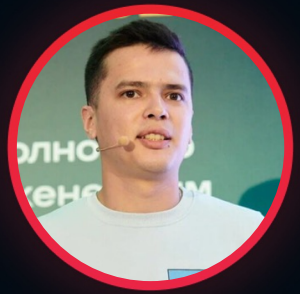
JEP-400 или UTF-8 РєРѕРґРёСЂРѕРІРєР° РїРѕ СѓРјРѕР»С‡Р°РЅРёСЋ



Дмитрий Сугробов
@voborgus

Joker<?>

Дмитрий Сугробов

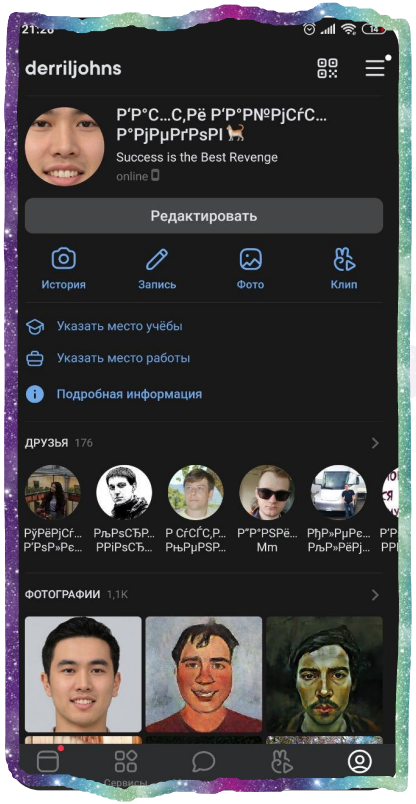
— Некогда фултаймовый джавист
— Член InnerSource Common Foundation
— Проповедник инженерных практик

InnerSource Commons

innersourcecommons.org

Мир переходит на Unicode

Мир переходит на Unicode




```
<?php
echo strlen("Привет Joker");
```

18

Источник картинки: https://t.me/theaftertimes

Мир переходит на Unicode

Какие проблемы?

1. Записываем файл с **Charset.defaultCharset()**
2. Перекидываем на другой хост
3. Считываем файл с **Charset.defaultCharset()**

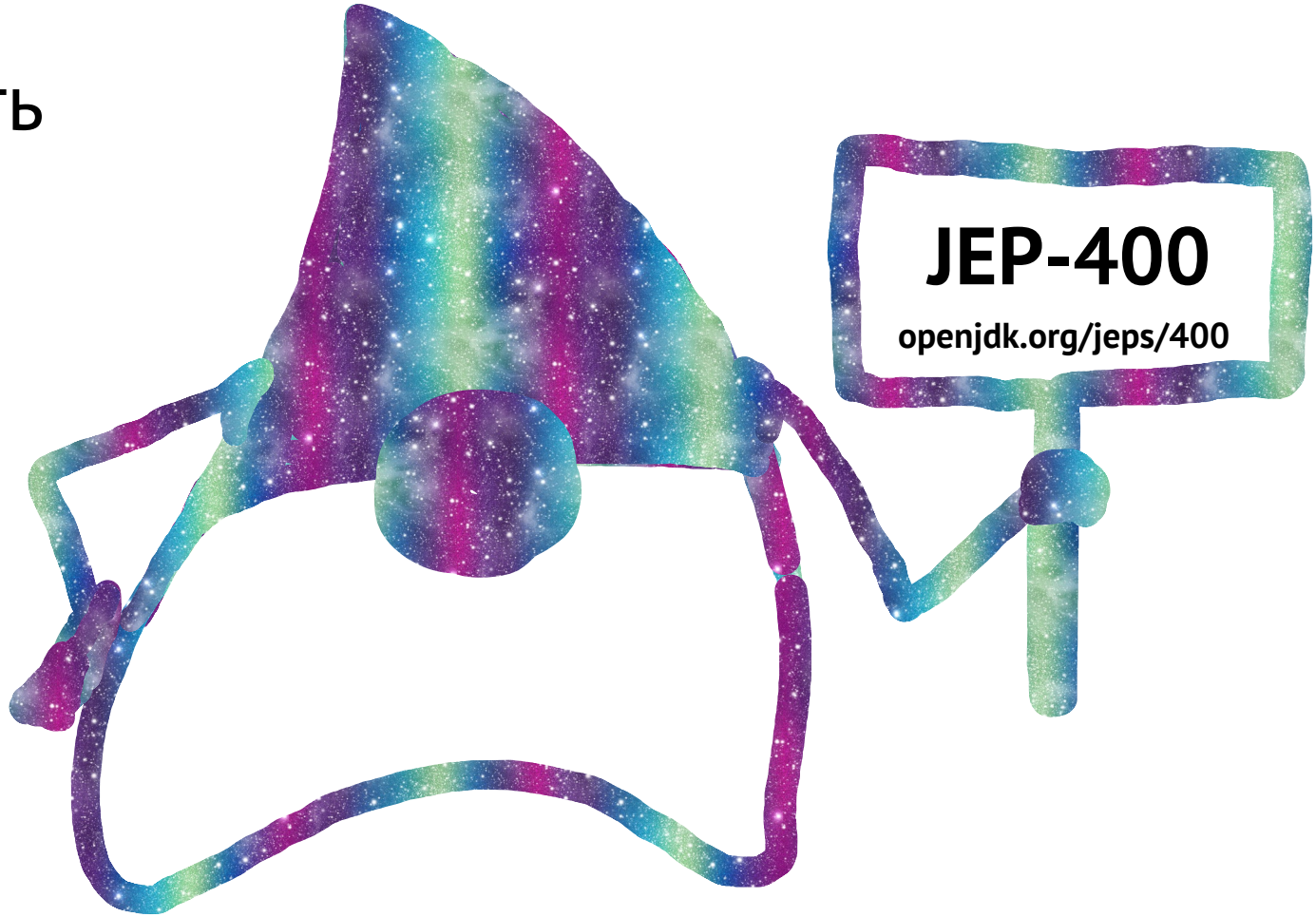
«write once, run anywhere»

