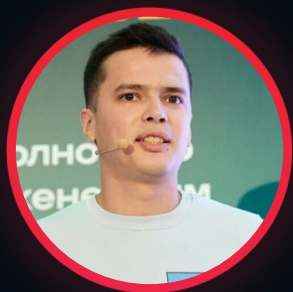


JEP-400

или UTF-8 РєРѕРјРёСЃРѕРІРєР°
РіРѕ СѓРјРѕ»СѓР°РѕРѕСЃ



Дмитрий
Сугробов
@voborgus

Joker<?>

Дмитрий Сугробов



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



innersourcecommons.org

Мир переходит на Unicode



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»

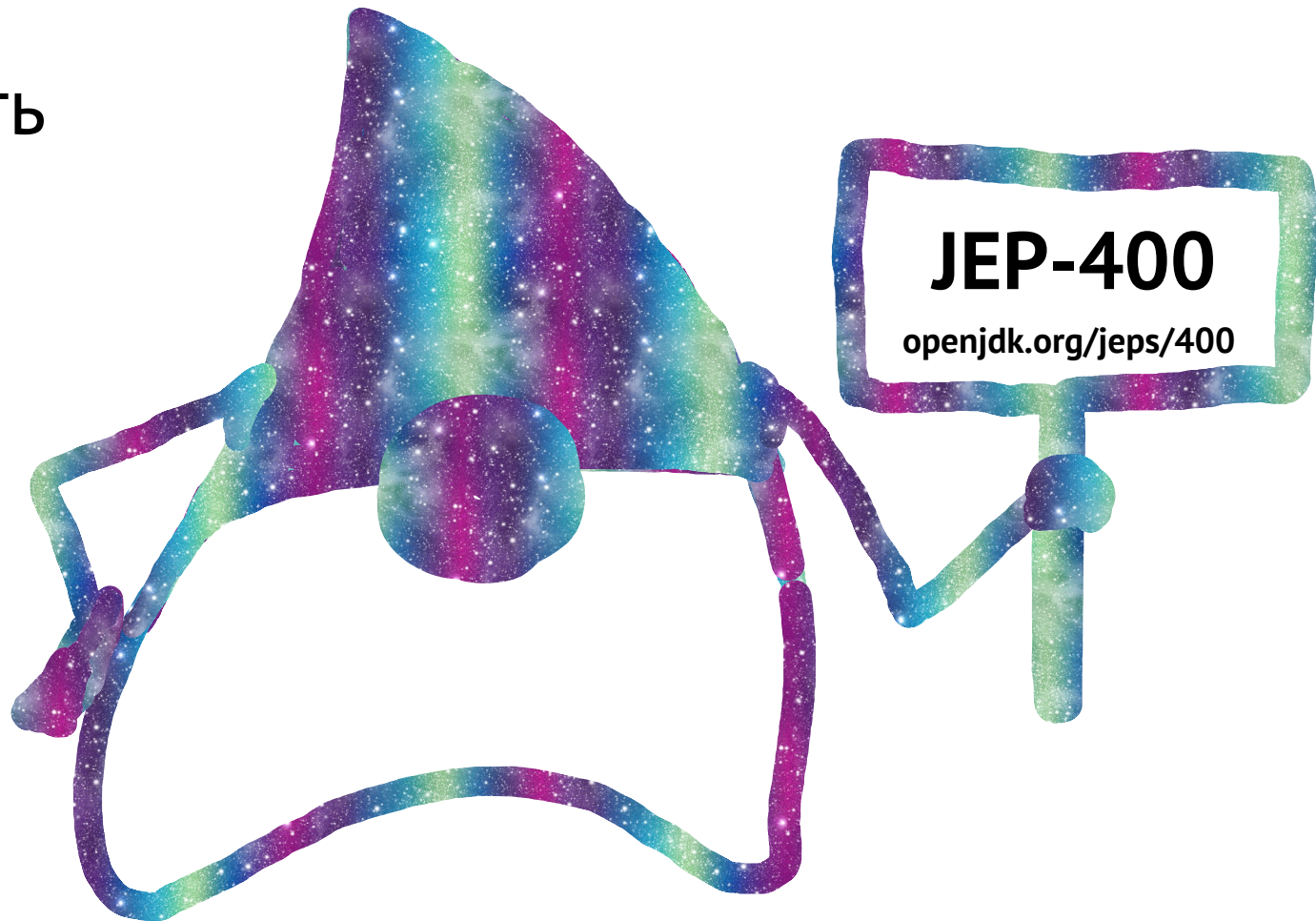
Какие проблемы?

