Ficheiros de Texto

Algoritmia e Java

Sumário Geral

Sumário

Enquadramento de Ficheiros

- **Estruturas de Dados**
  - **Simples**
    - Armazenam
      - Um dado de cada vez
      - Em memória principal (RAM) // memória volátil
    - Tipos
      - INTEIRO
      - REAL
      - BOOLEANO
      - CARATER
  - **Complexas**
    - Armazenam
      - Múltiplos dados ao mesmo tempo
    - Tipos
      - Array // guarda conjunto de dados do mesmo tipo ... em memória RAM
        - Vetor // dados organizados de forma linear
        - Matriz // dados organizados em linhas e colunas
      - **Ficheiro** // guarda conjunto de dados ... em memória secundária
        - Texto
        - Binário

Noção de Ficheiro

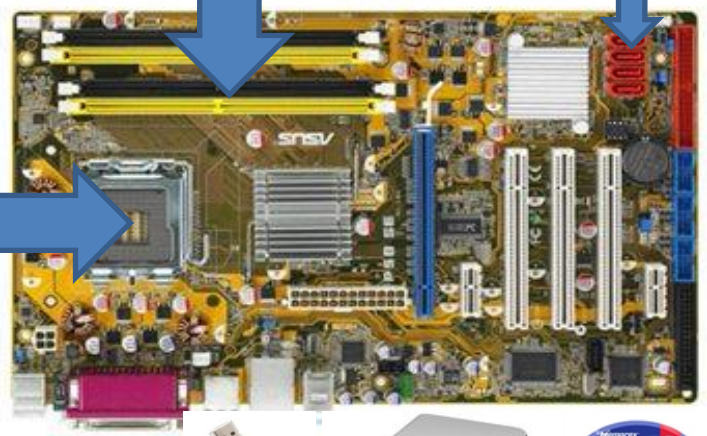
- **Estrutura de Dados Complexa**
- **Armazena**
  - Sequência de Dados
    - Comprimento arbitrário
    - Dados de tipos diferentes
  - Modo Permanente

Dado 1	Dado 2	...	Dado N

// ⇒ em memória secundária

Memória Principal
(RAM)

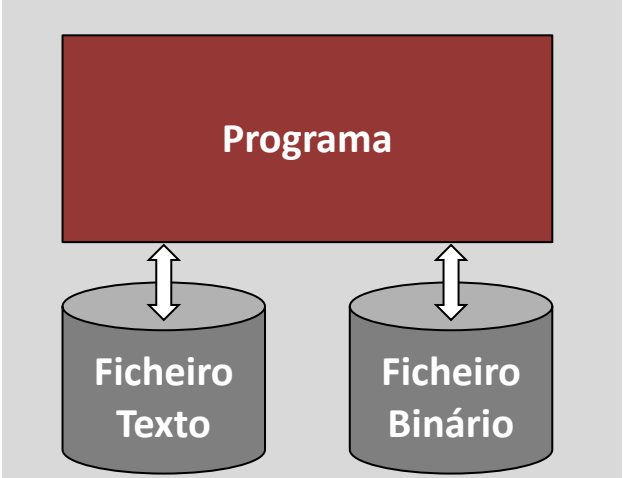
Memória Secundária
(Disco Interno)



Processador
(CPU)

Memória Secundária
(*PenDrive*, Disco Externo, CD, DVD)

Tipos de Ficheiro



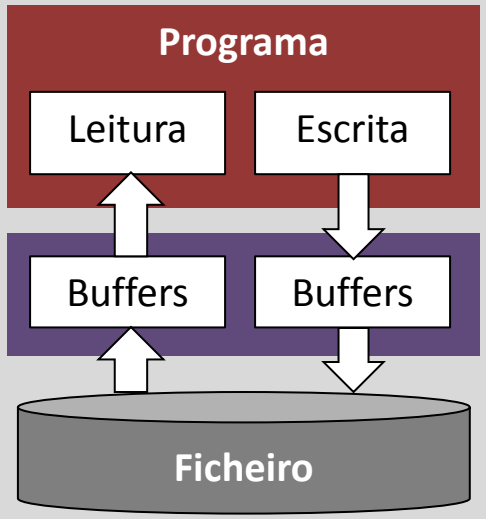
Tipos de Ficheiro

Tipos de Ficheiro	Sequência	Formato dos Dados	Processadores de Texto	Interesse
Texto	Carateres (Tabelas de Carateres: Unicode, ASCII, etc.)	Texto	Legível	Guardar Dados Públicos
Binário	Códigos Binários	Binário	Ilegível	Guardar Dados Privados