Какие проблемы?

1. Записываем файл с **Charset.defaultCharset()**
2. Меняем системную кодировку
3. Считываем файл с **Charset.defaultCharset()**

1. Записываем **Files.write(path, str.getBytes())**
2. Считываем **Files.readString(path)**

Выход есть



Жизнь до JEP-400

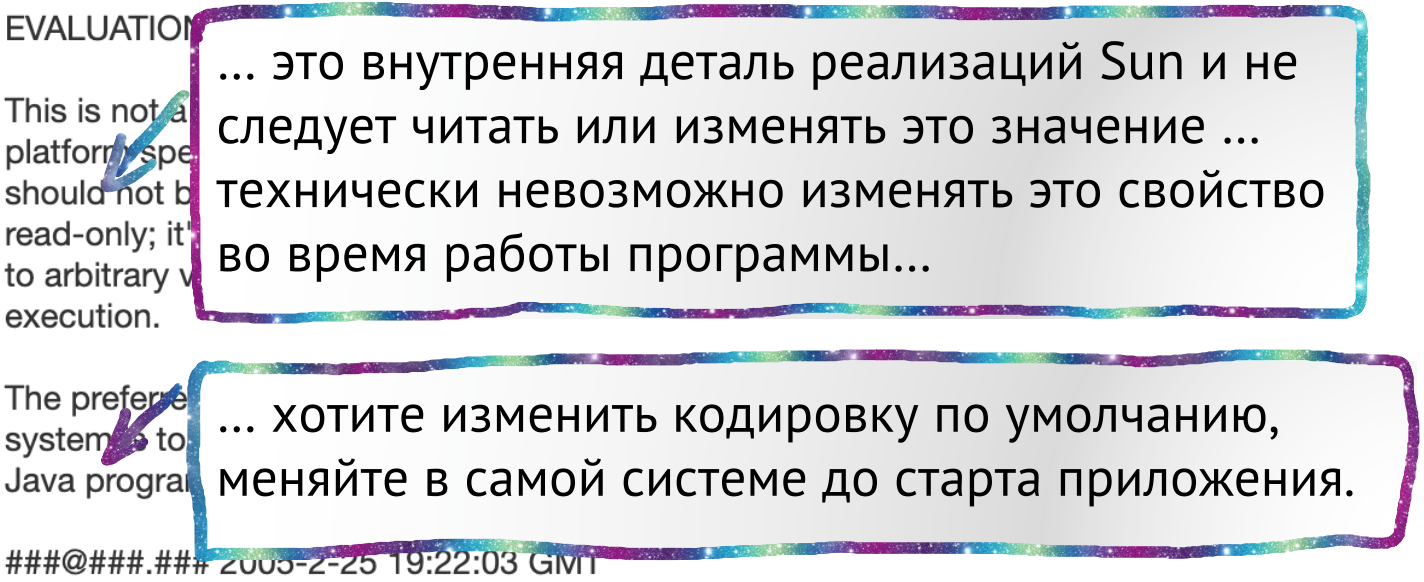
- Не все указывают кодировку
- Указание ломает конструкции (::) в стримах
- Проперти файлы считываются в UTF-8*
- Методы **java.nio.file.Files** по-умолчанию UTF-8
- **URLEncoder/Decoder** задепрекейтили дефолт
- По умолчанию смотрят на **Charset.defaultCharset()**:
  - **-Dfile.encoding**
  - Системная локаль

InputStreamReader/Writer
FileReader/Writer
PrintStream/Writer
Formatter
Scanner
String.getBytes()

