Estácio

Programação I

PROF. LUCAS CAMPOS DE M. NUNES

http://lattes.cnpq.br/2803226406709573

Brasília, agosto de 2017

Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

As convenções do Java

Java é “Case Sensitive”. As convenções utilizadas:

- Nome de variáveis e métodos começam com letras minúsculas
- Nome de classes iniciam com letras maiúsculas;
- Nome composto: utilizar letras maiúsculas para as iniciais das palavras;
- Letras maiúsculas para as constantes;

Case Sensitive – Sensível ao tamanho: em computação significa que um programa ou um compilador faz a diferenciação entre letras maiúsculas e minúsculas, ou seja, Maiúscula é diferente de maiúscula. O sistema operacional Linux é case sensitive bem como as linguagens C, Java, C Sharp entre outras.

Estruturas básicas de programação

- Blocos
- Escopo das variáveis
- Comando Condicional
- Desvios de Fluxo
- Estruturas de repetição ou laço

Blocos:

- Conjunto de linhas de códigos situadas entre um abre e um fecha chaves({}). É permitido criar blocos dentro de blocos.

```
{ //início de bloco
    ...
      /*bloco de comandos*/
    ...
} //fim de bloco
```

Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

Tipos de Dados

- Quando vamos escrever um programa de computador, em qualquer linguagem, vamos ter de usar variáveis;
- As variáveis servem para guardar valores dos mais variados tipos: podem guardar números inteiros ou decimais, valores alfanuméricos, somente alfabéticos, valores lógicos (como verdadeiro ou falso) e muitos outros;
- Existem linguagens que permitem que o programador crie seus próprios tipos;

Tipos de Dados

- Vamos usar um algoritmo bem simples para ilustrar:

```
algoritmo Soma
numero1, numero2, soma: inteiro
inicio
      soma = 0
      escreval("Digite o primeiro número inteiro:")
      leia(numero1)
      escreval("Digite o segundo número inteiro:")
      leia(numero2)
      soma = numero1 + numero2
      escreval("A soma é",soma)
fim
```

Tipos de Dados

- Como podemos ver no algoritmo de exemplo, é necessário criar 3 variáveis para guardar os dados que vamos ler do usuário e para armazenar o valor da soma dos dados lidos;
- Estas variáveis são declaradas antes de serem usadas e o seu tipo não é alterado durante a execução do programa. Em algumas linguagens é obrigatório declarar o tipo e a variável antes de serem usadas pelo programa;
- Em outras linguagens, isso não é obrigatório.

Tipos de Dados

Linguagem fortemente tipada: é aquela que a declaração do tipo da variável é obrigatória. Exemplo: java, c, c++, ect..
Linguagem fracamente tipada: é aquela que pode alterar o tipo durante a execução do programa. Exemplo: php, python, ruby, javascript.
Linguagens não tipada: é aquela em que só existe um tipo genérico para todo o programa ou nem existe. Exemplo: perl
Linguagem de tipo estático: neste tipo de linguagem, o compilador deve conhecer o tipo antes da execução do programa. Exemplo: java, c, c++, etc.
Linguagem de tipo dinâmico: o tipo da variável é conhecido somente na execução do programa. Exemplo: php, ruby, python

Tipos de Dados

- A linguagem Java, assim como outras linguagens, possui um conjunto de tipos necessários para as construções básicas da linguagem;
- Este conjunto é chamado de **tipos primitivos** e na **prática** são **implementados por palavras-chave**;
- Cada tipo primitivo possui um tamanho de memória (em bits) que é usado para armazenar o seu valor;
  - Além disso, ele possui uma escala de valores, ou seja, possui um conjunto de valores específicos;
- Vejamos:

Tipos de Dados

Tipo	Tamanho (bits)	Faixa de Valores
int	4 bytes	-2.147.483.648 até 2.147.483.647
short	2 bytes	-32.768 até 32.767
byte	1 bytes	-128 até 127
long	8 bytes	-9.223.372.036.854.775.808 até 9.223.372.036.854.775.807
float	4 bytes	+/- 3.40282347E+38F (aproximadamente 7 dígitos significativos)
double	8 bytes	+/- 1.79769313486231570E+308 (15 dígitos significativos)
char	2 bytes	0 até 65.536
boolean	1 bit	True ou false

Tipos de Dados

TIPO	VALOR PADRÃO
BYTE	0
SHORT	0
INT	0
LONG	0L
FLOAT	0.0f
DOUBLE	0.0d
CHAR	'\u0000'
BOOLEAN	false

Valores padrões para os tipos em Java.

Tipos de Dados - Literais

- Um literal é uma representação de algum valor fixo que o programador literalmente estabelece;
- Cada tipo primitivo possui um literal correspondente e, na tabela 1.2, estão representados os literais padrões para cada tipo. Observe os exemplos a seguir;
- São todos exemplos de literais:

```
boolean resultado = true;
charcMaiusculo = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
```

Tipos de Dados - Literais

- Você percebeu, na tabela 1.2, que os valores long, float e double tem letras no final do valor?
- Estas letras servem para distinguir o tipo do literal em algumas situações;
  - Portanto, se 100 é um long, podemos representar como 100L ou 100l (é preferível usar o “L” maiúsculo);
  - Se o 100 não tiver uma letra no final, será considerado um int;

Tipos de Dados - Literais

- Os valores literais para byte, short, int e long podem ser criados a partir de um literal int;
- Valores do tipo long que excedem a capacidade de um int podem ser criados por meio de literais do tipo long;
- Os literais do tipo int podem ser decimais, binários, octais ou hexadecimais;
- O binário e o hexadecimal são sistemas de numeração usados em programas que lidam e manipulam endereços de memória, por exemplo;

Tipos de Dados - Literais

```
// O número 26 em decimal
int decimal = 26;
// Em hexadecimal
int hexa = 0x1a;
// A declaração octal começa com 0
int octal  =  032
// E em binário
intbinario = 0b11010;
```

Comentários em Java

- Existem três formas de se inserir comentários :
  - // Comentário em uma linha
  - /* Comentário em uma ou mais linhas */
  - /** Documento Comentários */

- Quando o comentário tipo 3 é colocado imediatamente acima da declaração (de uma função ou variável), indica que o comentário poderá ser incluído automaticamente em uma página HTML (gerado pelo comando javadoc – gerador de documentação do Java).