Manipulação de Ficheiros

- **Procedimento Geral**

1. Declaração de Variável de Ficheiro // para identificar o ficheiro
2. Abertura de Ficheiro // atribuição do ficheiro à variável
   // alocar recursos do Sistema Operativo para leitura/escrita
   // Ex: buffers
3. Leitura / Escrita de Ficheiro
4. Fecho de Ficheiro // libertar recursos alocados



Buffers

- Memórias temporárias
- Guardam dados transferidos entre programas e ficheiros
- Interesse
  - Permitir leituras e escritas mais rápidas, devido à enorme diferença entre as velocidades do processador e do disco

Sumário Geral

Sumário

Declaração de Variável de Ficheiro

- **Algoritmia**
  - Sintaxe: FICHEIRO varFich
  - Exemplo

```
ED
    FICHEIRO f
INÍCIO
    ...
FIM
```

- **Java**

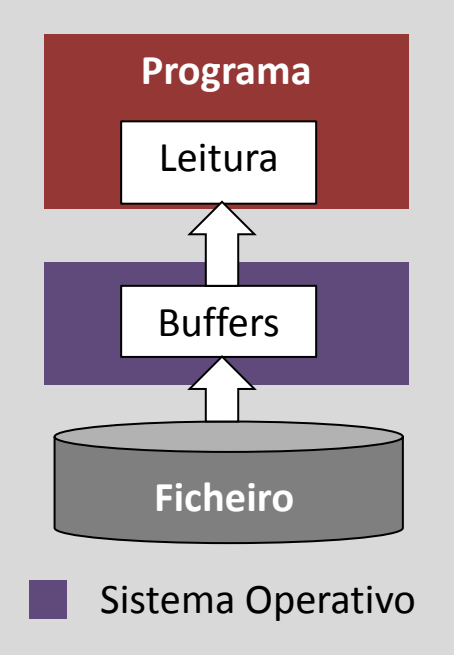
  File varFich;

```
import java.io.File;
public class DemoFicheiro {
    public static void main(String[] args) throws Exception {
        File f;
        ...
    }
}
```

Sumário

Procedimento de Leitura

1. Declaração de Variável de Ficheiro
2. Abertura de Ficheiro para Leitura
3. Leitura de Ficheiro
4. Fecho de Ficheiro



Abertura de Ficheiro para Leitura

- **Algoritmia**
  - Sintaxe (do procedimento)

    abrirFicheiroLeitura(varFich, nomeFich)

```
ED
   FICHEIRO f1, f2
INÍCIO
   // ficheiro num dado diretório
   abrirFicheiroLeitura( f1 , "c:\nomes.txt" )
   // ficheiro no diretório corrente
   abrirFicheiroLeitura( f2 , "notas" )
FIM
```

- **Java**

```
// criar e guardar objeto representativo do ficheiro
varFich = new File("nomeFich");
// criar e guardar objeto para ler o ficheiro
Scanner obj = new Scanner( varFich );

// Simplificação: evita variável do tipo File
Scanner obj = new Scanner( new File("nomeFich") );
```

```
import java.io.File;
import java.util.Scanner;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      File f = new File("c:\\nomes.txt");
      Scanner fi1 = new Scanner( f );
      Scanner fi2 = new Scanner(new File("c:/equipas.txt"));
      Scanner fi3 = new Scanner(new File("c:\\jogos"));
      Scanner fi4 = new Scanner(new File("d:\\aprog\\int"));
      // ficheiro no diretório corrente  (ou pasta projecto)
      Scanner fi5 = new Scanner(new File("notas"));
   }
}
```

Abertura de Ficheiro para Leitura

- **Java**
  - Leitura de carateres portugueses acentuados
    - Exemplos: ç, ã, ê

// criar e guardar objeto **para ler** um ficheiro
Scanner obj = new Scanner(new File("nomeFich"), "ISO-8859-1");

```
import java.io.File;
import java.util.Scanner;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      File f = new File("c:\\nomes.txt");
      Scanner fi1 = new Scanner( f , "ISO-8859-1");
      Scanner fi2 = new Scanner(new File("c:/equipas.txt"), "ISO-8859-1");
      Scanner fi3 = new Scanner(new File("c:\\jogos", "ISO-8859-1"));
      Scanner fi4 = new Scanner(new File("d:\\aprog\\ficheiros.txt"), "ISO-8859-1");
      // ficheiro no diretório corrente  (ou pasta projecto)
      Scanner fi5 = new Scanner(new File("notas"), "ISO-8859-1");
   }
}
```