* Начиная с JDK-9, раньше был Latin-1

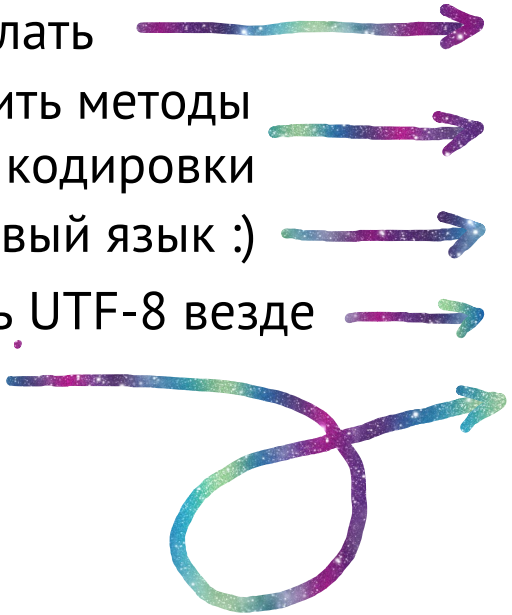
Недокументированный -Dfile.encoding

Comments

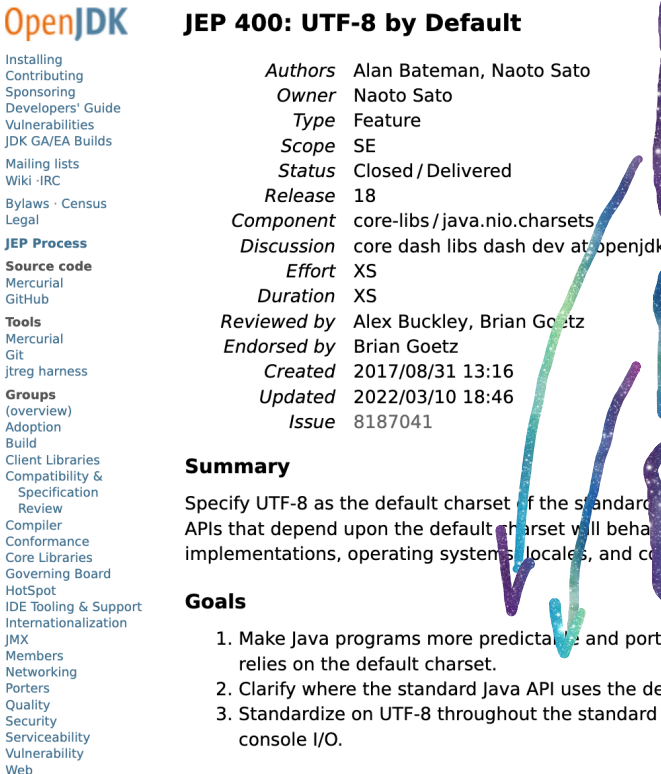


https://bugs.java.com/bugdatabase/view_bug.do?bug_id=4163515

Что поделать	Последствия
— Ничего не делать	— Не решит проблему
— Задепрекейтить методы без указания кодировки	— Удлинит код, риск получить непараметризуемый зоопарк
— Выпустить новый язык :)	— Не выход
— Захардкодить UTF-8 везде	— Ломаем обратную совместимость
— Путь JEP-400	— Так и победим



В далёком 2017-ом...

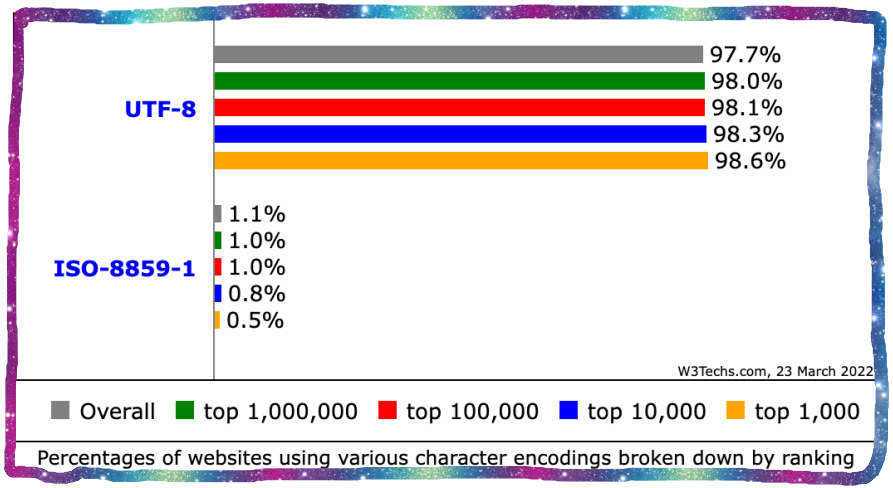


1. Сделать поведение программ более предсказуемым и портируемым в методах, где используется кодировка по умолчанию
2. Понять где Java API использует кодировку по умолчанию
3. Стандартизировать UTF-8 во всех Java API, кроме консоли

https://openjdk.java.net/jeps/400

Почему собсна UTF-8?

- Все братюни уже перешли
- XML и JSON давно уже там
- Стандарт де-факто© в вебе
- Уже захардкожено по умолчанию в **java.nio.file.Files**
- Проперти в UTF-8 – JEP-226