Tipos de Dados - Literais de ponto flutuante

- Um literal de ponto flutuante é do tipo float se terminar com a letra “F” ou “f ”, **caso contrário**, seu tipo **será double e pode terminar com “D” ou “d” opcionalmente**;
  - Os tipos de ponto flutuante (float e double) também podem ser expressos na notação científica usando a letra “E” ou “e”, “F” ou “f ” para um literal float e “D” ou “d” para um literal double. Observe os exemplos:

```
double d1 = 123.4;
// é o mesmo valor que d1 mas em notação cientifica
double d2 = 1.234e2;
float f1  = 123.4f;
```

Tipos de Dados - Literais Caracteres e String

- Os literais dos tipos char e String podem conter qualquer caractere Unicode;
- Dependendo do editor de texto que você estiver usando para programar, você pode digitar diretamente caracteres como “Ç”, “ã” e outros ou usar uma sequência chamada “escape para Unicode” para poder gerá-los;
  - Esta sequência são os caracteres \u. Veja os exemplos:

```
double d1 = 123.4;
// é o mesmo valor que d1 mas em notação cientifica
double d2 = 1.234e2;
float f1  = 123.4f;
```

Tipos de Dados - Literais Caracteres e String

- As strings em Java na verdade são instâncias de uma classe Java chamada String;
  - Existem várias maneiras de se representar uma string em Java e todas elas usam aspas duplas para a string desejada;
- Existe também o literal nulo, representado pela palavra-chave null;
  - O literal nulo representa ausência de valor e dado;
  - O literal nulo pode ser atribuído a qualquer variável exceto variáveis de tipos primitivos;

```
//exemplo de literal String
//observe que a classe String começa com S maiúsculo!

String s = "Exemplo de uma string";
```

Tipos de Dados – Declaração de Variáveis

- int x, y; //declarando duas variáveis inteiras
- x = 6; //atribuindo valores a variáveis
- y = 1000;
- float f = 3,141516f; //ponto flutuante
- double w = 3,2310834; //ponto flutuante - dupla precisão
- char ch = ‘a’; //Caractere
- final int MAX = 9; //Define a constante MAX com 9

Tipos de Dados – Escopo das Variáveis

- Escopo de uma variável indica em que parte do código ou bloco de comandos do programa que podemos utilizar ou enxergar a variável;
  - Existem variáveis locais e variáveis globais;
  - O escopo define também quando a variável será criada e destruída da memória;
  - As locais estão visíveis apenas dentro do bloco enquanto as globais estão disponíveis em qualquer bloco do programa;
- Observação: escopo é diferente de visibilidade, o qual se aplica as variáveis de classe e tem a ver com a utilização destas fora da classe;

Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

Operadores e expressões

- Assim como em qualquer linguagem de programação, uma expressão é uma unidade de código de programa que resulta em um valor por meio do uso de operadores e operandos;
- Normalmente podemos classificar **os operadores em três grupos:**
  - Aritméticos: quando ocorre algum cálculo matemático;
  - Relacionais: quando envolvem condições e comparações;
  - Lógicos: quando lidam com valores booleanos e quando queremos criar condições mais complexas.

Operadores Aritméticos

OPERAÇÃO	OPERADOR	EXEMPLO
Adição	+	2+2, a+b, a+2
Subtração	-	4-3, a-b, a-3
Divisão	/	8/5, a/b, b/4
Multiplicação	*	8*7, 8*a, a*b
Resto	%	8%4, a%b, a%2

Operadores aritméticos em Java.

Operadores Aritméticos

- Quando existir expressões com mais de um operador, pode-se usar parênteses, os quais têm prioridade maior do que a dos operadores;
  - Quando houver parênteses aninhados será realizada primeiro a expressão contida nos parênteses mais internos;
  - Caso exista mais de um operador e/ou grupo de parênteses em uma expressão, estes serão avaliados no sentido da esquerda para a direita;
  - No caso de parênteses aninhados, os parênteses mais internos são realizados em primeiro lugar;

Operadores Aritméticos

```
(32-2)/(2*10-(4+1))
→   →
A ordem de execução é:
     I.   (32-2)= 30
     II.  (4+1)= 5
     III.      2*10 = 20
     IV.  III-II = 20-5 = 15
     V.   I/IV = 30/15 = 2
```

Operadores Relacionais

OPERAÇÃO	OPERADOR	EXEMPLO
Igualdade	==	a==b, a==2
Diferença	!=	a!=b, a!=2
Maior	>	a>b, b>2
Menor	<	a<b, b<3
Maior ou igual	>=	a>=b, b>=3
Menor ou igual	<=	a<=b, b<=3

operadores relacionais em Java.

