



05. Encoding e Charsets

📅 Date	@17/10/2022
📁 Categoria	Java
📖 Curso	Java e java.io: Streams Reader e Writers

Revisão

Primeira data: @February 21, 2022

Tópicos

- Character Sets
- Certo ou Errado?
- Para saber mais: FAQ do UNICODE
- Encoding no Windows
- Encoding no Java
- Encoding com java.io
- UTF-8 no arquivo
- Mãos na massa: Encoding e Charsets
- O que aprendemos

O que aprendemos?

Nessa aula você aprendeu sobre Unicode, Encodings e Charsets.

Conheceu o problema dos **Encodings**: onde diferentes codepages são usados para escrever e exibir informações em seu computador. A solução foi dada por um consórcio que criou uma tabela genérica chamada **Unicode** contendo todos os caracteres do mundo em números denominados **codepoints**. A segunda parte da solução é aplicar diferentes Encodings para definir como os bytes são gravados nos arquivos. Os encodings são tabelas que transformam cada codepoint em seu caractere específico, dependendo de determinada região. Também observou que os encodings utilizados dependem muito de cada sistema operacional.

Usando o Windows, você implementou um programa para verificar a implementação do Java para Unicodes e Encodings e conheceu várias classes e métodos. Aprendeu que a classe String possui um método chamado `codePointAt()` para revelar o codepoint de determinado caractere a partir de sua posição na string. Descobriu que a classe que representa um encoding ou *Character Set* é `Charset` e o método estático para retornar uma referência com o charset default é `defaultCharset()`. Aprendeu que a classe String também possui um método para transformar os caracteres em bytes, o `getBytes()`, que usado sem argumento de entrada utiliza o charset padrão. Existem também duas sobrecargas para esse método, onde você pode informar o charset que deseja utilizar para a transformação. Conheceu a classe `StandardCharsets`, do pacote `java.nio`, que possui constantes para os principais charsets. Por fim simulou o problema de encodings, gerando uma nova string a partir de um construtor que tinha como argumentos os bytes transformados no charset e o charset desejado para transformação. A solução foi garantir que o mesmo charset fosse aplicado, tanto na entrada quanto na saída.

As classes `Scanner` e `InputStreamReader` possuem sobrecargas de construtores que recebem como argumento qual charset será utilizado para fazer a transformação dos bytes em strings. De modo análogo para escrita, a classe `PrintWriter` também permite informar qual charset será utilizado para transformar a string nos bytes específicos.