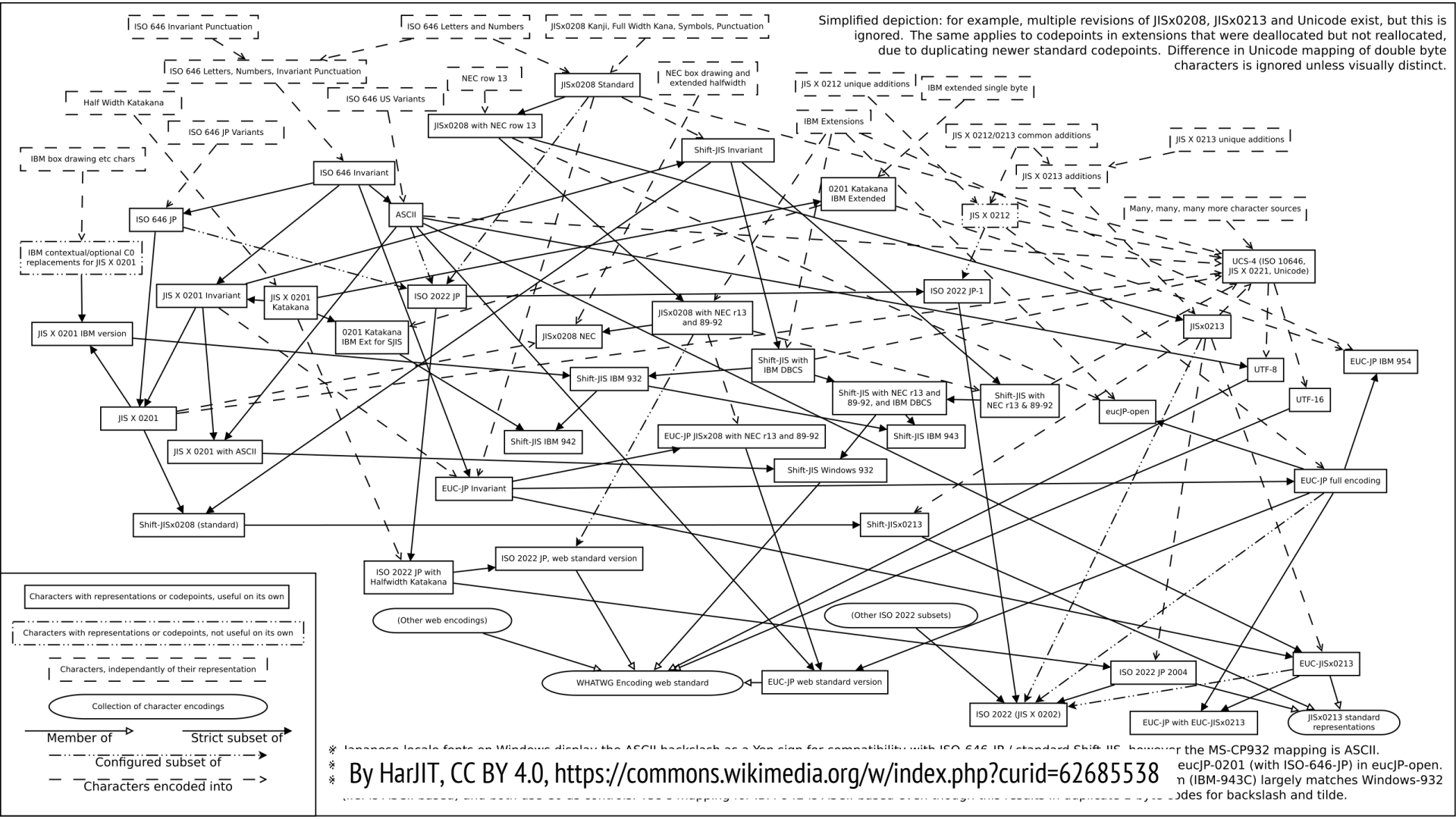
https://w3techs.com/technologies/cross/character_encoding/ranking

Экскурс в кодировки

– ISO-8859-1 (Latin-1)
– Windows-1252

– Windows-1251
– CP866
– KOI8-R
– …

Charset.availableCharsets().size()  173



Экскурс в кодировки

UNICODE

https://home.unicode.org/

Отличие между UTF-8/16/32

	J	Д	
	U+004A	U+0414	U+1F9A5
— UTF-8	0x4A	0xD0 0x94	0xF0 0x9F 0xA6 0xA5
— UTF-16	0x004A	0x0414	0xD83E 0xDDA5
— UTF-32	0x0000004A	0x00000414	0x0001F9A5

Сколько байт весит кириллица?

Диапазон номеров символов	Требуемое количество октетов
00000000–0000007F	1
00000080–000007FF	2
00000800–0000FFFF	3
00010000–0010FFFF	4

Википедия

название		диапазон кодов (hex)	версия Юникода
Cyrillic	Кириллица	0400—04FF	1.1
Cyrillic Supplement	Дополнение к кириллице	0500—052F	3.2
Cyrillic Extended-A	Расширенная кириллица — A	2DE0—2DFF	5.1
Cyrillic Extended-B	Расширенная кириллица — B	A640—A69F	5.1
Cyrillic Extended-C	Расширенная кириллица — C	1C80—1C8F	9.0

Википедия

Как в Java хранятся строки?

```java
public class RussianText {
    public static void main(String args[]) {
        String text = "Привет Joker";
        byte[] bytes = text.getBytes(StandardCharsets.UTF_8);

        System.out.println("Length: " + text.length());
        System.out.println("Bytes: " + bytes.length);
        System.out.println("Content: " + Arrays.toString(bytes));
    }
}

>> Length: 12
>> Bytes: 18
>> Content: [-48, -97, -47, -128, -48, -72, -48, -78, -48, -75, -47, -126, 32, 74, 111, 107, 101,
114]
```

Как в Java хранятся строки?

```
public class RussianText {
    public static void main(String args[]) {
        String text = "🐛";
        byte[] bytes = text.getBytes(StandardCharsets.UTF_8);

        System.out.println("Length: " + text.length());
        System.out.println("Bytes: " + bytes.length);
        System.out.println("Content: " + Arrays.toString(bytes));
    }
}

>> Length: 2
>> Bytes: 4
>> Content: [-16, -97, -112, -101]
```

U+1F41B

UTF-16: **0xD83D 0xDC1B**

— Почему UTF-8, а не UTF-16?

