53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

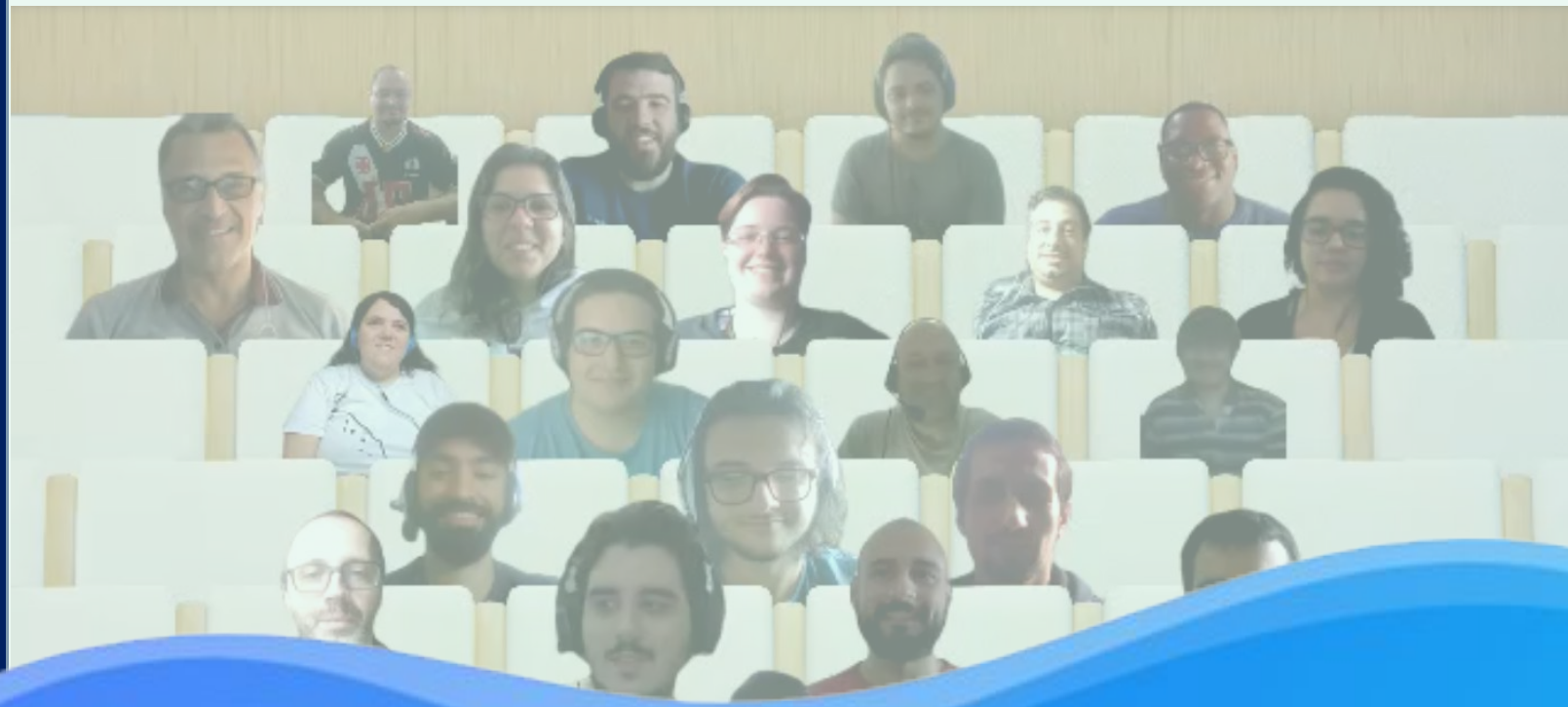
RESIDÊNCIA DE

SOFTWARE

CAPACITAR
TREINAR
EMPREGAR
TRANSFORMAR

Firjan SENAI

serratec
Parque Tecnológico Região Serrana



Trabalhar com arquivos e fluxo de dados

Data: 26/04/2022

- Datas
  - Faça um programa para ler uma data no formato dd/MM/yyyy e apresente ela no formato yyyyMMdd

```
public class MainFormatData {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String data = in.next();
        in.close();
        DateTimeFormatter formatadorBarra = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        DateTimeFormatter formatadorSaida = DateTimeFormatter.ofPattern("yyyyMMdd");

        LocalDate dataLd = LocalDate.parse(data, formatadorBarra);
        System.out.println(dataLd.format(formatadorSaida));
    }
}
```

ARQUIVOS – INTRODUÇÃO

- A memória principal de um computador é volátil e precisamos armazenar dados em dispositivos secundários
  - A estrutura de dados que guarda essas informações no armazenamento secundário é o **arquivo**

- Computadores utilizam de arquivos para armazenar grande volume de dados que se mantêm mesmo depois da execução dos programas

