Qual lhe parece a melhor solução? E porquê?
   Exemplo: dado o número 15, o programa deve escrever o seguinte:
   ```
   0 2 4 6 8 10 12 14
   ```

3. Escreva um programa semelhante ao anterior mas que escreve duas listas de números separados por vírgulas. os números pares e os números ímpares.
   Exemplo: dado o número 15, o programa deve escrever o seguinte:
   ```
   0, 2, 4, 6, 8, 10, 12, 14
   1, 3, 5, 7, 9, 11, 13
   ```

4. Escreva um programa que dadas duas temperaturas em graus Fahrenheit (valores inteiros) produz uma tabela com todos os valores nesse intervalo intervalados de meio grau Fahrenheit. Por exemplo, a execução do programa para os valores 30 e 35 deve produzir o seguinte *output*:

   ```
   Indique valores do intervalo separados por espaço (min max):
   30
   35
   Fahrenheit Celsius
   30.00  -1.11
   30.50  -0.83
   31.00  -0.56
   31.50  -0.28
   32.00   0.00
   32.50   0.28
   33.00   0.56
   33.50   0.83
   34.00   1.11
   34.50   1.39
   35.00   1.67
   ```

Uma forma de obter a formação para cada linha de números será com a seguinte string num printf: `"%6.2f\t\t%6.2f\n"`. O `\t` significa tabulação; o `6` significa seis espaços no total para cada número. O `2` significa dois espaços para a parte decimal de cada número.

5. Escreva um programa que dado um número inteiro maior do que zero, escreve todos os números entre zero e esse número. Utilize um ciclo **`while`**.

6. Escreva um programa que dado um número inteiro maior do que zero, escreve todos os números entre zero e esse número. Utilize um ciclo **`do while`**.

7. Escreva um programa que dado um número inteiro maior do que zero, escreve a soma de todos os números entre zero e esse número. Utilize um ciclo **`while`**. Procure evitar repetição de código.

8. Escreva um programa que dado um número inteiro maior do que zero, escreve a soma de todos os números entre zero e esse número. Utilize um ciclo **`do while`**. Procure evitar repetição de código.

9. Nos quatro programas anteriores (5 a 8) acrescente um ciclo **`do while`** para garantir que o valor indicado pelo utilizador está entre zero e 20.

10. Escreva um programa que recebe uma quantidade de horas minutos e segundos e devolve o mesmo tempo em segundos.

11. Escreva um programa que recebe uma quantidade de segundos e devolve o mesmo tempo em horas minutos e segundos. Note que deve utilizar o operador % para obter o resto da divisão inteira.

12. Escreva um programa que recebe as notas de n testes e calcula a nota final como sendo a média aritmética das notas nos n testes. O programa começa por perguntar quantos testes são. Seguidamente pergunta a nota em cada um dos testes. Finalmente escreve a nota final. Por exemplo, para 3 testes o output do output no terminal será o seguinte:

```
Indique a quantidade de testes:
3
Indique a nota no teste 1:
10
Indique a nota no teste 2:
15
Indique a nota no teste 3:
20
A média dos testes é de 15.00.
```

13. Escreva um programa que recebe duas pontuações (números inteiros); ordena essas pontuações utilizando instruções **`if`** e fazendo trocas entre elas e escreve ambas por ordem crescente no ecrã. Note que pode trocar o valor de duas variáveis utilizando uma terceira variável:

```
int a = 2;
int b = 3;
a = b;
b = a; // não funciona
```

```
int a = 2;
int b = 3;
int tmp = a;
a = b;
b = tmp; // funciona!
```

14. Escreva um programa que recebe três pontuações (números inteiros); ordena essas pontuações utilizando instruções **`if`** e fazendo trocas entre elas e escreve as três por ordem crescente no ecrã.

15. Escreva um programa que recebe a quantidade de produtos iguais comprados e o preço unitário. Se o preço total (de todos os produtos somados) for inferior ou igual a 100 não é aplicada taxa; se o preço for maior do que 100 e menor ou igual a 500 é aplicada uma taxa de 20%; se o preço for superior a 500 é aplicada uma taxa de 30%. O programa deve mostrar dois preços totais a pagar: antes e depois de aplicada a taxa.

16. Escreva um programa que recebe um valor *n* e produz um "triângulo" de asteriscos. Por exemplo para *n* igual a 1, 2, 3, 4 e 20 o *output* deve ser o seguinte:

```
n:
1
*
n:
2
*
**
n:
3
*
**
***
n:
4
*
**
***
****
n:
20
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
*****************
******************
*******************
********************
n:
```

Note que deve utilizar dois ciclos, de preferência **`for`**: um para linha e outro, dentro desse, para os asteriscos nessa linha. Após cada "triângulo", o programa pede sempre um novo valor para *n*.

Deve concluir a resolução deste guia depois da aula. Traga as dúvidas para a próxima aula ou coloque-as no fórum de dúvidas da disciplina. As sugestões para melhorar este texto também são bem-vindas.