— **String** хранится в 1-байтовом Latin-1 или 2/4 байтовом UTF-16*
  UTF-16 оптимальнее UTF-8 на больших текстах в иероглифах
  В остальных случаях проигрывает

— А чего не UTF-32?

— Можно индексировать, быстро искать и заменять
  Очень не оптимально по памяти, крайние байты редко используются

...

— Ладно, давай UTF-8.

* Compact Strings завезли в JDK-9

JDK-17: Подготовка к JEP-400

JDK / JDK-8265989

System property for the native character encoding name

Details

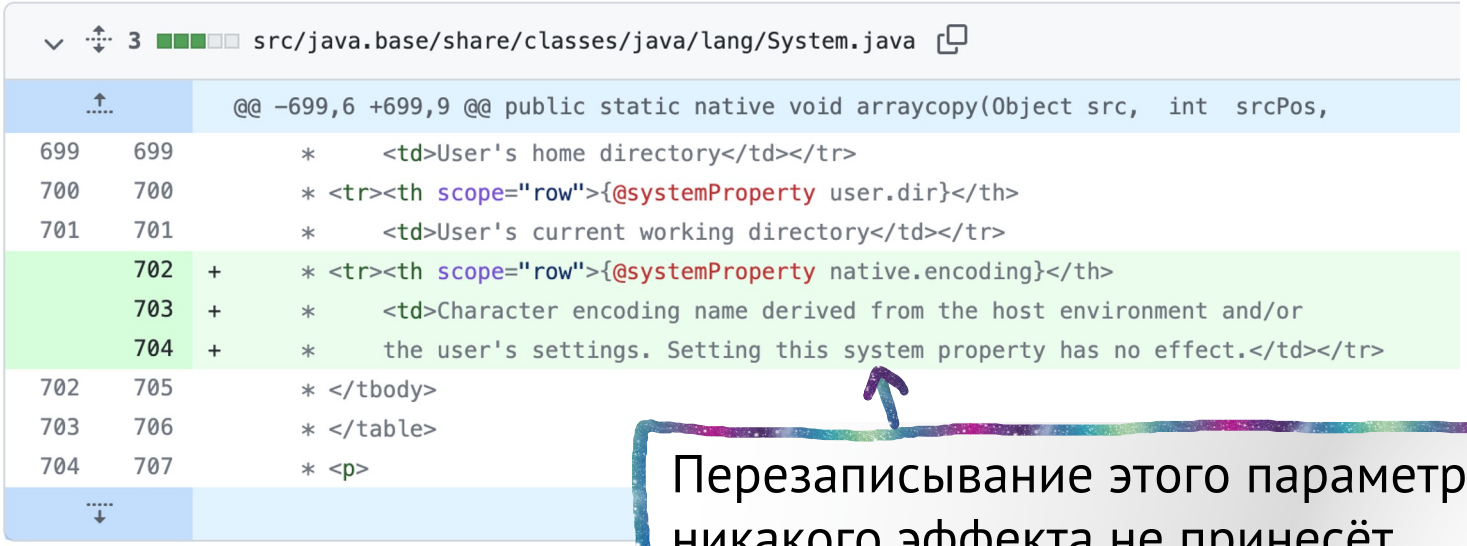
Type:	Enhancement	Status:	RESOLVED
Priority:	P4	Resolution:	Fixed
Affects Version/s:	17	Fix Version/s:	17
Component/s:	core-libs		
Labels:	release-note=yes		
Subcomponent:	java.lang		
Resolved In Build:	b21		
CPU:	generic		
OS:	generic		

Ввести новый параметр **-Dnative.encoding**, принимающий значение кодировки в системе.