FLUXO DE DADOS (STREAM)

- Para o Java um arquivo é um **fluxo sequencial de caracteres ou bytes** finalizados por uma marca de final de arquivo ou pelo número total de bytes registrados.
- Temos dois tipos de fluxo de entrada e saída:
  - Os que se baseiam em bytes: arquivos binários
  - Os que se baseiam em caracteres: arquivos texto

Arquivos Texto

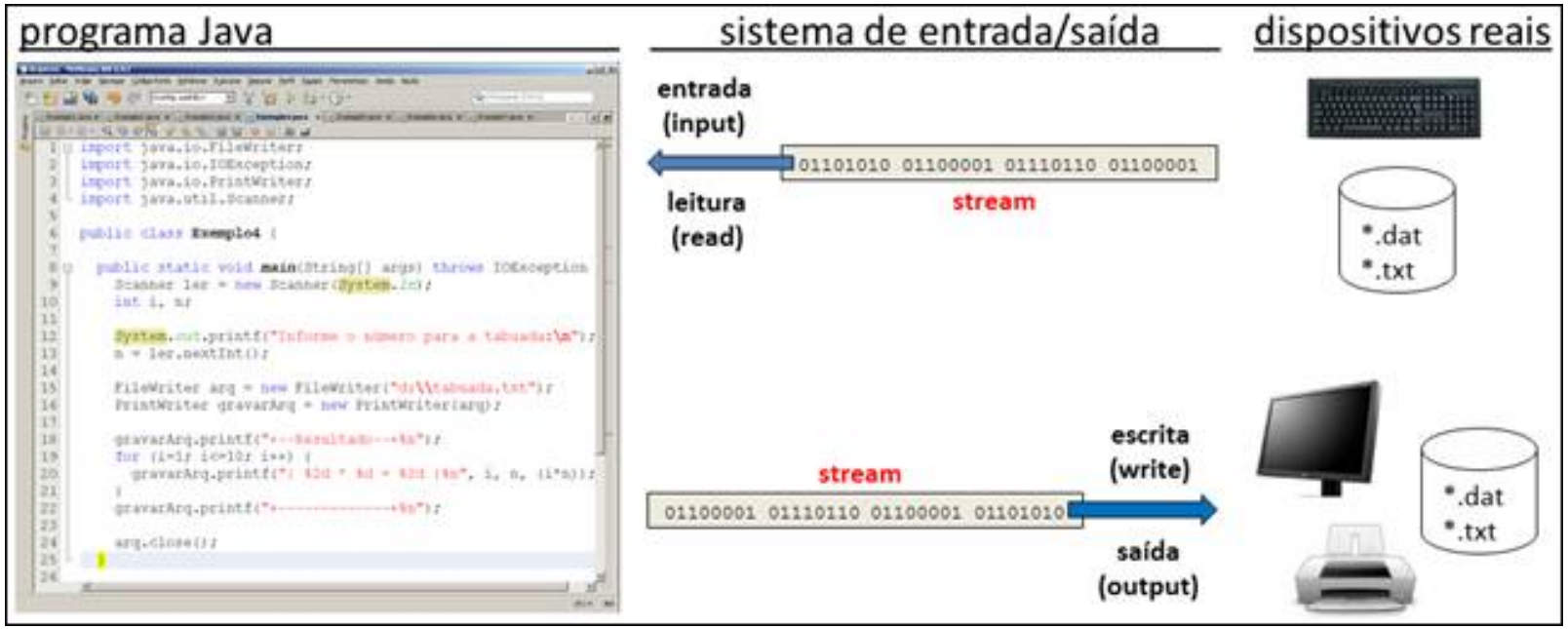
Os arquivos texto os dados são representados linha à linha. Na prática é uma sequência de bytes representando caracteres. As linhas são representadas pelo caracter de quebra de linha, por exemplo: '\n'.
Todos os dados são armazenados como caracteres. por exemplo o número 20 utilizaria um byte para cada dígito do número.

Arquivos Binários

Os arquivos binários são representados por uma sequência de bytes sem o conceito de quebra de linha. Ele armazena o dado literal, ou seja, não são caracteres. O número 20 ocuparia apenas 1 byte cujo valor em binário seria 00010100.

FLUXO DE DADOS (STREAM)

- Java fornece uma série de recursos para o programador utilizar, independente do dispositivo real que será acessado. Isso estabelece uma grande abstração entre o programa e dispositivo acessado
  - Teclado, disco, rede e outros



FLUXO DE DADOS (STREAM)

- Java cria três objetos de fluxo que são associados a dispositivos de entrada ou saída sempre que um programa inicia a execução:
  - **System.in**: objeto de fluxo de entrada padrão, normalmente utilizado pelo programa para obter dados a partir do teclado;
  - **System.out**: objeto de fluxo de saída padrão, normalmente utilizado pelo programa para enviar resultados para a tela do computador; e
  - **System.err**: objeto de fluxo de erro padrão, normalmente utilizado pelo programa para gerar saída de mensagens de erro na tela.

