08

Mãos na massa: Encoding e Charsets

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

Encoding e Charset

1) No projeto **java-io**, crie a classe `TesteUnicodeEEncoding`, no pacote **br.com.alura.java.io.teste**, com o método `main`:

```
public class TesteUnicodeEEncoding {

    public static void main(String[] args) {

    }

}
```

2) Crie uma string com o caractere `C` e imprima o seu *code point*:

```
public class TesteUnicodeEEncoding {

    public static void main(String[] args) {

        String s = "C";
        System.out.println(s.codePointAt(0));
    }

}
```

3) Guarde o *charset* padrão do sistema operacional em uma variável e imprima o seu nome:

```
public class TesteUnicodeEEncoding {

    public static void main(String[] args) {

        String s = "C";
        System.out.println(s.codePointAt(0));

        Charset charset = Charset.defaultCharset();
        System.out.println(charset.displayName());

    }

}
```

Se você estiver utilizando o Windows, o *charset* deve ser **windows-1252** e se você estiver utilizando o macOS/Linux, o *charset* deve ser **UTF-8**.

4) Guarde os bytes em um array, e *force* o *charset* para **windows-1252** (você precisará adicionar um `throws`). Em seguida, imprima o tamanho do array:

```
public class TesteUnicodeEEncoding {

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

        String s = "C";
        System.out.println(s.codePointAt(0));

        Charset charset = Charset.defaultCharset();
        System.out.println(charset.displayName());

        byte[] bytes = s.getBytes("windows-1252");
        System.out.println(bytes.length + ", windows-1252");

    }

}
```

5) Faça o mesmo para o *charset* **UTF-16** e para o **US_ASCII** (para esse, utilize a classe `StandardCharsets`):

```
public class TesteUnicodeEEncoding {

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

        String s = "C";
        System.out.println(s.codePointAt(0));

        Charset charset = Charset.defaultCharset();
        System.out.println(charset.displayName());

        byte[] bytes = s.getBytes("windows-1252");
        System.out.println(bytes.length + ", windows-1252");

        bytes = s.getBytes("UTF-16");

        System.out.println(bytes.length + ", UTF-16");

        bytes = s.getBytes(StandardCharsets.US_ASCII);
        System.out.println(bytes.length + ", US_ASCII");

    }

}
```

6) Agora, para cada *charset*, transforme os bytes em string, especificando o *charset* **windows-1252**, e imprima a nova string. Melhore a impressão, deixando-a somente em uma linha:

```
public class TesteUnicodeEEncoding {

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

        String s = "C";
```

```
        System.out.println(s.codePointAt(0));

        Charset charset = Charset.defaultCharset();
        System.out.println(charset.displayName());

        byte[] bytes = s.getBytes("windows-1252");
        System.out.print(bytes.length + ", windows-1252, ");
        String sNovo = new String(bytes, "windows-1252");
        System.out.println(sNovo);


        bytes = s.getBytes("UTF-16");
        System.out.print(bytes.length + ", UTF-16, ");
        sNovo = new String(bytes, "windows-1252");
        System.out.println(sNovo);

        bytes = s.getBytes(StandardCharsets.US_ASCII);
        System.out.print(bytes.length + ", US_ASCII, ");
        sNovo = new String(bytes, "windows-1252");
        System.out.println(sNovo);

    }

}
```

Repare como dá problema com o **UTF-16**.

7) Modifique a string para o caractere **ç** e altere a transformação dos bytes em string para o seu respectivo *charset*:

```
public class TesteUnicodeEEncoding {


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

        String s = "ç";
        System.out.println(s.codePointAt(0));

        Charset charset = Charset.defaultCharset();
        System.out.println(charset.displayName());

        byte[] bytes = s.getBytes("windows-1252");
        System.out.print(bytes.length + ", windows-1252, ");
        String sNovo = new String(bytes, "windows-1252");
        System.out.println(sNovo);

        bytes = s.getBytes("UTF-16");
        System.out.print(bytes.length + ", UTF-16, ");
        sNovo = new String(bytes, "UTF-16");
        System.out.println(sNovo);

        bytes = s.getBytes(StandardCharsets.US_ASCII);
        System.out.print(bytes.length + ", US_ASCII, ");
        sNovo = new String(bytes, "US-ASCII");
        System.out.println(sNovo);

    }
```

```
}
```

Desta vez o problema ocorre com o **US-ASCII**.

Encoding com java.io

8) Baixe aqui (https://caelum-online-public.s3.amazonaws.com/857-java-io/05/contas.csv) o novo arquivo **contas.csv** e copie-o para a **raiz** do projeto **java-io**, substituindo o antigo.

9) Se o *encoding* do **contas.csv** não estiver correto, você pode modificá-lo, clicando com o botão da direita do mouse sobre o arquivo e acessando ***Properties***. Em ***Text file encoding***, modifique o *encoding*. Faça o mesmo com os arquivos **lorem.txt** e **lorem2.txt**.

10) Abra a classe `TesteLeitura2`. Na instanciação do `Scanner`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
Scanner scanner = new Scanner(new File("contas.csv"), "UTF-8");
```

11) Abra a classe `TesteLeitura`. Na instanciação do `InputStreamReader`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
Reader isr = new InputStreamReader(fis, "UTF-8");
```

12) Abra a classe `TesteEscritaPrintStreamPrintWriter`. Na instanciação do `PrintWriter`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
PrintWriter pw = new PrintWriter("lorem2.txt", "UTF-8");
```

13) Você também pode modificar o *encoding* do projeto modificá-lo, clicando com o botão da direita do mouse sobre o mesmo e acessando ***Properties***. Em ***Text file encoding***, modifique o *encoding*.