Operadores Relacionais

- Assim como nos operadores aritméticos, é possível usar parênteses para priorizar as partes desejadas em uma expressão;
- Além disso, existe uma precedência de operadores quando a expressão possuir vários operadores juntos:
  1. Os operadores >, <, >=, <= são aplicados primeiro;
  2. Os operadores ==, != são aplicados em seguida.
- Lembre-se de que, quando numa expressão houver vários destes operadores, assim como em qualquer linguagem de programação, a sequência de execução é da esquerda para a direita;

Operadores Lógicos

OPERAÇÃO	OPERADOR
E	&&
Ou	||
Negação	!
Ou exclusivo	^

Operadores lógicos em Java.

Operadores Lógicos

- As regras de precedência dos operadores lógicos são:
  1. A negação tem a maior precedência;
  2. Depois temos o “E” (&&);
  3. Depois o “Ou exclusivo” (^);
  4. E finalmente o “Ou” (||);

Curiosidade: além dos operadores && e || existem os operadores "&" e "|". Os operadores "&&" e "||" são mais rápidos que o "&" e "|" pois quando percebem que a resposta não mudará mais, eles param de verificar as outras condições. Por isso, eles são chamados de operadores de curto circuito (short circuit operators).

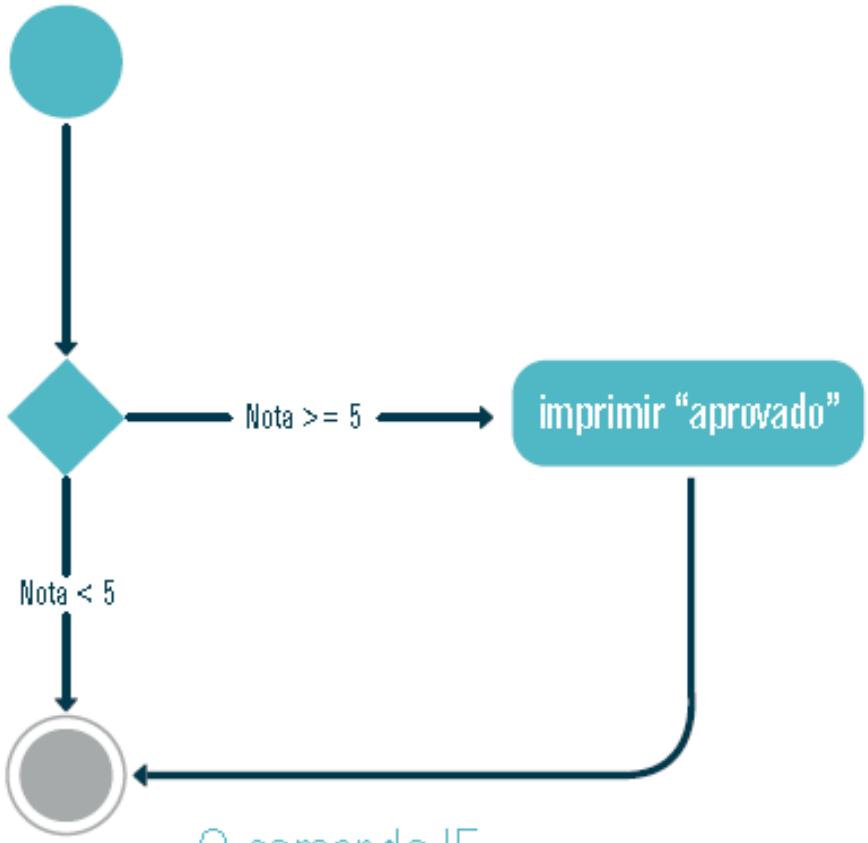
Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

Estruturas de Decisão

- As estruturas de decisão também são chamadas de estruturas de seleção ou condicionais e são executadas caso uma condição especificada seja verdadeira;
- Assim como outras linguagens, é possível construir algumas combinações deste tipo de estrutura;
- Para poder mostrar o funcionamento dos comandos em Java vamos usar o diagrama de atividades da UML;
- Desta forma, você pode entender visualmente como o comando é executado e compará-lo com o código em Java correspondente;