ARQUIVOS

- Programas Java implementam o processamento de arquivos utilizando as classes do pacote **java.io**. A hierarquia de classes oferecida por este pacote, oferece mais de 50 classes distintas para o processamento de entrada e saída em arquivos baseados em bytes e caracteres e arquivos de acesso aleatório. Os arquivos são abertos criando-se objetos através de uma das classes de fluxo. Algumas principais são:
  - **FileInputStream**: para entrada baseada em bytes de um arquivo;
  - **FileOutputStream**: para saída baseada em bytes para um arquivo;
  - **RandomAccessFile**: para entrada e saída baseada em bytes de e para um arquivo;
  - **FileReader**: para entrada baseada em caracteres de um arquivo;
  - **FileWriter**: para saída baseada em caracteres para um arquivo.

- Hierarquia das classes de acesso a fluxos:
  - Object
    - File
    - InputStream
      - FileInputStream
    - OutputStream
      - FileOutputStream
    - Reader
      - BufferedReader
      - InputStreamReader
    - Writer
      - OutputStreamWirter
      - PrintWriter

Classes para entrada ou saída baseada em bytes

Classes para entrada ou saída baseada em caracteres

ARQUIVOS

- A classe **File** é utilizada para recuperar informações sobre arquivos ou diretórios em disco. Os objetos da classe File não abrem arquivos de dados e também não fornecem capacidades de processamento de arquivos, apenas são utilizados para especificar arquivos ou diretórios.
- Ela possui vários métodos que permitem recuperar informações sobre o objeto instanciado.
  - boolean delete()
  - String getName()
  - boolean isFile()
  - boolean mkdir()
  - boolean exits()

ARQUIVOS

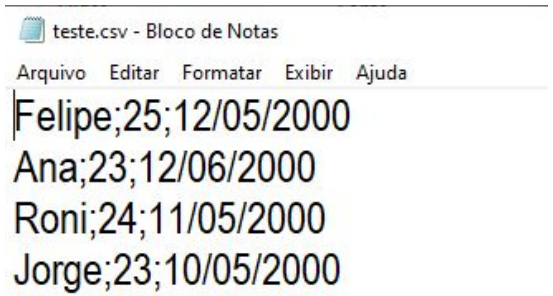
- **Leitura de Arquivos utilizando Scanner**
  - A classe Scanner é muito poderosa, podendo ser utilizada com outros canais de entrada de dados como texto, arquivo, etc.

```
public class LerArquivoScanner {

    public static void main(String[] args) throws IOException {
            File arquivo = new File("C:\\teste\\exercicios.txt");
             Scanner sc = new Scanner(arquivo);
             while (sc.hasNext()) {
                 System.out.print(sc.nextLine());
             }
             sc.close();
    }

}
```

ARQUIVOS

Utilizando delimitadores com Scanner

Criar um arquivo texto na pasta C:\Aula/teste.csv com o conteúdo abaixo




```java
public class ExampleDelimiter {

    public static void main(String[] args) throws FileNotFoundException {
        Scanner scanner = new Scanner(new File("/Aula/teste.csv"));
            scanner.useDelimiter(";");
            while(scanner.hasNext()){
                System.out.print(scanner.next()+" ");
            }
            scanner.close();


    }
}
```

ARQUIVOS

Exemplo de código que apresenta as informações de um arquivo ou diretório

```
public class ExemploFile {
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);

        System.out.printf("Informe o nome de um arquivo ou diretório:\n");
        String nome = ler.nextLine();

        File file = new File(nome);
        if (file.exists()) {
           if (file.isFile()) {
              System.out.printf("\nArquivo (%s) existe - tamanho: %d bytes\n",
                file.getName(),  file.length());
           }
           else {
             System.out.printf("\nConteúdo do diretório:\n");
             String diretorio[] = file.list();
             for (String item: diretorio) {
               System.out.printf("%s\n", item);
             }
           }
        } else {
            System.out.printf("Arquivo ou diretório informado não existe!\n");
        }
        ler.close();
    }
```

ARQUIVOS - PROCESSAMENTO

- Quando falamos em processamento de arquivo, passamos por três etapas:
  1. Abertura ou criação de um arquivo
  2. Leitura ou gravação de dados
  3. Fechamento o arquivo
- Caractere separador – utilizado para separar diretórios e arquivos em um caminho.
  - O Windows utiliza \
  - O UNIX utiliza /
  - O Java processa ambos os caracteres. **File.pathSeparator** pode ser utilizado para obter o caractere separador adequado do computador local

