Estruturas de controlo

- Estruturas de controlo em Java
  - Sequência
    - Implementada pela ordem de colocação das instruções
  - Seleção
    - Instrução if
    - Instrução if - else
    - Instrução switch
  - Repetição
    - Instrução while
    - Instrução do
    - Instrução for

Estruturas de controlo

- Instrução composta
  - Uma instrução composta é definida como sendo um grupo de uma ou mais instruções limitadas por chavetas:

```
{
instrução 1;
...
instrução n;
}
```

  - Pode ser colocada em qualquer ponto de um programa onde a sintaxe do Java permita a colocação de uma instrução simples

Estruturas de controlo

- Instrução composta
  - Se a instrução composta tiver apenas uma instrução, as chavetas podem ser omitidas
  - Daqui para a frente, quando se falar de instrução, estamos a falar de instrução composta

Estruturas de controlo

- Seleção simples
  - Implementada pela instrução if
  - Sintaxe da instrução:
    - *if (condição)*

      instrução;
  - A condição é uma expressão lógica
  - Caso o resultado da condição seja true a instrução é executada, se for false a instrução não é executada

Estruturas de controlo

- Seleção em alternativa
  - Implementada pela instrução if - else
  - Sintaxe da instrução:
    - *if (condição)*

      instrução 1;

      else instrução 2;
  - A condição é uma expressão lógica
  - Se o resultado da condição for true a instrução 1 é executada, se for false é executada a instrução 2

Estruturas de controlo

- Selecionar entre várias alternativas
  - A execução de uma estrutura encadeada de instruções if permite escolher uma de várias alternativas, em função do resultado das diversas condições utilizadas.
  - As condições são calculadas por ordem até que uma delas dê true, sendo então executadas as instruções respetivas. Depois passa a ser executada a instrução seguinte a esta estrutura.
  - Caso nenhuma expressão dê true são executadas as instruções relativas ao último else (se existir).

Estruturas de controlo

- Selecionar entre várias alternativas
  - Ex: Selecionar uma de três alternativas em função do valor de uma variável. Se nenhuma se verificar deve fazer a instrução 4.

```
if (num >= 1 && num <= 10)
    instrução 1;
else if (num > 10 && num <= 20)
        instrução 2;
    else if (num > 20 && num <= 30)
            instrução 3;
        else instrução 4;
```

Estruturas de controlo

- Seleção - Exemplo 1
  - Determinar se um número é par ou impar

```
public class ParOuImpar {
   public static void main (String[] args) {
   int num;
   Scanner sc = new Scanner(System.in);
   System.out.print (“Qual o número ? “);
   num = sc.nextInt ();
   if (num % 2 == 0)
            System.out.println (“Número é par”);
   else System.out.println (“Número é impar”);
   }
}
```

Estruturas de controlo

- Seleção - Exemplo 2
  - Dada a idade de uma pessoa definir o tipo de bilhete

```
public class Bilhete {
   public static void main(String[] args) {
   int idade;
   Scanner sc = new Scanner(System.in);
   System.out.print (“Idade: “);
   idade = sc.nextInt();
   if (idade <= 6) {
          System.out.println (”Isento de pagamento");
   } else if (idade <= 12) {
                 System.out.println (”Bilhete de criança ");
          } else if (idade <= 65) {
                        System.out.println (”Bilhete normal");
                 } else System.out.println (”Bilhete de terceira
   idade");
   }
}
```

Estruturas de controlo

- Seleção múltipla
  - A instrução switch permite escolher uma de várias alternativas, em função do valor de uma expressão. Esta instrução tem a seguinte sintaxe:

```
switch (expressão) {
 case valor1 : instrução 1; break;
 case valor2 : instrução 2; break;
 ...
 case valorn : instrução n; break;
 default : outra instrução; break;
}
```

Estruturas de controlo

- Seleção múltipla
  - A expressão utilizada numa instrução switch terá que ter um resultado inteiro ou caracter
  - A instrução break é colocada no fim das instruções relativas a cada case. Tem como função provocar o fim do switch
  - A opção default (pode não existir) é executada se o valor da expressão não for igual a nenhum dos valores utilizados no switch (valor1, … valorn)

Estruturas de controlo

- Seleção múltipla - Exemplo
  - Dados dois operandos (x e y) e um operador (op) determinar o resultado (x op y)

```
switch (op) {
   case '+' : resul = x + y; break;
   case '-' : resul = x - y; break;
   case '*' : resul = x * y; break;
   case '/' : if (y != 0)
                  resul = x / y;
              else {
                  System.out.println ("Atenção! Divisão por zero");
                  resul = 0.0;}
              break;
}
```

Estruturas de controlo

- Repetição - Enquanto Faz
  - Implementada pela instrução while
  - Sintaxe da instrução:

    while (condição)

    instrução;
  - A condição é uma expressão lógica
  - A instrução começa por calcular o valor da condição. Se der true é executado a instrução (composta) após o que a condição é novamente avaliada. Esta repetição será mantida enquanto o valor da condição se mantiver true. Quando tal não acontecer a instrução termina.