Estruturas de Decisão – IF-THEN



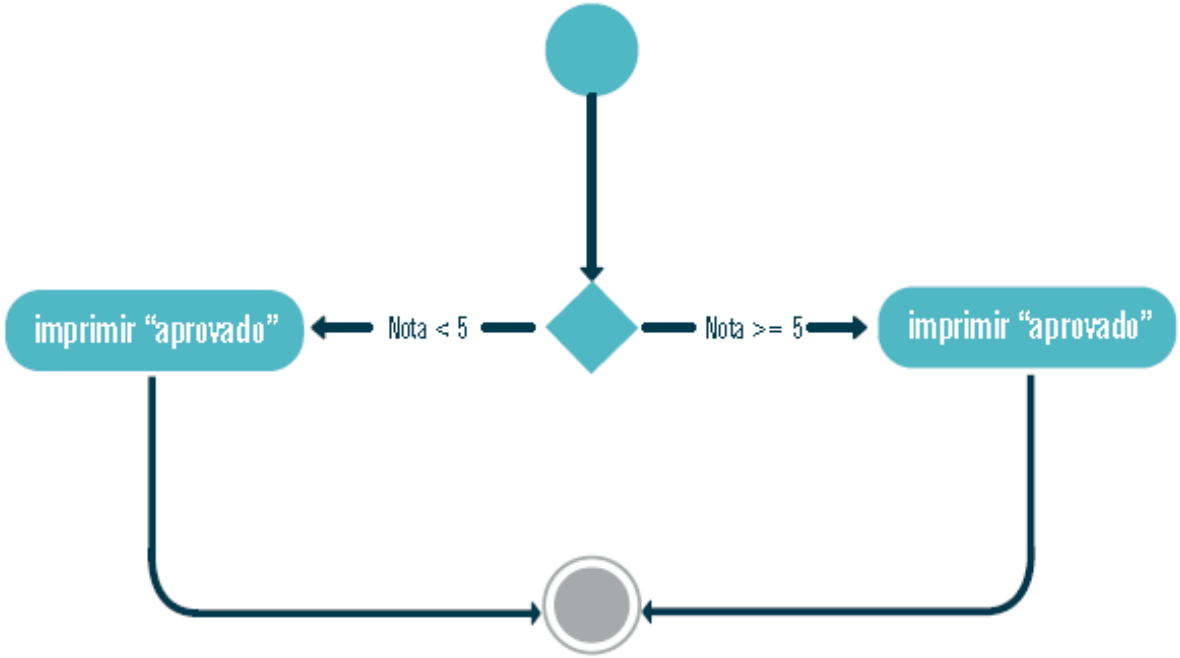
O comando IF

Estruturas de Decisão – IF-THEN

- Observe que, neste caso, temos apenas uma instrução que é executada se a condição do *if* for verdadeira;
  - É claro que é possível ter várias instruções sendo executadas, porém, neste caso, teremos que abrir um bloco, como no exemplo a seguir, onde temos 2 instruções dentro de um bloco:

```
if (nota>=5) {        // início do bloco. Use o { para abrir
    System.out.println("Aprovado!");
    System.out.println("Você pode agora imprimir o seu boletim.");
}             // fim do bloco. Use o } para fechar o bloco
```

Estruturas de Decisão – IF-THEN-ELSE



O IF-THEN-ELSE em Java.

Estruturas de Decisão – IF-THEN-ELSE

- O IF-THEN-ELSE em Java pode ser representado na figura 1.6. Podemos observar que só foi colocada uma instrução após a condição verdadeira e uma após a condição falsa;
- E, assim como ocorre no IF-THEN, é possível ter um bloco após cada condição;
- Notação ternária no JAVA:

```
System.out.println(nota >=5 ? “Aprovado” : “Reprovado”);
```

Podemos ler a instrução assim:
"Nota é maior ou igual a 5?
Caso verdadeiro, imprima "Aprovado",
em caso contrário (após o :), imprima "Reprovado")

Estruturas de Decisão – IF-THEN-ELSE

- Outra possibilidade é aninhar várias instruções IF-THEN ou IF-THENELSE, observe com atenção o código a seguir;

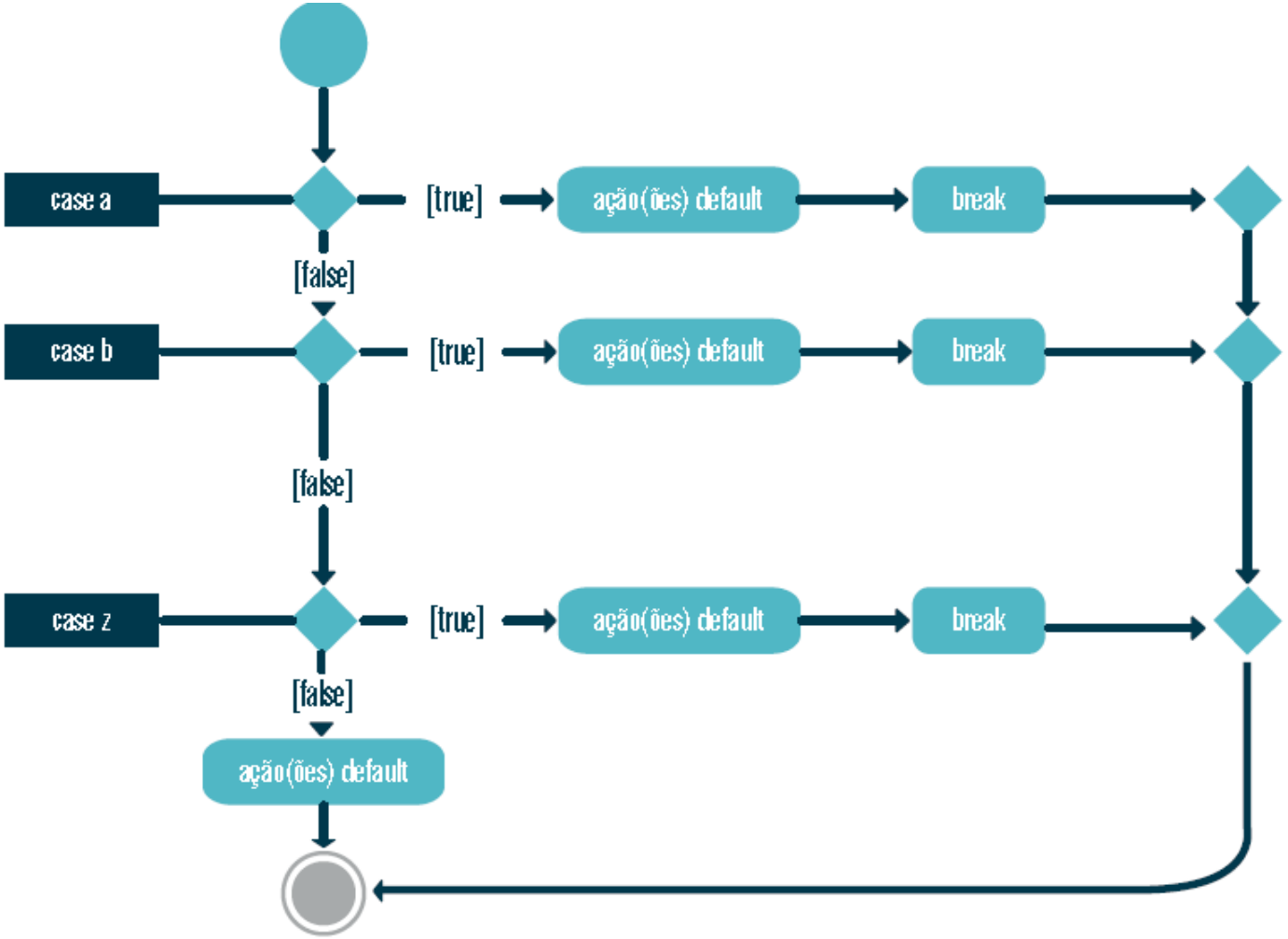
```
if (nota>=9)
    System.out.println("Conceito A");
else
    if (nota>=8)
        System.out.println("Conceito B");
    else
        if (nota>=7)
            System.out.println("Conceito C");
        else
                if (nota>=6)
                System.out.println("Conceito D");
                else {
                System.out.println("Conceito E");
                System.out.println("Você está reprovado!");
                }
```