Description

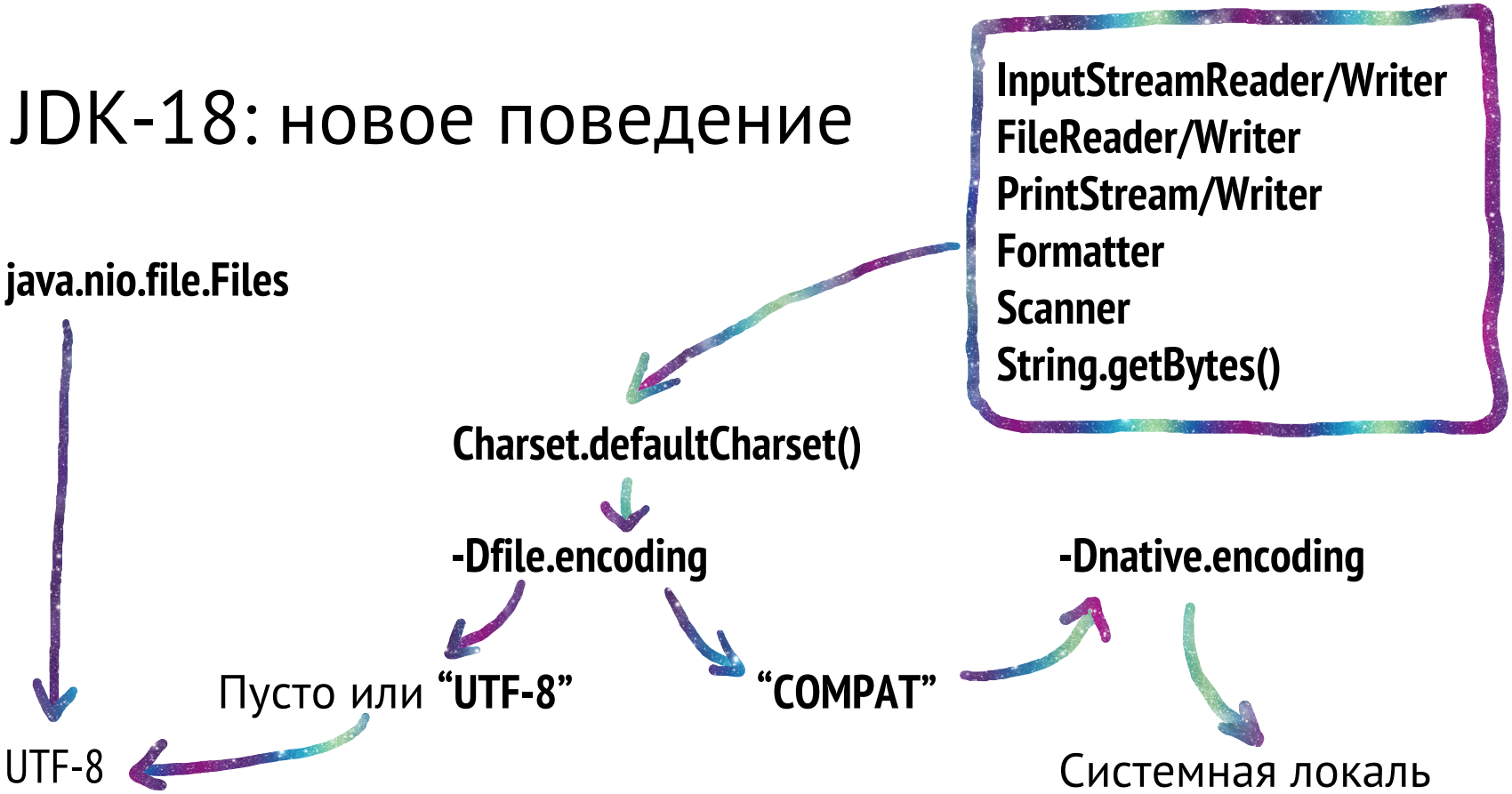
In preparation for the JEP 400, a new system property that represents the encoding of the host environment and the user's setting is desired.

JDK-17: Подготовка к JEP-400



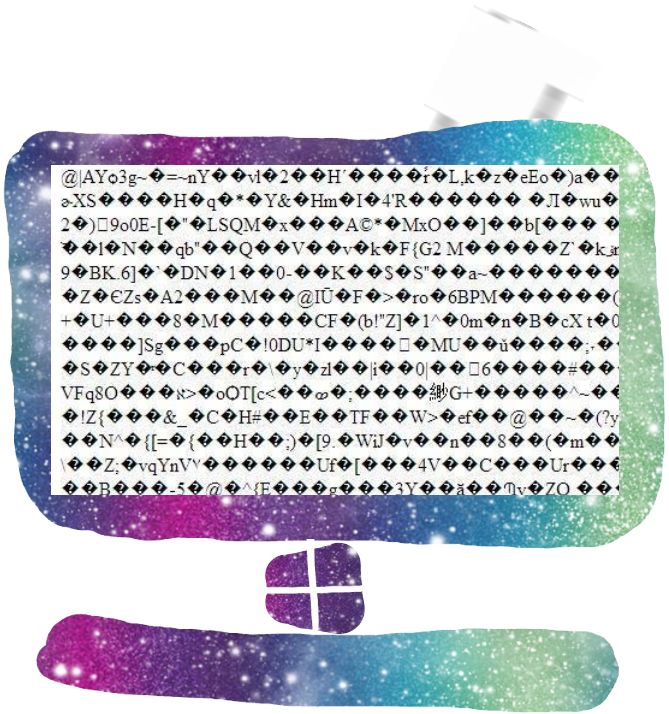
https://github.com/openjdk/jdk/commit/4e96b310425de541675b28493fdbe195780623c3