Leitura de Ficheiro

- **Algoritmia**
  - Leitura de uma linha/palavra
    - Sintaxe

```
TEXTO linha, palavra
linha ← lerLinhaSeguinte( varFich )
palavra ← lerPalavraSeguinte( varFich )
```

  - Indicação de Fim de Ficheiro
    - Sintaxe

```
BOOLEANO fim
fim ← fimFicheiro( varFich )
```

- **Java**

```
String linha, palavra
linha = objetoScanner.nextLine();
palavra = objetoScanner.next();
```

```
boolean fim;
fim = objetoScanner.hasNextLine();
fim = objetoScanner.hasNext();
```

```
ED
   FICHEIRO f
INÍCIO
   abrirFicheiroLeitura( f , "c:\nomes.txt" )
   // leitura de ficheiro linha-a-linha
   ENQUANTO (NAO  fimFicheiro( f )) FAZER
      ESCREVER( lerLinhaSeguinte(f) )
   FENQ
FIM
```

```
import java.io.File;
import java.util.Scanner;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      Scanner fi = new Scanner(new File("notas"));
      // leitura de ficheiro linha-a-linha
      while ( fi.hasNextLine() ){
         System.out.println( fi.nextLine() );
      }
   }
}
```

Fecho de Ficheiro

- **Algoritmia**
  - Sintaxe

    fecharFicheiro(varFich)

```
ED
   FICHEIRO f
INÍCIO
   // ficheiro no diretório corrente
   abrirFicheiroLeitura( f , "notas" )
   // leitura do ficheiro linha-a-linha
   ENQUANTO (NAO  fimFicheiro( f )) FAZER
      ESCREVER( lerLinhaSeguinte(f) )
   FENQ
   fecharFicheiro(f)
FIM
```

- **Java**
  - Sintaxe

    objetoScanner.close();

```
import java.io.File;
import java.util.Scanner;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      // ficheiro no diretório corrente (ou pasta projeto)
      Scanner fi = new Scanner(new File("notas"));
      // leitura do ficheiro linha-a-linha
      while ( fi.hasNextLine() ){
         System.out.println( fi.nextLine() );
      }
      fi.close();
   }
}
```

Exemplo de Leitura num Módulo/Método

- Determinar o número de linhas de um ficheiro de texto

- Algoritmia

```
DEFINIR INTEIRO numLinhas()

ED   INTEIRO n
     FICHEIRO f
INÍCIO
   abrirFicheiroLeitura( f, "c:\nomes.dat" );
   n ← 0
   ENQUANTO ( NÃO fimFicheiro(f) ) FAZER
        n ← n + 1
        lerLinhaSeguinte( f )
   FENQUANTO
   fecharFicheiro( f )
   RETORNAR n
FDEF
```

- Java

```
private static int numLinhas()  throws Exception {

        int n=0;
        File f ;

        f = new File( "c:\\nomes.dat" );
        Scanner fin = new Scanner( f );
        while ( fin.hasNextLine() ){
           n++;
           fin.nextLine();
        }
        fin.close();
        return n;
}
```

Classe String : método split

- **Interesse**
  - Decompor uma string em partes separadas por delimitadores: iguais ou diferentes
- **Exemplo: Delimitadores Iguais**

```
String registo = "joão/1961/10/1";
```

- **Obtenção dos campos do registo**

```
String[] campos;                    // vetor para guardar partes da string registo (campos)
campos = registo.split("/");        // 1. Decompõe o registo pelo delimitador /
                                    // 2. Cria um vector de strings (comprimento = nº partes)
                                    // 3. Guarda as partes em elementos diferentes
```