ARQUIVOS TEXTO

- Leitura de arquivo texto

Opção 1

```
public class Main {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new FileReader("arquivo.txt"));
            while(br.ready()) {
                System.out.println(br.readLine());
            }
            br.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

Opção 2

```
public class Main {
    public static void main(String[] args) {
        try {
            Scanner in = new Scanner(new InputStreamReader(new FileInputStream("arquivo.txt"), "UTF-8"));
            while(in.hasNext()) {
                System.out.println(in.nextLine());
            }
            in.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

ARQUIVOS TEXTO

- Escrevendo em arquivo texto

Opção 1

```
public class Main {
    public static void main(String[] args) {
        try {
            BufferedWriter out = new BufferedWriter(new FileWriter("arquivo.txt"));
            out.append("abc");
            out.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

Opção 2

```
public class Main {
    public static void main(String[] args) {
        try {
            OutputStreamWriter out = new OutputStreamWriter(new FileOutputStream("arquivo.txt"));
            out.append("abc");
            out.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

Se o arquivo não existir ele cria.
Caso exista, ele substitui.
Para adicionar dados no arquivo, devemos passar um segundo parâmetro **true** no construtor da classe **FileOutputStream**

EXERCÍCIO

- 2)Escreva um programa que leia um arquivo texto e retorne a quantidade de caracteres ele possui.

```
public class MainExercicio {
    public static void main(String[] args) {
        try {
            BufferedReader in = new BufferedReader(new FileReader("arquivo.txt"));
            int qntCaracteres = 0;
            while(in.ready()) {
                String linha = in.readLine();
                qntCaracteres += linha.length();
            }
            System.out.println(qntCaracteres);
            in.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

ARQUIVOS BINÁRIO

- Escrevendo em arquivo binário

```
public class MainBinario {
    public static void main(String[] args) {
        try {
            FileOutputStream arq = new FileOutputStream("arquivo.dat");
            DataOutputStream gravarArq = new DataOutputStream(arq);
            gravarArq.writeChars("Bom dia");
            gravarArq.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

Se o arquivo não existir ele cria. Caso exista, ele substitui.
Para adicionar dados no arquivo, devemos passar um segundo parâmetro **true** no construtor da classe **FileOutputStream**

ARQUIVOS BINÁRIO

- Lendo em arquivo binário

```
public class MainBinario {
    public static void main(String[] args) {
        try {
            FileInputStream arq = new FileInputStream("arquivo.dat");
            DataInputStream lerArq = new DataInputStream(arq);
            String resultado = lerArq.readUTF();
            System.out.println(resultado);
            lerArq.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

ARQUIVOS BINÁRIO

- Lendo em arquivo binário

```
public class MainBinario {
    public static void main(String[] args) {
        try {
            FileInputStream arq = new FileInputStream("arquivo.dat");
            DataInputStream lerArq = new DataInputStream(arq);
            String resultado = lerArq.readUTF();
            System.out.println(resultado);
            lerArq.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

EXERCÍCIO

- Faça um programa que leia no console o nome, idade e altura de uma pessoa e escreva em um arquivo binário.
- Depois leia este arquivo e apresente os dados no console.

```java
public class ExercicioBinario {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String nome = in.next();
        int idade = in.nextInt();
        double peso = in.nextDouble();
        in.close();
        try {
            FileOutputStream arqEscrever = new FileOutputStream("arquivo.dat");
            DataOutputStream gravarArq = new DataOutputStream(arqEscrever);
            gravarArq.writeUTF(nome);
            gravarArq.writeInt(idade);
            gravarArq.writeDouble(peso);
            gravarArq.close();


            FileInputStream arqLer = new FileInputStream("arquivo.dat");
            DataInputStream lerArq = new DataInputStream(arqLer);
            String nomeLeitura = lerArq.readUTF();
            int idadeLeitura = lerArq.readInt();
            double pesoLeitura = lerArq.readDouble();
            System.out.println(nomeLeitura);
            System.out.println(idadeLeitura);
            System.out.println(pesoLeitura);
            lerArq.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```

EXERCÍCIO

- Faça um programa que leia um texto onde as palavras estão separadas por “–” e as apresentem sem os traços.
  Exemplo:
  - O-rato-roeu-a-roupa-do-rei-de-Roma

```
public class ExercicioFrase {
    public static void main(String[] args) {
        try {
            BufferedReader in = new BufferedReader(new FileReader("arquivoFrase.txt"));
            String linha = in.readLine();
            String[] palavras = linha.split("-");
            for (String string : palavras) {
                System.out.print(string+" ");
            }
            System.out.println("\n");
            in.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar arquivo");
        }
    }
}
```