JDK-18: новое поведение

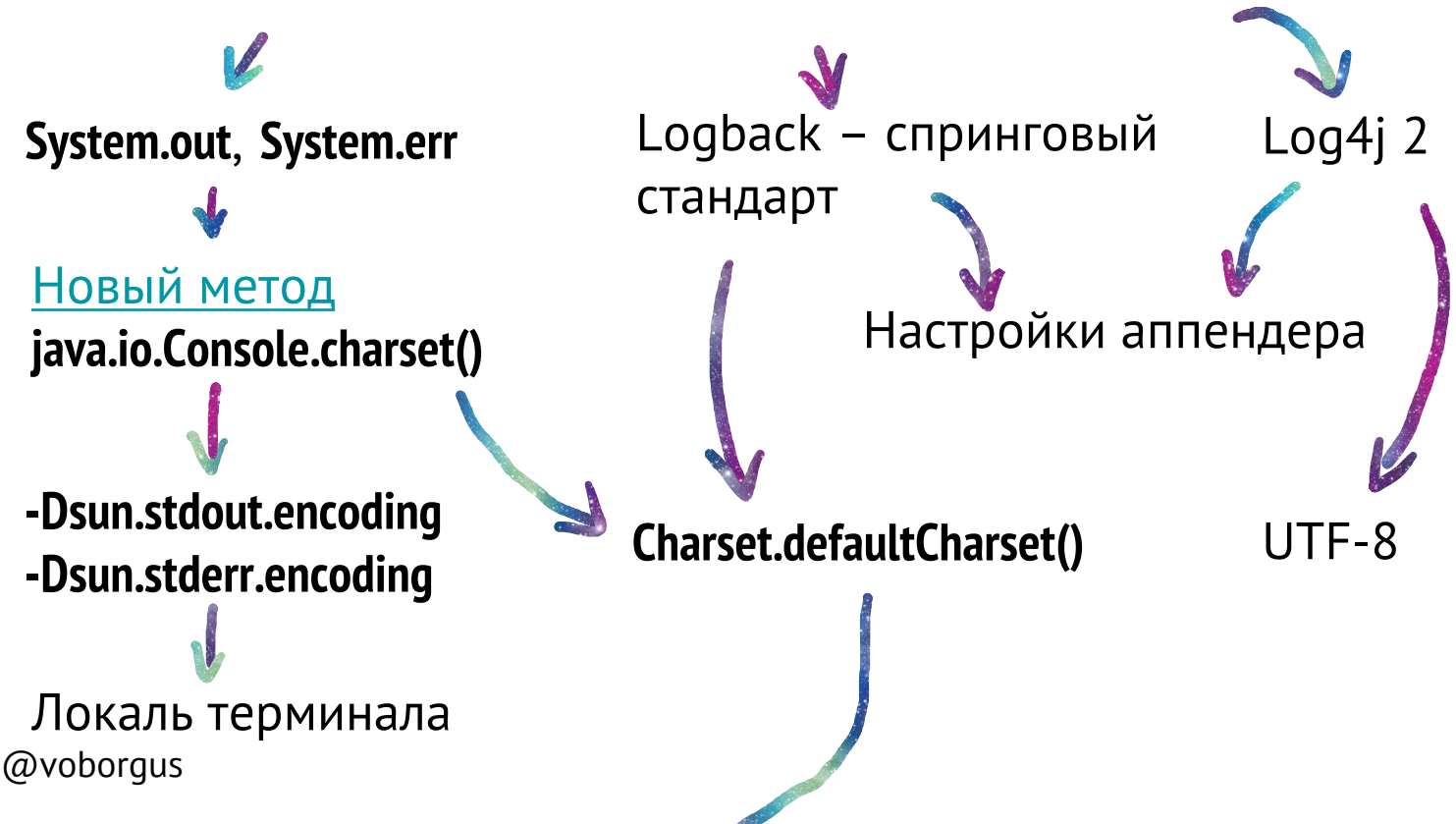


JDK-18: новое поведение

1. Записываем **Files.write(path, str.getBytes())**
2. Считываем **Files.readString(path)**
3. Выводим **System.out.println(string)**



JDK-18: а что с консолью – вывод



JDK-18: а что с консолью – ввод

* Если он есть, а если нет – UTF-8

** https://bugs.openjdk.org/browse/JDK-8124977

JDK-18: что осталось прежним

Зависит от системы

↓

-Dsun.jnu.encoding

Путь в файловой системе

— 1-байтовое Latin-1
— 2/4-байтовое UTF-16

Внутреннее представление **String**

Зависит от системы

↓

-encoding в **javac**

Кодировка исходного кода

UTF-8

Проперти файлы

Какие есть риски

Коррупция данных

Некорректное отображение

Некорректное поведение

Как их митигировать

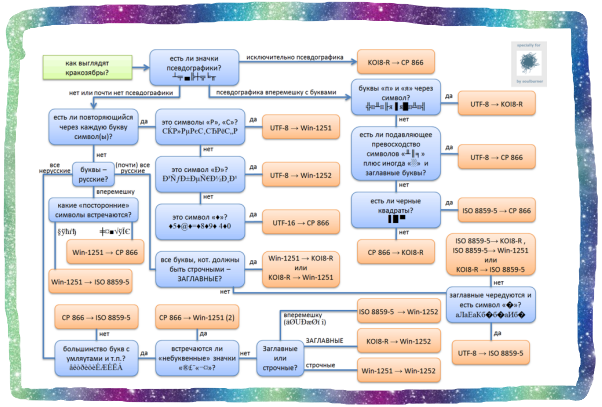
- Всегда указывать кодировку
- Работать только с UTF-8
- Если хочется нативного чтения и исходная кодировка не известна:

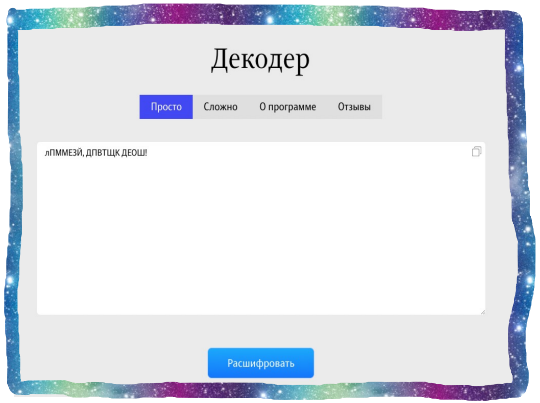
```
String encoding = System.getProperty("native.encoding");

Charset cs = (encoding != null) ? Charset.forName(encoding) : Charset.defaultCharset();

var reader = new FileReader("file.txt", cs);
```