- **Resultado**

campos	"joão"	"1961"	"10"	"1"
	0	1	2	3

- **Algoritmia**

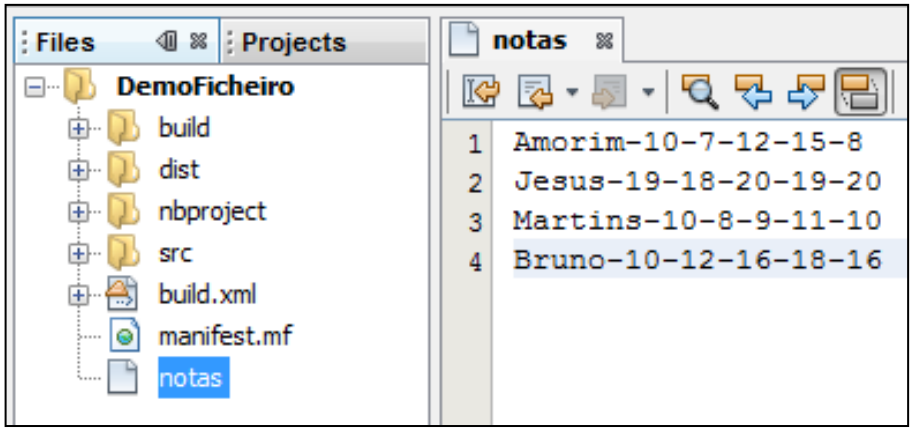
```
ED   TEXTO registo, campos[ ]
INÍCIO
   registo ← "joão/1961/10/1"
   campos ← separar(registo, "/")
   ...
FIM
```

- **Exemplo: Delimitadores Diferentes**

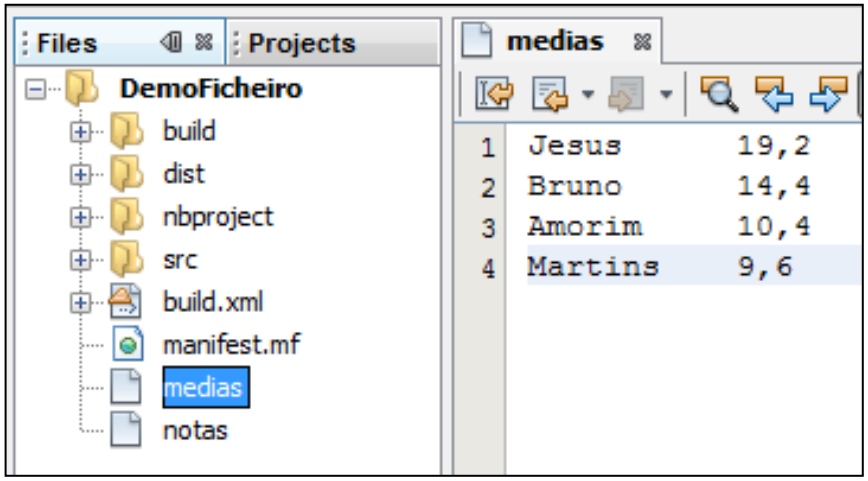
```
String s = "5-6+7-8";
String[] c = s.split("[+-]");
// c[0]="5", c[1]="6", c[2]="7", c[3]="8"
```

Classe String : método split

- **Exemplo**
  - Lê ficheiro de texto, chamado notas, contendo nomes e notas de alunos separados por um traço



  - Escreve noutro ficheiro de texto, chamado medias, os nomes dos alunos e as respetivas médias por ordem decrescente e separados por espaços



Classe String : método split


```
ED
        INTEIRO n
        REAL medias[ ]
        TEXTO nomes[ ]
INÍCIO
        n ← numAlunos()
       criar medias[n]
       criar nomes[n]
       preencherVetores(nomes, medias);
       ordenarVetores(nomes,medias);
       escreverFicheiro(nomes,medias);
FIM

DEFINIR INTEIRO numAlunos()
ED
        INTEIRO n
        FICHEIRO f
INÍCIO
        n ← 0
        abrirFicheiroLeitura( f, "notas") )
        ENQUANTO ( NÃO fimFicheiro( f ) ) FAZER
                n ← n + 1
                lerLinhaSeguinte( f )
        }
        fecharFicheiro( f );
        RETORNAR n
FDEF
```

continua

```
import java.io.File;
import java.util.*;

public class DemoFicheiro {

   public static void main(String[] args) throws Exception {

      int n = numAlunos();

      float[] medias = new float[n];

      String[] nomes = new String[n];

      preencherVetores(nomes, medias);

      ordenarVetores(nomes,medias);

      escreverFicheiro(nomes,medias);
   }

   private static int numAlunos() throws Exception {

      int n = 0;

      Scanner fin = new Scanner(new File("notas"));

      while (fin.hasNextLine()) {

         n++;

         fin.nextLine();
      }

      fin.close();

      return n;
   }
```