Estruturas de Decisão – SWITCH

- Podemos perceber que o IF-THEN é uma estrutura do tipo seleção única, ou seja, se a condição existente for verdadeira, um bloco de código é executado e a estrutura é finalizada;
- No IF-THEN-ELSE a estrutura possui dupla seleção: se a condição for verdadeira, um bloco é executado ou senão o bloco correspondente à condição falsa será executado;
  - Porém o Java possui uma estrutura na qual uma instrução de seleção múltipla pode realizar diferentes ações baseadas nos possíveis valores de uma variável ou expressão;
  - Instrução switch possui sua estrutura geral mostrada na figura 1.7;

Comandos de Controle de Fluxo

Estruturas de Decisão – SWITCH



Instrução de seleção múltipla (SWITCH) com instruções break.

Comand

Es CH

```
1   public class Selecao {
2   public static void main(String[] args) {
3
4       int mes = 8;
5           String nomeMes;
6           switch (mes) {       //inicio do bloco
7               case 1:  nomeMes = "Janeiro";
8                        break;   //perceba o comando break no final de cada condição
9               case 2:  nomeMes = "Fevereiro";
10                       break;
11              case 3:   nomeMes = "Março";
12                       break;
13              case 4:  nomeMes = "Abril";
14                       break;
15              case 5:  nomeMes = "Maio";
16                       break;
17              case 6:  nomeMes = "Junho";
18                       break;
19              case 7:  nomeMes = "Julho";
20                       break;
21              case 8:  nomeMes = "Agosto";
22                       break;
23              case 9:  nomeMes = "Setembro";
24                       break;
25              case 10: nomeMes = "Outubro";
26                       break;
27              case 11: nomeMes = "Novembro";
28                       break;
29              case 12: nomeMes = "Dezembro";
30                       break;
31              default: nomeMes = "Mês inválido";
32                       break;
33      }                        //fim do bloco
34      System.out.println(nomeMes);
35  }
36 }
```

Estruturas de Decisão – SWITCH

- Desvia o fluxo natural do programa de acordo com o resultado de um teste lógico.

```
if (expressão booleana)
   comando1 ou {bloco de comandos1}
else
   comando2 ou {bloco de comandos2}
```

Estruturas de Decisão – SWITCH

- Quando existe um conjunto de opções, podemos utilizar a estrutura switch – case

```
switch(variável)
{       case(valor1):comando1; break;
        case(valor2):comando2; break;
        case(valor3):comando3; break;
....
        default:comando_genérico; break;
}
```

Desvios de Fluxo - SWITCH

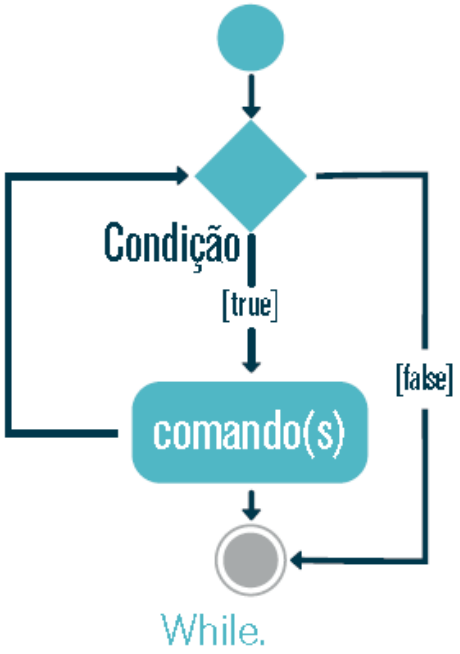
- Existem dois tipos de desvios de fluxo
  - Break
  - Continue;
- **break**; O comando termina a execução de um loop sem executar o resto dos comando e força a saída do laço.
- **continue**; O comando termina a execução de um laço sem executar o resto dos comandos, voltando para o início do laço, para uma nova iteração.