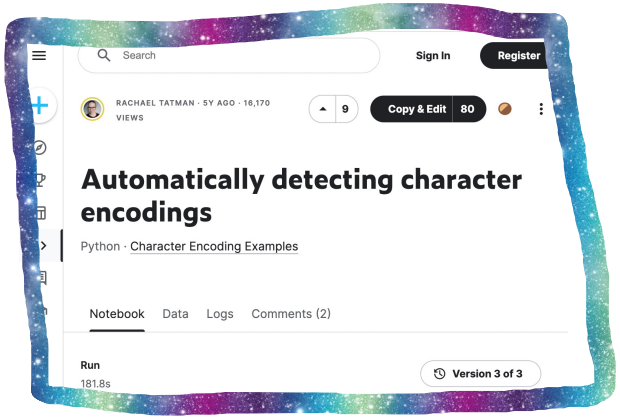
- Работать с **-Dfile.encoding=COMPAT**

Меня закрякозябрило, как найти оригинал?

– Никак.







habr.com/ru/post/147843

artlebedev.ru/decoder
online-decoder.com
2cyr.com/decode

github.com/chardet/chardet
github.com/brianmario/charlock_holmes
github.com/polygonplanet/encoding.js

Что хорошего получили

UTF-8

по-умолчанию

-Dfile.encoding

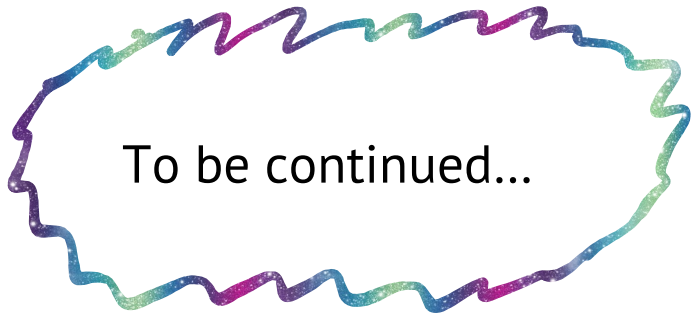
стандартизация

-Dnative.encoding

появление

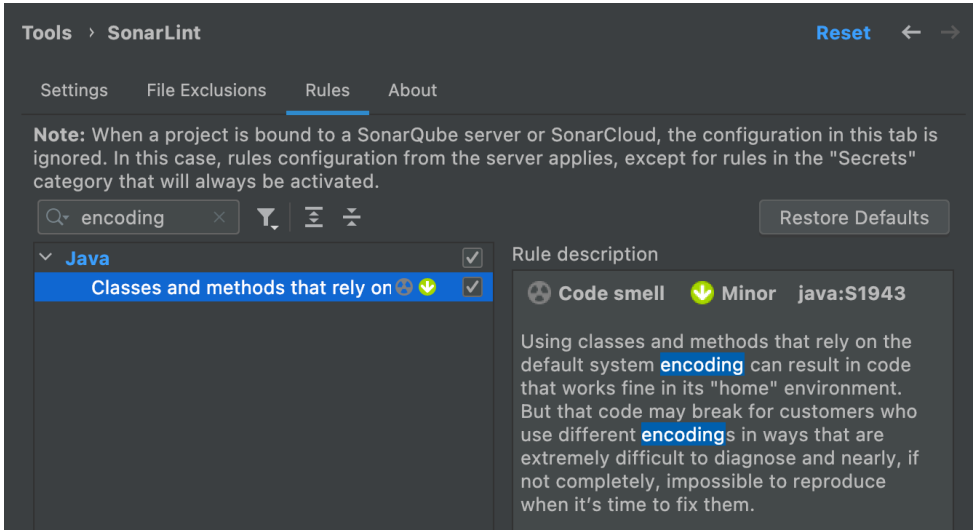
Какие, как говорится, зоны развития

- Поломали немного обратной совместимости
- Все еще разница в подходах **java.nio.file.Files** и методах с default
- Есть пространство для стандартизации кодировок
- Более удобный API для получения нативной кодировки



Что делать:

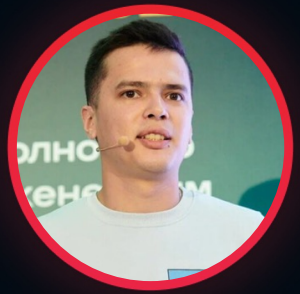
- Запустить с **files.encoding** и прогнать автотесты
- Поставить галочку в Sonarlint
- RSPEC-1943 в SonarQube
- Rewrite Recipe
- Forbidden APIs
- Error Prone



Двумя словами

— Начиная с JDK 18 по умолчанию UTF-8 кодировка
— Есть нюансы с консолью и путями в файловой системе
— Разобрались в Unicode
— Знаем, как определяется кодировка в JDK
— Разработка языка полна компромиссов
— Следить за разработкой интересно и легко

JEP-400 или UTF-8 кодировка по-умолчанию



Дмитрий Сугробов
@voborgus

Joker<?>