continua

Classe String : método split


```
DEFINIR ordenarVetores(TEXTO[] nomes, REAL[] med)
ED    INTEIRO i, j
      REAL tmp1
      TEXTO tmp2
INÍCIO
   PARA ( i ← 0  ATÉ comprimento(med)-2 PASSO 1)
      PARA ( j ← i+1  ATÉ comprimento(med)-1 PASSO 1)
         SE (med[j]>med[i]) ENTÃO
               tmp1 ← med[i]
               med[i] ← med[j]
               med[j] ← tmp1
               tmp2 ← nomes[i]
               nomes[i] ← nomes[j]
               nomes[j] ← tmp2
         FSE
      FPARA
   FPARA
FDEF
```

continua

```
private static void ordenarVetores(String[] nomes, float[]
med) {
   for (int i = 0; i < med.length-1; i++) {
      for (int j = i+1; j < med.length; j++) {
         if(med[j]>med[i]){
            float tmp = med[i];
            med[i]= med[j];
            med[j]=tmp;
            String tmp2 = nomes[i];
            nomes[i]=nomes[j];
            nomes[j]=tmp2;
         }
      }
   }
}
```

continua

Classe String : método split


```
DEFINIR preencherVetores( TEXTO[ ] nomes, REAL[ ] med )
ED    INTEIRO i, j, soma
      FICHEIRO f
      TEXTO linha, partes[ ]
INÍCIO
      abrirFicheiroLeitura( f, "notas“ )
      i ← 0
      ENQUANTO (NÃO fimFicheiro( f ) ) FAZER
         linha ← lerLinhaSeguinte( f  )
         partes ← separar( linha, "-“ )
         nomes[i] ← partes[0]
         soma ← 0
         PARA ( j ←  1 ATÉ comprimento(partes)-1 PASSO 1 )
            soma ← soma +  INT( partes[j] )
         FPARA
         med[i] ← soma / (comprimento( partes )-1)
         i ← i + 1
      FENQ
      fecharFicheiro( f )
FDEF
```

continua

```
private static void preencherVetores( String[] nomes,
 float[] med ) throws Exception {
    Scanner fin = new Scanner(new File("notas"));
    int i=0;
    while (fin.hasNextLine()) {
       // lê e guarda a linha seguinte
       String linha = fin.nextLine();
       // decompõe a linha em partes delimitadas por -
       String[] partes =linha.split("-");
       nomes[i]=partes[0];
       int soma = 0;
       for (int j = 1; j < partes.length; j++) {
          soma += Integer.parseInt(partes[j]);
       }
       med[i]=(float)soma/(partes.length-1);
       i++;
    }
    fin.close();
 }
```

continua

Classe String : método split

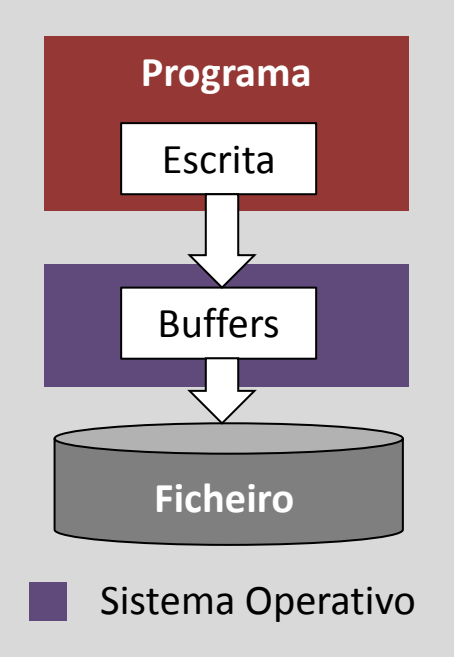

```
DEFINIR escreverFicheiro(TEXTO[] nomes, REAL[] med)
   ED    FICHEIRO f
         INTEIRO i
         TEXTO linha
   INÍCIO
      abrirFicheiroEscrita( f, "medias")
      PARA (i ← 0 ATÉ comprimento(nomes)-1 PASSO 1 ) {
         linha ← nomes[i] + " " + med[i]
         escreverLinha(f, linha);
      FPARA
      fecharFicheiro(f);
FDEF
```

```
   private static void escreverFicheiro(String[] nomes, float[]
    med) throws Exception {
      Formatter fout = new Formatter(new File("medias"));
      for (int i = 0; i < med.length; i++) {
         fout.format("%-10s %.1f%n", nomes[i],med[i]);
      }
      fout.close();
   }
}
```