Estruturas de controlo

- Repetição - Enquanto Faz
  - A condição é avaliada antes da execução da instrução a repetir, pelo que, se a expressão for falsa à partida, a instrução nunca chega a ser executada
  - Há que ter em conta a possibilidade de criação de ciclos infinitos (situação esta que deve ser evitada)
    - Deve haver o cuidado de incluir instruções que, em alguma situação, alterem o valor da condição de controle do ciclo, de modo a que este termine.
  - A inicialização da(s) variável(is) que controla(m) o ciclo e a própria condição de controle devem ser vistas com cuidado, de modo a que o ciclo repita o número de vezes desejado

Estruturas de controlo

- Repetição - Enquanto Faz - Exemplo
  - Escrever no ecrã todos os inteiros entre 1 e 5:

```
public class WhileDemo {
    public static void main(String[] args) {
     int i = 1;

     while (i <= 5) {
         System.out.println  (i);
         i = i + 1;
        }
     }
 }
```

Estruturas de controlo

- Repetição - Faz Enquanto
  - Implementada pela instrução do
  - Sintaxe da instrução:

```
do
   instrução;
while (condição)
```

  - A condição é uma expressão lógica
  - Começa por executar a instrução, após o que calcula o valor da condição. Se der true a instrução é executada novamente após o que a condição é de novo avaliada.
  - Esta repetição será mantida até que o valor da condição passe a false. Quando tal acontecer a instrução termina.

Estruturas de controlo

- Repetição - Enquanto Faz
  - A condição é avaliada depois da primeira execução da instrução a repetir, pelo que, mesmo que a expressão seja falsa à partida, a instrução será sempre executada uma vez (pelo menos)
  - Devem ser observados os mesmos cuidados referidos relativamente à instrução while (ciclos infinitos e número de repetições)

Estruturas de controlo

- Repetição - Faz Enquanto - Exemplo
  - Escrever no écrã todos os inteiros entre 1 e 5:

```
public class WhileDemo {
    public static void main(String[] args) {
    int i = 1;

    do {
        System.out.println  (i);
        i = i + 1;
     } while (i <= 5);
   }
}
```

Estruturas de controlo

- Repetição - Faz
  - Implementada pela instrução for
  - Sintaxe da instrução:

    for (início; teste; ação)
    instrução
  - Começa por executar a expressão início (só é executada uma vez)
  - Depois verifica o valor da condição teste. Se der false a repetição termina. Se der true a instrução é executada seguindo-se-lhe a expressão ação. Após isto o teste é de novo feito e, caso dê true, a instrução e a ação são de novo executadas. A repetição termina quando o teste der false

Estruturas de controlo

- Repetição - Faz
  - A condição é avaliada antes da primeira execução da instrução a repetir, pelo que a instrução poderá nunca ser executada
  - Devem ser observados os mesmos cuidados referidos relativamente à instrução while (ciclos infinitos e número de repetições)

Estruturas de controlo

- Repetição - Faz - Exemplo
  - Escrever no écrã todos os inteiros entre 1 e 5:

```
public class ForDemo {
    public static void main(String[] args) {
     int i;

     for (i=1; i <=5; i++)
        System.out.println (i);
    }
}
```

Estruturas de controlo

- Operadores de incrementação / decrementação
  - Estes operadores permitem incrementar ou decrementar o valor de uma variável. O operador ++ adiciona 1 ao seu operando, enquanto que o -- subtrai-lhe 1.
  - Exemplo:

    exp++ ou ++exp <==> exp = exp +1;
  - Existem duas formas destes operadores, antes ou depois dos operandos. Esta distinção só é relevante quando fazem parte de uma expressão maior.

Estruturas de controlo

- Operadores de incrementação / decrementação
  - Por exemplo, supondo que a variável n tem o valor 7, a expressão `x = ++n;` incrementa o valor de n (passa a 8) e atribui-o a x (por isso x passa a ter o valor 8).
  - Se utilizarmos `x = n++;` primeiro é feita a atribuição de n a x (fica com o valor 7) e só depois é que é feita a incrementação de n;
  - O operador de decrementação -- tem um funcionamento análogo ao ++

Estruturas de controlo

- Instruções de repetição - Exemplo
  - Calcular a soma dos dígitos de um número

```
public class SomaDigitos {
    public static void main(String[] args) {
     long num;
     int soma = 0;
     System.out.println (“Escreva o número: “);
     Scanner sc = new Scanner(System.in);
     num = sc.nextLong();
     for(;num > 0; num = num / 10) {
         soma = soma + (num % 10);
     }
     System.out.println (“Soma dos dígitos = ” + soma);
  }
}
```

Estruturas de controlo

- Instruções de repetição - Exemplo
- Programa para escrever uma pirâmide de números. O número de linhas é dado pelo utilizador:

```
public class Piramide {
    public static void main(String[] args) {
     int n, i, j, k;
     System.out.print (“N= “);
     Scanner sc = new Scanner(System.in);
     n = sc.nextInt();
     for (j = 1; j <= n; j++) {
         for (k = 1; k <= n-j; k++) System.out.print (“ “);
         for (i = 1; i <= j; i++) System.out.print (i);
         for (i = j-1; i > 0; i--) System.out.print (i);
         System.out.println ();
   }
  }
```