Estruturas de Repetição

- Estas estruturas permitem que uma ação possa ser executada várias vezes;
- Estas estruturas também são chamadas de laços ou loops;
  - Existem vários tipos de repetição e todos fazem basicamente a mesma coisa: repetir uma ação várias vezes (ou não);
  - Os tipos diferentes de repetição oferecem várias formas de determinar quando este irá começar ou terminar;
  - Existem situações em que é mais fácil resolver um problema usando um determinado tipo de laço do que outro;

Estruturas de Repetição - While

- O comando while executa repetidamente um bloco enquanto uma condição particular for verdadeira;
- A sintaxe do comando e o seu diagrama de atividade estão representados na figura 1.8;



While.

```
while   (condição)   {
    comandos (s)
}
```

Estruturas de Repetição - While

- O comando *while* avalia a condição a qual sempre retorna um valor *boolean* (*true ou false*);
  - Se a condição retorna *true*, o *while* executa o(s) comando(s) no bloco;
  - O *while* continua testando a condição e executando o bloco até que a condição seja false;

```
class Repete {
    public static void main(String[] args){
        int conta = 1;
        while (conta < 11) {
            System.out.println("Contando: " + conta);
            conta = conta+1;
        }
    }
}
```

Estruturas de Repetição – DO-While

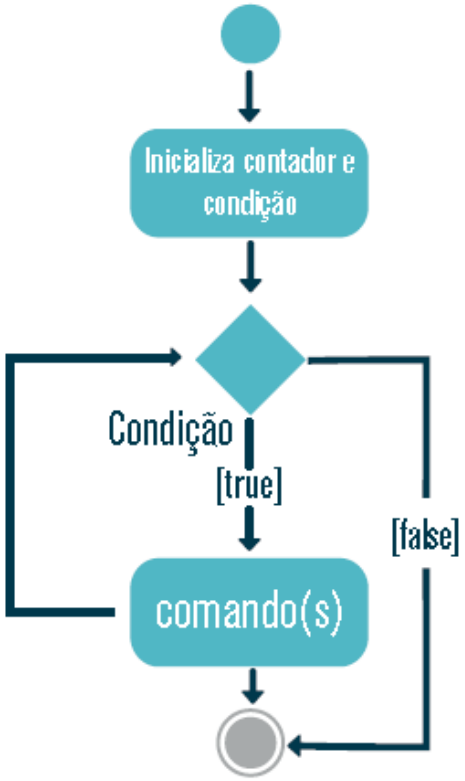
- A linguagem Java também possui um comando DO-WHILE, o qual possui a seguinte sintaxe:

```
do {
    comando(s)
} while (condição);
```

- A diferença entre o DO-WHILE e WHILE é que o DO-WHILE analisa a expressão no final da repetição ao invés do início, ou seja, pelo menos uma vez o bloco de comandos será executado;

Estruturas de Repetição – FOR

- O comando for é um meio compacto de fazer uma repetição sobre um intervalo de valores;



O comando FOR.

```
for (inicialização;término;incremento) {
    comando (s);
}
```

Estruturas de Repetição – FOR

- Algumas observações devem ser feitas para o comando for:
  - A expressão de inicialização começa o loop. É executada uma única vez assim que o loop começa; Quando a expressão de término é analisada e é falsa, o loop finaliza;
  - A expressão de incremento é chamada a cada interação dentro do loop; Normalmente é feito um incremento ou um decremento;

```
class Repete3 {
    public static void main(String[] args){
        for(int i=1; i<11; i++){
            System.out.println("Contando: " + i);
        }
    }
}
```

Estruturas de Repetição – FOR

- A estrutura de repetição “FOR” pode ser utilizada em coleções (vetores e classes *Collections*);
  - Esse tipo de for é chamado de *enhancedfor* e pode ser usado para fazer iterações dentro dessas estruturas de dados, percorrendo-as;

```
class EnhancedFor {
    public static void main(String[] args){
            //declarando um vetor inteiro contendo uma lista de números
        int[] numeros = {1,2,3,4,5,6,7,8,9,10};
            //o for abaixo percorre o vetor e mostra na tela cada um dos números
        for (int item : numeros) {
            System.out.println("Contando: " + item);
        }
    }
}
```

Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

Comandos de Controle de Fluxo

E/S - dados via console e com JOptionPanel

- Existem várias formas de se exibir informações na tela;
  - O que temos visto até aqui é o que chamamos de console, usando o println() ou print() da classe System.out;
- E para ler dados do console?

```
Scanner sc = new Scanner(System.in);
System.out.println("Qual e' o seu nome?");
String valor = sc.nextLine();
System.out.println("Oi "+valor+", tudo bem?");

Veja a execução deste programa:
Qual e' o seu nome?
Fabiano
Oi Fabiano, tudo bem?
```

E/S - dados via console e com JOptionPanel

- Veja outras formas de ler valores do console, por meio da classe *Scanner*:

```
float num = sc.nextFloat();
intnum = sc.nextInt();
bytebaite = sc.nextByte();
longlg = sc.nextLong();
booleanbool = sc.nextBoolean();
double num = sc.nextDouble();
Stringstr = sc.nextLine();
```

E/S - dados via console e com JOptionPanel

- Portanto, fazer a entrada de dados em Java pelo Console é simples e prático com a classe Scanner e o objeto System.in;
- Também é possível realizar entrada e saída em Java com diálogos da biblioteca Swing;
- Esta biblioteca possui uma classe chamada *JOptionPaneque* e é também muito utilizada para a entrada e saída de dados de usuários;