Sumário

Procedimento de Escrita

1. Declaração de Variável de Ficheiro
2. Abertura de Ficheiro para Escrita
3. Escrita de Ficheiro
4. Fecho de Ficheiro



Abertura de Ficheiro para Escrita

- **Algoritmia**
  - Sintaxe (do procedimento)

    abrirFicheiroEscrita(varFich, nomeFich)

```
ED
   FICHEIRO f
INÍCIO
   // ficheiro num dado diretório
   abrirFicheiroEscrita( f , "c:\nomes.txt" )
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "notas" )
FIM
```

- **Java**

```
// criar e guardar objeto representativo do ficheiro
varFich = new File("nomeFich");
// criar e guardar objeto para escrever no ficheiro
Formatter obj = new Formatter( varFich );

// Simplificação : evita variável do tipo File
Formatter obj = new Formatter( new File("nomeFich") );
```

```
import java.io.File;
import java.util.Formatter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      File f = new File("c:\\nomes.txt");
      Formatter fo1 = new Formatter( f );
      Formatter fo2 = new Formatter(new File("c:\\nomes.txt"));
      Formatter fo3 = new Formatter (new File("c:/equipas.txt"));

      // ficheiro no diretório corrente (ou pasta do projeto)
      Formatter fo4 = new Formatter (new File("notas"));
   }
}
```

criado sempre novo ficheiro

Escrita no Ficheiro

- **Algoritmia**
  - Leitura de uma linha/palavra
    - Sintaxe

      TEXTO linha

      escreverLinha(varFich , linha)

```
ED
   FICHEIRO f
   INTEIRO i
INÍCIO
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "numeros" )
   // escreve 10 linhas
   PARA (i ← 1 ATÉ 10 PASSO 1) FAZER
      escreverLinha(f, INT(ALEATORIO()*10))
   FPARA
FIM
```

- **Java**

```
objetoFormatter.format(String format, args);
// permite escrita formatada
// semelhante ao System.out.printf() e String.format()
// consultar slides Java - Classes
```

```
import java.io.File;
import java.util.Formatter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      // ficheiro no diretório corrente (ou pasta do projeto)
      Formatter fo = new Formatter (new File("numeros"));
      // escreve 10 linhas
      for(int i =1; i<11; i++){
         fo.format("%d%n", (int)(Math.random()*10));
      }
   }
}
```

Fecho de Ficheiro

- **Algoritmia**
  - Sintaxe

    fecharFicheiro(varFich)

```
ED
   FICHEIRO f
   INTEIRO i
INÍCIO
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "numeros" )
   // escreve 10 linhas
   PARA (i ← 1 ATÉ 10 PASSO 1) FAZER
      escreverLinha(f, INT(ALEATORIO()*10))
   FPARA
   fecharFicheiro(f)
FIM
```

- **Java**
  - Sintaxe

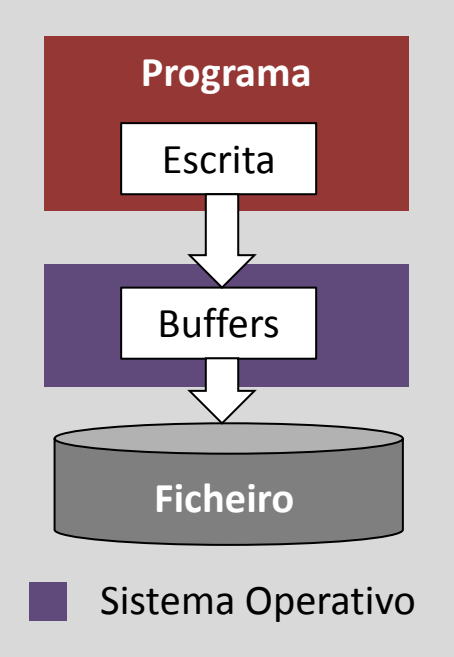
    objetoFormatter.close();

```
import java.io.File;
import java.util.Formatter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      // ficheiro no diretório corrente (ou pasta projeto)
      Formatter fo = new Formatter (new File("numeros"));
      // escreve 10 linhas
      for(int i =1; i<11; i++){
         fo.format("%d%n", (int)(Math.random()*10));
      }
      fo.close();
   }
}
```

Sumário

Procedimento de Escrita

1. Declaração de Variável de Ficheiro
2. Abertura de Ficheiro para Escrita e Atualização
3. Escrita no Fim de Ficheiro
4. Fecho de Ficheiro



