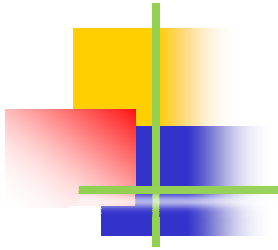


deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Programação 1

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

`http://moodle.ua.pt/`

Aula 1

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- Apresentação da disciplina
- Organização de um computador
- Desenvolvimento de um programa
- Conceitos base da linguagem JAVA
  - Estrutura de um programa
  - Tipos de dados
  - Variáveis e constantes
  - Operadores e expressões
  - Classes da linguagem JAVA
  - Leitura e escrita de dados
  - Escrita formatada

Objetivos

- Compreensão clara, ainda que elementar, do que é um computador, como funciona, para que serve, que limitações tem e como se comunica com ele.
- Desenvolvimento de estratégias para a especificação precisa do problema que se pretende pôr o computador a resolver.
- Estabelecimento de métodos para descrição detalhada e rigorosa de soluções que possam ser implementadas num computador.
- Aprendizagem de uma linguagem de programação (JAVA).
- Familiarização com um ambiente de desenvolvimento onde os programas possam ser escritos, documentados, testados e validados.

Programa

- Introdução à Linguagem JAVA: elementos
- Estruturas de controlo: instruções decisórias
- Estruturas de controlo: instruções repetitivas
- Programação procedimental (Funções)
- Criação de novos tipos de dados (Registos)
- Sequências de caracteres (Strings)
- Sequências (Arrays)
- Ficheiros de texto
- Pesquisa e ordenação
- Sequências de tipos-referência (Arrays de Strings e de registos; Arrays bi-dimensionais)

Organização das aulas

- Aulas teórico-práticas:
  - apresentação dos temas da disciplina;
  - aulas baseadas em slides e exemplos que serão colocados on-line no final da “semana”;
  - não é permitido o uso de computador;
  - o objetivo dos dois últimos pontos é levar os alunos a aprenderem a tomar notas nas aulas.
- Aulas práticas:
  - Aplicação dos conhecimentos à resolução de problemas concretos;
  - filosofia base: “só se aprende fazendo”.

Bibliografia

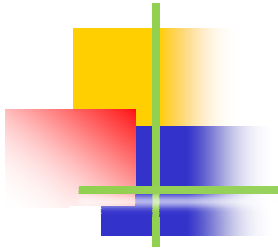
- António Adrego da Rocha, Osvaldo Rocha Pacheco, "Introdução à Programação em Java", 1ª edição, FCA editores, 2009.

Bibliografia complementar

- Elliot B. Koffman, "Problem Solving with JAVA", Addison Wesley.
- João Pedro Neto, "Programação e Estruturas de Dados", Escolar Editora.
- Kris Jamsa, "Programação em JAVA", Edições CETOP.
- F. Mário Martins, "JAVA 5 e Programação por Objectos", FCA.
- J. Brookshear, "Computer Science, An overview", Addison Wesley.
- Y. Daniel Liang, "Introduction JAVA Programming", Pearson, Prentice-Hall.

Avaliação

- A disciplina tem avaliação discreta com quatro momentos de avaliação à componente prática:
  - MT1, 10%, inicio da aula prática (semana de 16 e 21 de Outubro);
  - TPI, 30%, 6 de Novembro (definido em Conselho Pedagógico)
  - MT2, 10%, inicio da aula prática (semana de 11 e 16 de Dezembro);
  - EP, 50%, época de exames.
- A frequência das aulas práticas é obrigatória para todos os alunos ordinários.
- Os trabalhadores-estudantes serão avaliados nos mesmos moldes.
- O exame prático de recurso vale 100% da nota.
- Notas finais superiores a 17 poderão ter de ser defendidas.