Comandos de Controle de Fluxo

Estácio

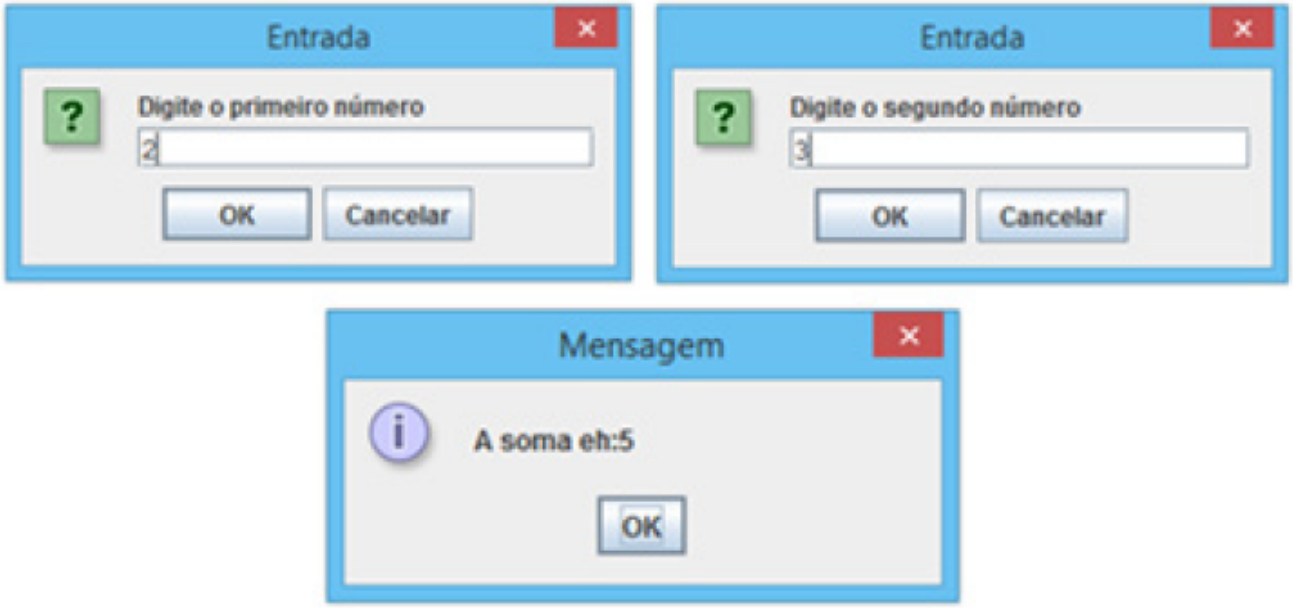
E/S - dados via console e com JOptionPanel

```
importjavax.swing.JOptionPane;

1    public classAdicao {
2        public static void main(String[] args) {
3            String num1, num2;
4            int n1, n2, soma;
5
6            num1 = JOptionPane.showInputDialog("Digite o primeiro número");
7            n1 = Integer.parseInt(num1);
8
9            num2 = JOptionPane.showInputDialog("Digite o segundo número");
10           n2 = Integer.parseInt(num2);
11
12           soma = n1+n2;
13
14           JOptionPane.showMessageDialog(null, "A soma eh:"+soma);
15       }
16 }
```

E/S - dados via console e com JOptionPanel

Novamente enfatizamos o estudo da api do java e neste caso especificamente a api do swing para poder saber as possíveis formas de uso da classe joptionpane e seus inúmeros (e úteis) métodos para criar diálogos com o usuário. Não deixe de ver o link: <http://docs.Oracle.Com/javase/6/docs/api/javax/swing/package-summary.html>.



E/S - dados via console e com JOptionPanel

- Perceba no programa que, para usar as classes *Swing*, é obrigatória a importação do pacote *javax.swing.JOptionPane;*
  - O método *showInputDialog()* serve para pedir uma entrada do usuário;
  - Esta entrada é sempre uma String e, como vamos somar as duas entradas (e fazer uma conta aritmética), é preciso converter a entrada para int;

E/S - dados via console e com JOptionPanel

- O método s*howMessageDialog()* possui vários construtores e foi usado na linha 14 a forma mais simples;
  - Observe que o segundo parâmetro do método é a *String* que desejamos mostrar na tela;
  - O primeiro parâmetro normalmente é usado com null, porém ele se refere ao container no qual o componente está inserido;
  - Caso o componente seja mostrado em outro container, ele deve ser especificado aqui;
  - Mas isso não ocorre com frequência, portanto, cuidado quando for usá-lo no caso de não ser *null;*

Agenda

- Tipos de Dados Primitivos;
  - Literais;
  - Literais de ponto flutuante;
  - Literais Caracteres e Strings;
- Constantes e variáveis;
- Operadores e expressões;
  - Operadores Aritméticos;
  - Operadores relacionais;
  - Operadores lógicos;
- Comandos de controle de fluxo;
  - Estruturas de decisão (IF/IF-THEN-ELSE/SWITCH);
  - Estruturas de repetição (WHILE/FOR);
- Entrada e Saída de dados via console e com JOptionPanel;
- Conversão simples de tipos;
  - Conversão de strings para números;
  - Convertendo números para strings;

Conversão de Tipos

- Qualquer linguagem de programação possui várias formas de se fazer conversões entre seus tipos suportados, e Java não é diferente;
- Porém, as conversões mais simples que podemos ter são as **conversões entre números e strings**;
- Os links a seguir mostram onde encontrar mais referências sobre a linguagem java: <http://www.Oracle.Com/technetwork/java/api-141528.html>: especificações de todas as api java <http://docs.Oracle.Com/javase/7/docs/api/>: especificação da api do java 7;