Abertura de Ficheiro para Escrita e Atualização

- **Algoritmia**
  - Sintaxe (do procedimento)

```
abrirFicheiroEscrita(varFich, nomeFich)
```

```
ED
   FICHEIRO f
INÍCIO
   // ficheiro num dado diretório
   abrirFicheiroEscrita( f , "c:\nomes.txt" )
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "notas" )
FIM
```

- **Java**

```
// criar e guardar objeto representativo do ficheiro
varFich = new File("nomeFich");
// criar e guardar objeto para escrever no ficheiro,
   acrescentando novas linhas no fim do ficheiro
FileWriter obj = new FileWriter( varFich , true);
// Simplificação: evita variável do tipo File
FileWriter obj = new FileWriter(new File("nomeFich"),true);
```

- Adicionar linha no fim do ficheiro existente
- FileWriter não formata linha

```
import java.io.File;
import java.io.FileWriter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      File f = new File("c:\\nomes.txt");
      FileWriter fo1 = new FileWriter( f, true );
      FileWriter fo2=new FileWriter (new File("c:\\nomes.txt"),true);
      FileWriter fo3=new FileWriter (new File("c:/equipas.txt"),true);
      // ficheiro no diretório corrente (ou pasta do projeto)
      FileWriter fo4 = new FileWriter (new File("notas"),true);
   }
}
```

Escrita no Ficheiro

- **Algoritmia**
  - Leitura de uma linha/palavra
    - Sintaxe

      TEXTO linha

      escreverLinha(varFich , linha)

```
ED
   FICHEIRO f
   INTEIRO i
INÍCIO
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "numeros" )
   // escreve 10 linhas
   PARA (i ← 1 ATÉ 10 PASSO 1) FAZER
      escreverLinha(f, INT(ALEATORIO()*10))
   FPARA
FIM
```

- **Java**

```
objetoFileWriter.write(...);
```

```
import java.io.File;
import java.io.FileWriter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      // ficheiro no diretório corrente (ou pasta do projeto)
      FileWriter fo = new FileWriter (new File("numeros"), true);
      // acrescenta 10 linhas no final do ficheiro
      for(int i =1; i<11; i++){
         fo.write((int)(Math.random()*10));      // não formatada
      }
   }
}
```

Fecho de Ficheiro

- **Algoritmia**
  - Sintaxe

    fecharFicheiro(varFich)

```
ED
   FICHEIRO f
   INTEIRO i
INÍCIO
   // ficheiro no diretório corrente
   abrirFicheiroEscrita( f , "numeros" )
   // escreve 10 linhas
   PARA (i ← 1 ATÉ 10 PASSO 1) FAZER
      escreverLinha(f, INT(ALEATORIO()*10))
   FPARA
   fecharFicheiro(f)
FIM
```

- **Java**
  - Sintaxe

    objetoFileWriter.close();

```
import java.io.File;
import java.io.FileWriter;
public class DemoFicheiro {
   public static void main(String[] args) throws Exception {
      // ficheiro no diretório corrente (ou pasta projeto)
      FileWriter fo = new FileWriter (new File("numeros"),true);
      // escreve 10 linhas
      for(int i =1; i<11; i++){
         fo.write( (int)(Math.random()*10));
      }
      fo.close();
   }
}
```

Sumário Geral

Sumário

Representação

- **Valores**
  - Tipo char
  - Categorias
    - Letras: 'A' , 'B', ..., 'a','b', ..., 'ã', 'ç', ...
    - Dígitos: '1' , ..., '9',
    - Símbolos: '{' , ']', '@', '♥', ...
- **Representação Computacional**
  - Códigos Numéricos
  - Tabela de Carateres
    - UTF-16
      - U - Unicode
      - T - Transformation
      - F - Format
      - 16 bits = 2 bytes
- **Representação Java**
  - 'A' ⇔ 65 ⇔ '\u0041'
    - \u é carater especial (u = unicode)
  - Exemplos
    - char c;
    - c = 'A'; ⇔ c = 65; ⇔ c = '\u0041';

Representação de Carateres Java

Constantes char	Código Unicode	
	Hexadecimal	Decimal
'A'	'\u0041'	65
'B'	'\u0042'	66
...	...	...
'Z'	'\u005A'	90
...	...	...
'a'	'\u0061'	97
'b'	'\u0062'	98
...	...	...
'z'	'\u007A'	122
...	...	...
'♥'	'\u2665'	9829
...	...	...
'0'	'\u0030'	48
'1'	'\u0031'	49