Feita a apresentação…

Aula 1

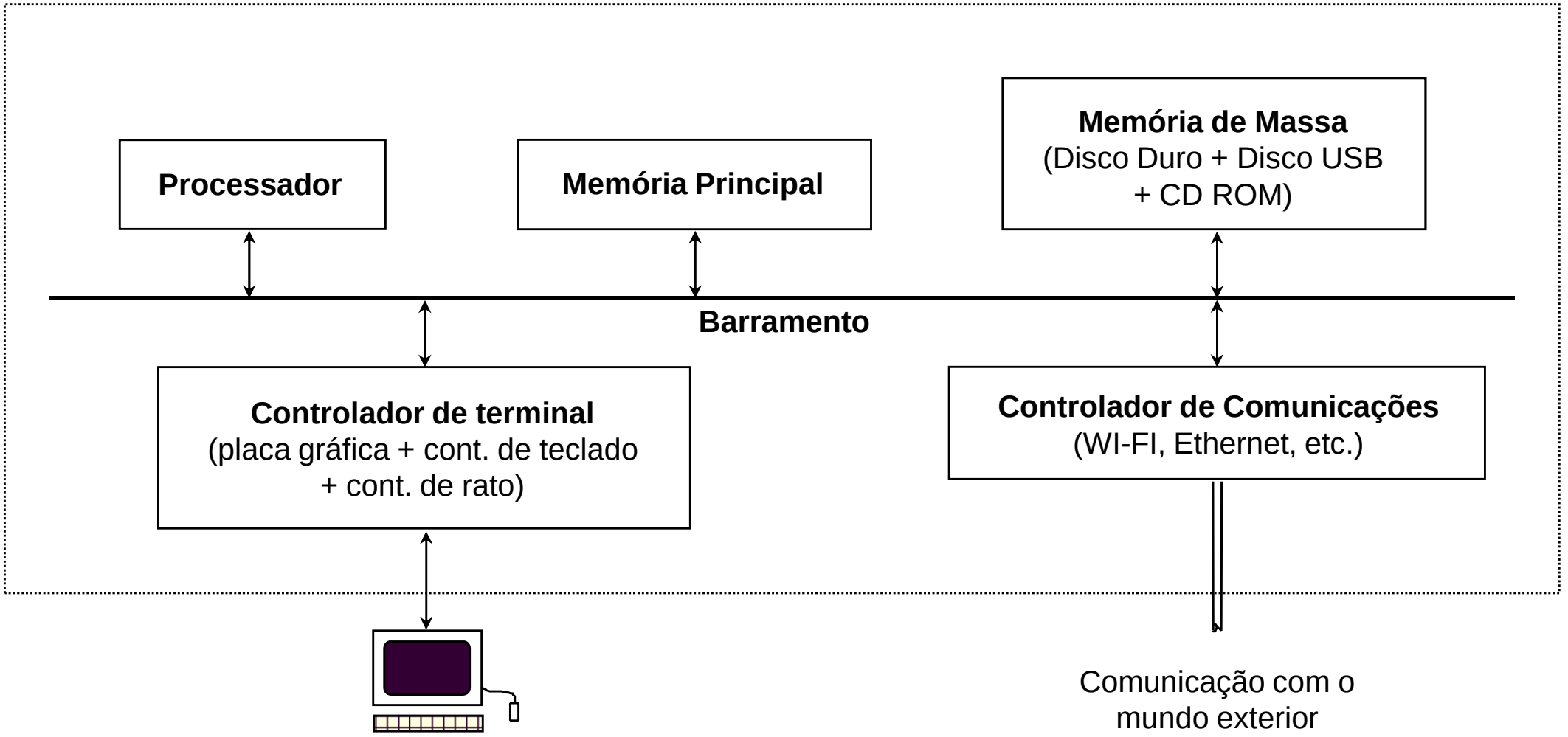
O computador e os elementos básicos da linguagem JAVA

Computador…

- Máquina programável que processa informação de forma autónoma.
- Executa, com uma cadência muito rápida, sequências de operações elementares sobre informação recebida, devolvendo ao utilizador resultados.
- A sequência de operações elementares, designada habitualmente por **programa**, pode ser alterada ou substituída por outra, sempre que se deseje.
- Durante a execução do programa, a sequência de operações elementares e os valores temporários produzidos estão armazenados num dispositivo interno, chamado memória.