Conversão de Tipos – Strings para Números

- É muito comum armazenar dados numéricos em objetos *string;*
  - O CPF, por exemplo, é um caso;
  - Todo CPF é um número com 11 dígitos e, dependendo do programa que é desenvolvido, é necessário fazer operações aritméticas com este dado;
  - Porém dependendo do programa, ao obter o valor do CPF do usuário pelo teclado, o valor é guardado em uma string sendo necessária sua conversão posteriormente;

Conversão de Tipos - Strings para Números

- Para tratar deste tipo de situação, a classe Number possui subclasses que envolvem os tipos primitivos numéricos (*Byte, Integer, Double, Float, Longe Short* – perceba que todas essas são classes, veja as iniciais maiúsculas, e não os tipos primitivos);
- **Cada uma dessas classes possui um método de classe chamado valueOf() que converte uma string em um objeto daquele tipo**;

Conversão de Tipos - Strings para Números

```
1   public class TesteValueOf {
2       public static void main(String[] args) {
3           if (args.length == 2) {
4               // converte strings emnumeros
5           float a = (Float.valueOf(args[0])).floatValue();
6           float b = (Float.valueOf(args[1])).floatValue();
7
8               // algumas contas
9           System.out.println("a + b = " + (a+b));
10          System.out.println("a - b = " + (a-b));
11          System.out.println("a * b = " + (a*b));
12          System.out.println("a / b = " + (a/b));
13          System.out.println("a % b = " + (a%b));
14          } else {
15              System.out.println("Digite dois números:");
17          }
18      }
19 }
```

Conversão de Tipos - Números para Strings

- Também é frequente converter um número para uma string;
  - Há várias formas de se fazer isso e vamos mostrar uma delas.

```
int i;
// Concatena "i" com uma string vazia; a conversão é feita “na mão”, usando o “+”
String s1 = "" + i;
ou
// O método de classe valueOf().
String s2 = String.valueOf(i);
```

- Cada uma das classes Number possui um método toString() para converter um número em uma string;
- Veja o exemplo a seguir:

Conversão de Tipos - Números para Strings

```
public class TestaToString {

    public static void main (String[] args) {
        double d = 858.48;
        String s = Double.toString(d);

        int ponto = s.indexOf('.');

        System.out.println(ponto + " dígitos " + "antes do ponto decimal.");
        System.out.println((s.length()-ponto-1)+"   dígitos   depois   do   pon-
to decimal.");
    }
}
```

Exercícios

1. Escreva um programa que leia 3 valores e diga qual o maior valor digitado;
2. Construa um programa que verifica se o número é par ou ímpar;
3. Imprima todos os números de 150 até 300;
4. Imprima a soma de 1 até 1000;
5. O custo ao consumidor de um carro novo é a soma do custo de fábrica com a percentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que a percentagem do distribuidor seja de 28% e os impostos de 45%, escreva um algoritmo que leia o custo de fábrica de um carro e escreva o custo ao consumidor;

Exercícios

06. Faça um programa em Java que verifique se os clientes de uma loja excederam o limite do cartão de crédito. Para cada cliente, temos os seguintes dados:

– número da conta corrente

– saldo no início do mês

– total de todos os itens comprados no cartão

– total de créditos aplicados ao cliente no mês

– limite de crédito autorizado

Todos esses dados são inteiros. O programa deve mostrar o novo saldo de acordo com a seguinte fórmula (saldo inicial + despesas – créditos) e determinar se o novo saldo excede o limite de crédito. Para aqueles clientes cujo novo saldo excedeu, o programa deve mostrar a frase: "Limite de crédito excedido".

Exercícios

07. Considere o seguinte fragmento de código:

```
if (umNumero >= 0)
    if (umNumero == 0)
        System.out.println("Primeira string");
    else
        System.out.println("Segunda string");
    System.out.println("Terceira string");
```

a) O que você acha que será impresso se *umNumero* = 3?

b) Escreva um programa de teste contendo o código acima; assuma que umNumero = 3. Qual a saída do programa? Foi o que você respondeu na questão a? Explique a saída; em outras palavras, qual é o fluxo de controle do fragmento do código?

c) Usando somente espaços e quebras de linha, reformate o fragmento para torná-lo mais legível.

d) Use parênteses, colchetes, chaves e o que for necessário para deixar o código mais claro.

Exercícios

08. Uma pesquisa sobre algumas características físicas da população de uma determinada região coletou os seguintes dados, referentes a cada habitante, para serem analisados:

- sexo (masculino, feminino);
- cor dos olhos (azuis, verdes, castanhos);
- cor dos cabelos (louros, castanhos, pretos);
- idade em anos.

Para cada habitante, foi perfurado um cartão com esses dados, e o último cartão, que não corresponde a ninguém, conterá o valor da idade igual a -1. Implementar um algoritmo que determine e escreva:

a) a maior idade dos habitantes;
b) porcentagem de indivíduos do sexo feminino cuja idade esteja entre 18 e 35 anos, inclusive, e que tenham olhos verdes e cabelos louros;

Exercícios

09. Uma pesquisa sobre algumas características físicas da população de uma determinada região coletou os seguintes dados, referentes a cada habitante, para serem analisados:

- sexo (masculino, feminino);
- cor dos olhos (azuis, verdes, castanhos);
- cor dos cabelos (louros, castanhos, pretos);
- idade em anos.

Para cada habitante, foi perfurado um cartão com esses dados, e o último cartão, que não corresponde a ninguém, conterá o valor da idade igual a -1. Implementar um algoritmo que determine e escreva: a) a maior idade dos habitantes; b) porcentagem de indivíduos do sexo feminino cuja idade esteja entre 18 e 35 anos, inclusive, e que tenham olhos verdes e cabelos louros