Carateres Portugueses Acentuados

Constantes char	Código Unicode	
	Hexadecimal	Decimal
'À'	'\uc380'	192
'Á'	'\uC381'	193
'Â'	'\uC382'	194
'Ã'	'\uC383'	195
...	...	...
'Ç'	'\uC387'	199
...	...	...
'É'	'\uC389'	201
...	...	...
'Í'	'\uC38D'	205
...	...	...
'Ó'	'\uC393'	211
'Ô'	'\uC394'	212
'Õ'	'\uC395'	213
...	...	...
'Ú'	'\uC39A'	218

Constantes char	Código Unicode	
	Hexadecimal	Decimal
'à'	'\uc3A0'	224
'á'	'\uC3A1'	225
'â'	'\uC3A2'	226
'ã'	'\uC3A3'	227
...	...	...
'ç'	'\uC3A7'	231
...	...	...
'é'	'\uC3A9'	233
...	...	...
'í'	'\uC3AD'	237
...	...	...
'ó'	'\uC3B3'	243
'ô'	'\uC3B4'	244
'õ'	'\uC3B5'	245
...	...	...
'ú'	'\uC3BA'	250

Classe Character

- **Package**
  - java.lang
- **Documentação**
  - http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Character.html
- **Interesse**
  - Fornece métodos para:
    - Determinar Categoria de um Carater
      - Letra
        - Minúscula
        - Maiúscula
      - Dígito
    - Converter Carateres
      - Minúsculas para maiúsculas
      - Vice-versa
    - Determinar Valor
      - Carateres do tipo dígito

Classe Character

- **Métodos de Classe**
  - boolean **isLetter**(char **c**) // Verifica se o carater **c** é uma letra
  - boolean **isDigit**(char **c**) // Verifica se o carater **c** é um dígito
  - boolean **isSpaceChar**(char **c**) // Verifica se o carater **c** é o carater espaço
  - boolean **isUpperCase**(char c) // Verifica se o carater **c** é um carater maiúsculo
  - boolean **isLowerCase**(char c) // Verifica se o carater **c** é um carater minúsculo
  - char **toUpperCase**(char c) // Converte o carater **c** num carater maiúsculo
  - char **toLowerCase**(char c) // Converte o carater **c** num carater minúsculo
  - int **getNumericValue**(char c) // Retorna o valor int que o carater **c** representa
    // Retorna -1 se o carater **c** não representa um int
- **Exemplos**

```
char c = 'A';
if ( Character.isLetter(c) )
   System.out.println(c + " é uma letra");

char minusculo = Character.toLowerCase(c);

String s = "1T2X";
int digito = Character.getNumericValue(s.charAt(0));        // digito = 1
int codigo = s.charAt(0);                                   // codigo = 49
```

Sumário Geral

Sumário

Introdução

- **Package**
  - java.io
- **Documentação**
  - http://docs.oracle.com/javase/1.5.0/docs/api/java/io/File.html
- **Interesse**
  - Representação Abstrata de *Pathname*
    - Ficheiro // Ex: File fich = new File("d:/aprog/algoritmia.txt");
    - Diretório // Ex: File dir = new File("c:/Windows");
- **Objeto File**
  - Imutável
    - *Pathname* do ficheiro/diretório representado não pode ser alterado
- **Construtor**
  - Sintaxe
    - File(String pathname) // *pathname* de ficheiro/diretório
  - Funcionalidade
    - Cria objeto representativo de um *pathname* de ficheiro/diretório

Métodos de Instância

- boolean **delete**() // Elimina ficheiro/diretório com esse *pathname* (representado pelo objeto File)

```
Exemplo
    File f = new File("c:/texto.txt");
    if( f.delete() )
       System.out.println("Ficheiro eliminado");
```

- boolean **exists**() // Verifica se existe ficheiro/diretório com esse *pathname*

```
Exemplo
    File f = new File("c:/texto.txt");
    if( !f.exists() )
       System.out.println("Ficheiro não encontrado");
```

- boolean **isDirectory**() // Verifica se esse *pathname* é de diretório
- boolean **isFile**() // Verifica se esse *pathname* é de ficheiro
- int **length**() // Retorna o comprimento do ficheiro com esse *pathname*
  // Ficheiros de texto: comprimento = nº de carateres

Métodos de Instância

- String[] **list**() /* Retorna vetor com nomes dos ficheiros e diretórios do diretório com esse *pathname* */
- File[] **listFiles**() // Retorna vetor com *pathnames* de ficheiros do diretório com esse *pathname*
- boolean **mkdir**() // Cria diretório com esse *pathname*