Organização de um computador

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática



Organização de um computador

- O computador utiliza tecnologia e lógica binária (valor ‘0’ ou ‘1’).
- Todos os dados (números inteiros, reais, texto, etc.) são armazenados em *bits*. Um conjunto de 8 bits corresponde a um *byte*.
- A memória do computador organiza-se em endereços (normalmente com um identificador associado) e dados :

Endereços	**“Identificador”**	**Dados**	***Significado***
0xFF0000	idade	0011…1001	40
0xFF0001	peso	1001…0101	34.50
…	…	…	…
0xFF00FE	fimDeCiclo	0000…0000	false
0xFF00FF	msg	1101…1001	‘Olá’

Homem Vs. Computador

Homem

a abordagem é criativa

- aprende com a experiência passada;
- associa conceitos distintos, conseguindo isolar elementos comuns;
- usa em larga medida um raciocínio de tipo indutivo (intuição);

propõe soluções

- descobre métodos de resolução;

comete erros

- as inferências produzidas são muitas vezes incorrectas;
- está sujeito a lapsos de concentração provocados por cansaço.

Computador

a abordagem é não criativa

- não tem capacidade directa de aprendizagem;
- só associa conceitos cuja afinidade foi previamente estabelecida;
- usa mecanismos de raciocínio dedutivo;

não propõe soluções

- possibilita a validação das soluções encontradas;

não comete erros

- salvo avaria, limita-se a executar de um modo automático a sequência de operações estabelecida.

Tipos de problemas que o computador resolve

Problemas completamente especificados:

- as variáveis de entrada e de saída estão perfeitamente identificadas;
- se conhece uma **solução**; ou seja, um método que permite obter, de forma unívoca, os valores das variáveis de saída em função dos valores das variáveis de entrada;
- deve considerar-se sempre a resolução dos problemas no âmbito mais lato possível; ou seja, deve considerar-se a resolução de classes de problemas e não de problemas particulares;
- a gama de valores permitida para as variáveis de entrada deve ser claramente estabelecida;
- a solução descrita deve contemplar alternativas para toda a gama de valores das variáveis de entrada, eliminando toda e qualquer ambiguidade.

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

Conversão de distâncias (milhas para Km)

- Dada uma distância, expressa em milhas, que é lida do teclado, convertê-la para quilómetros e escrevê-la no ecrã do computador (terminal).

Variável de entrada:

MILHAS (distância expressa em milhas)
valor numérico positivo ou nulo

Variável de saída:

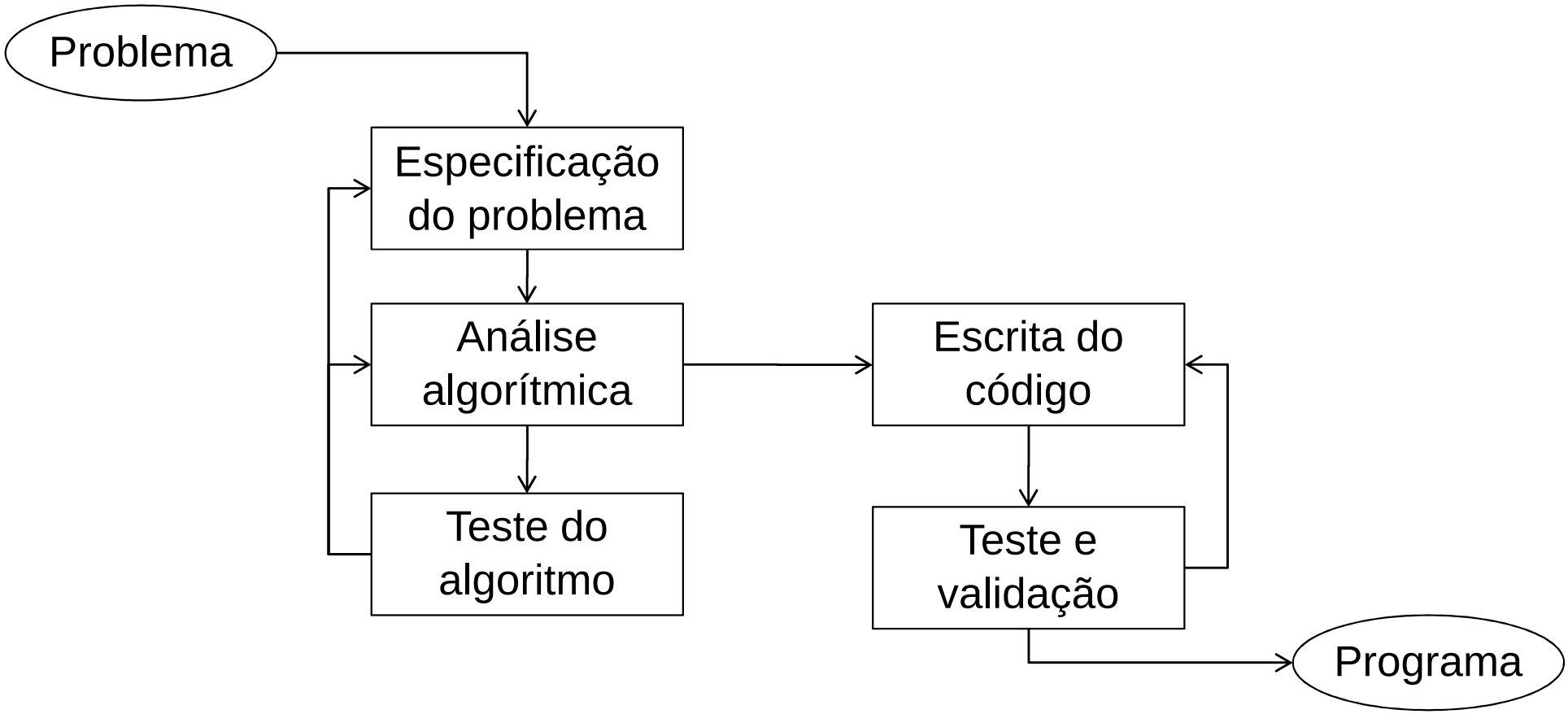
KILOMETROS (distância expressa em quilómetros)
valor numérico representado com 3 casas decimais

Solução:

KILOMETROS = 1.609 * MILHAS

Fases de desenvolvimento de um programa

- As duas etapas básicas do desenvolvimento de um programa são a **análise do problema** e a **implementação da aplicação**.



Algoritmo

- Designa-se por **algoritmo** a descrição detalhada e rigorosa da solução do problema.
- A transcrição do algoritmo para uma linguagem de programação dá origem ao **programa**.
- Supõe-se que o conjunto de operações descrito no algoritmo é realizado segundo uma ordem pré-estabelecida: só se inicia uma dada operação, quando a anterior estiver terminada - **execução sequencial**.
- Exemplo:

  leitura dos valores das variáveis de entrada

  processamento

  escrita dos valores das variáveis de saída

Estrutura de um programa


```
inclusão de classes externas
public class Programa
{
  declaração de constantes e variáveis globais
  public static void main (String[] args)
  {
    declaração de constantes e variáveis locais
    sequências de instruções
  }
}
definição de tipos de dados (registos)
```

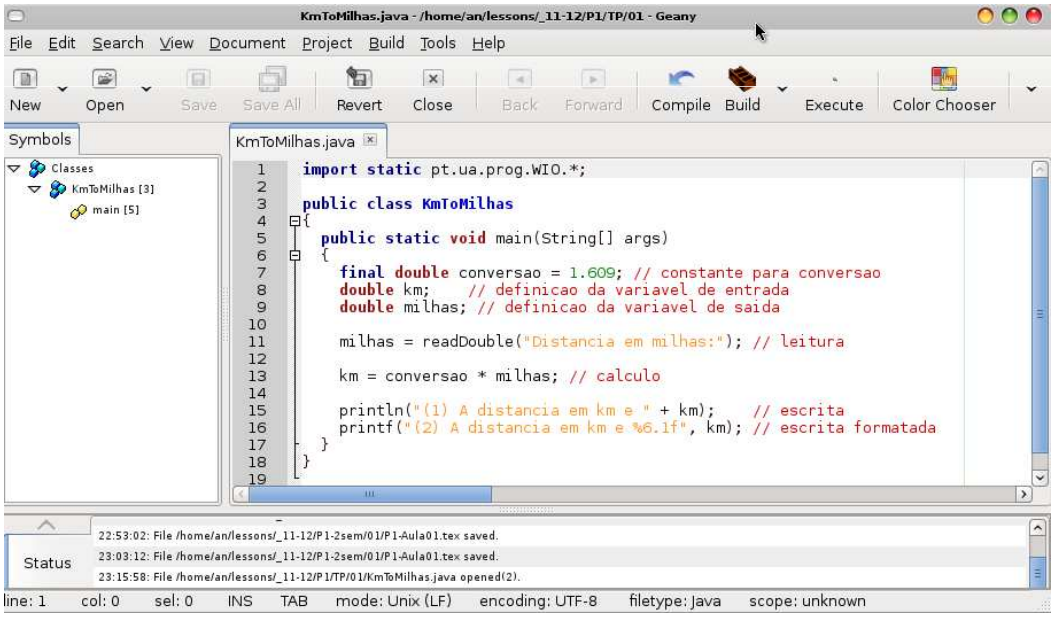
Exemplo de um programa

Ficheiro KmToMilhas.java

```
import java.util.Scanner;
public class KmToMilhas{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    double km, milhas;
    System.out.print("Distancia em milhas:");
    milhas = sc.nextDouble();
    km = 1.609 * milhas;
    System.out.println("A distancia em km é " + km);
  }
}
```

Desenvolvimento de um programa

- **Edição**:
  - geany KmToMilhas.java



- **Compilação**
  - javac KmToMilhas.java
- **Execução**
  - java KmToMilhas

Elementos básicos da linguagem JAVA

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- **Palavras reservadas** – símbolos que têm um significado bem definido em JAVA e que não podem ser usadas para outro fim (ex. `class, break, switch, final, if, then, else, while`, ...).
- **Identificadores** – nomes utilizados para designar todos os objectos existentes num programa. Devem começar por uma letra ou por símbolo '_' e só podem conter letras, números e o símbolo '_' (ex. `nome, idade, i, j, cont_1, dia_mes, res, _km` ...).
- **Comentários** – melhoram a legibilidade de um programa (todos os caracteres na mesma linha que se seguem ao símbolos '`//`' e blocos `/* comentários (podem ser várias linhas) */`).
- **Constantes** – "valor específico" de um certo tipo (ex. `10, -10, 5.5, .5, -0.8, "Aveiro", true`, ...).
- **Operadores e separadores** – símbolos ou combinações de símbolos que especificam operações e usados na construção de instruções: `()[]{}<>;.,:?!'"&|=+-*/%~^#\_$`

Tipos de dados primitivos

- `byte, short, int, long` - números inteiros (`10, -10, 0, ...`)
- `float, double` - números reais (`10.5, -10.5, .2, ...`)
- `boolean` – apenas dois valores possíveis (`true, false`)
- `char` – caracteres (`'a', '1', '!', ...`)
- Definição de uma variável:

```
tipo identificador;
```

- Exemplos:
  - `double peso, altura, largura, erro;`
  - `int idade, dia_mes, ano;`
  - `boolean resultado;`
  - `char letra, op;`
  - `final double PI = 3.1415; //definição de constante real`
  - `final int LIMITE = 100; //outra constante inteira`

Inicialização de variáveis

- Uma variável (posição de memória no PC) pode ser considerada como uma caixa cujo conteúdo inicialmente não está definido.
- Antes de uma variável poder ser utilizada deve ser-lhe atribuído um valor:
  - na altura da definição

```
double num = 10.5;
int idade = 18;
```

  - usando uma instrução de atribuição (símbolo '=')

```
double peso;
…
peso = 50.5;
```

  - lendo um valor do teclado ou de outro dispositivo (ex. ficheiro)

```
double milhas;
…
milhas = sc.nextDouble("Valor real:");
```

Conversões

- Sempre que uma expressão tenha operandos aritméticos de tipos diferentes, os operandos com menor capacidade de armazenamento são automaticamente convertidos para o tipo com maior capacidade:

```
byte -> short (ou char) ->
int -> long -> float -> double
```

- A conversão inversa não é admitida e gera um erro de compilação.
- Podemos sempre forçar uma conversão através de um operador de conversão (*cast* em inglês):

```
double x;
int y;
y = (int)x; // estamos a forçar a conversão para int
```

Operadores e expressões

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- Operadores:
  - Aritméticos: `*, /, +, -, %`
  - Relacionais: `<, <=, >, >=, ==, !=`
  - Lógicos: `!, ||, &&`
  - Manipulação de bits: `&, ~, |, ^, >>, <<`
- Expressões:

```
int x, z;
double y;
x = 10 + 20; //o valor 30 é armazenado em x
y = 8.4 / 4.2; //o valor 2.0 é armazenado em y
```

  - As expressões são calculadas da esquerda para a direita.
  - Atenção às prioridades dos operadores e aos parênteses.

Operadores aritméticos unários

- simétrico: `- (-x)`
- incremento de 1: `++ (++x, x++)`
- decremento de 1: `-- (--x, x--)`
- Os operadores unários de incremento e decremento só podem ser utilizados com variáveis e atualizam o seu valor de uma unidade.
- Colocados antes são pré-incremento e pré-decremento. Neste caso a variável é primeiro alterada antes de ser usada.
- Colocados depois são pós-incremento e pós-decremento e neste caso a variável é primeiro usada na expressão onde está inserida e depois atualizada.

Algumas classes da linguagem JAVA

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- A linguagem java disponibiliza um vasto conjunto de classes que permitem manipular dados e realizar diversas operações.
- Serão apresentadas conforme forem sendo necessárias. Ficam três exemplos:
- Classe `Math`:
  - `double Math.cos(double);`
  - `double Math.acos(double);`
  - `double Math.sin(double);`
  - `double Math.asin(double);`
  - `double Math.sqrt(double);`
  - `double Math.pow(double, double);`
  - `double Math.toRadians(double);`
- Classe `Integer e Double`:
  - `Integer.MAX_VALUE` `Double.MAX_VALUE`
  - `Integer.MIN_VALUE` `Double.MIN_VALUE`

Leitura e escrita de dados

- Leitura do teclado (classe `Scanner`)
  - `import java.util.Scanner;`
  - `nextInt(), nextDouble(), nextLine(), ...`
  - Exemplos

```
Scanner sc = new Scanner(System.in);
integer x;
x = sc.nextInt();
```

- Escrita no terminal (classe `PrintStream` - `System.out`)
  - `print(), println(), printf();`
  - Exemplos:

```
System.out.print("O valor de x é " + x); // não muda de linha
System.out.println("O valor de x é " + x); // muda de linha
System.out.printf("O valor de x é %3d\n", x); // formatada
```

Escrita formatada

- A função `printf` permite escrever informação formatada.

```
System.out.printf("formato de escrita", lista de variáveis);
```

- O formato de escrita é uma sequência de caracteres, que pode conter especificadores de conversão.
- O especificador de conversão é composto pelo símbolo % seguido de um caracter que indica qual o tipo de dados que queremos escrever:

```
%d, %f, %c, %s, …
```

- Este caracter pode ser precedido de um número com o qual se controla o formato:

```
%3d, %5.1f, %3c, %10s, …
```

- Exemplo:

```
System.out.printf("Int.: %6d", 15);      // Int.: _ _ _ _ 1 5
System.out.printf("Real: %6.2f", 14.2); // Real: _ 1 4 . 2 0
```