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

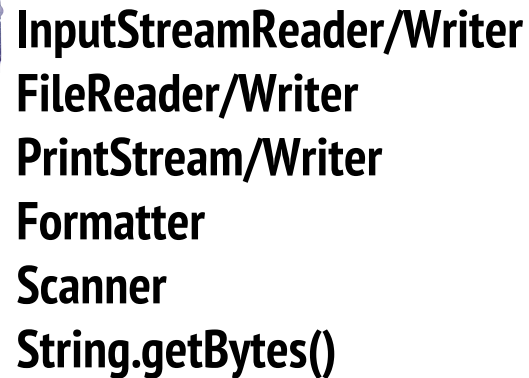


Жизнь до JEP-400

- 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()

Недокументированный -Dfile.encoding

Comments

EVALUATION

This is not a platform specific property; it should not be read-only; it is subject to arbitrary change at execution.






... это внутренняя деталь реализаций Sun и не следует читать или изменять это значение ...
технически невозможно изменять это свойство во время работы программы...

The preferred system property to use for Java programs is -Dfile.encoding.

... хотите изменить кодировку по умолчанию, меняйте в самой системе до старта приложения.

###@###.### 2005-2-25 19:22:03 GMT

Что поделатать

- Ничего не делать 
- Задепрекейтить методы без указания кодировки 
- Выпустить новый язык :) 
- Захардкодить UTF-8 везде 
- Путь JEP-400 

Последствия

- Не решит проблему
- Удлинит код, риск получить непараметризуемый зоопарк
- Не выход
- Ломаем обратную совместимость
- Так и победим

В далёком 2017-ом...

OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds
Mailing Lists
Wiki -IRC

Bylaws - Census
Legal

JEP Process

Source code
Mercurial
GitHub

Tools
Mercurial
Git
jreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility &
Specification
Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization

JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

JEP 400: UTF-8 by Default

<i>Authors</i>	Alan Bateman, Naoto Sato
<i>Owner</i>	Naoto Sato
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	18
<i>Component</i>	core-libs / java.nio.charsets
<i>Discussion</i>	core dash libs dash dev at openjdk dot java dot net
<i>Effort</i>	XS
<i>Duration</i>	XS
<i>Reviewed by</i>	Alex Buckley, Brian Goetz
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2017/08/31 13:16
<i>Updated</i>	2022/03/10 18:46
<i>Issue</i>	8187041

Summary

Specify UTF-8 as the default charset of the standard Java APIs that depend upon the default charset will behave consistently across implementations, operating systems, locales, and code pages.

Goals

1. Make Java programs more predictable and portable when their code relies on the default charset.
2. Clarify where the standard Java API uses the default charset.
3. Standardize on UTF-8 throughout the standard Java APIs, except for console I/O.

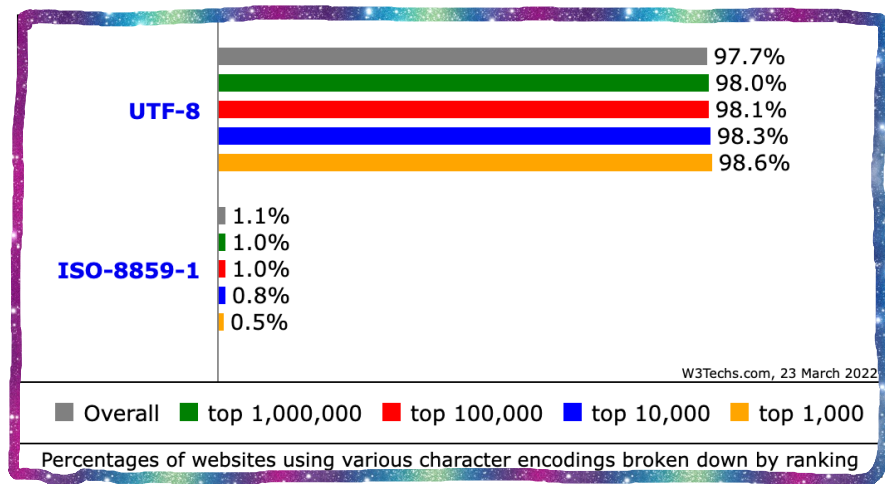
1. Сделать поведение программ более предсказуемым и портируемым в методах, где используется кодировка по умолчанию

2. Понять где Java API использует кодировку по умолчанию

3. Стандартизировать UTF-8 во всех Java API, кроме консоли

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](#)



Экскурс в кодировки

- ISO-8859-1 (Latin-1)
- Windows-1252

- Windows-1251
- CP866
- KOI8-R
- ...

Charset.availableCharsets().size()  173


Экскурс в кодировки



КАК МНОЖАТСЯ СТАНДАРТЫ:
(С.М.: ЗАРЯДНЫЕ УСТРОЙСТВА, КОДИРОВКИ, МГНОВЕННЫЕ СООБЩЕНИЯ И Т.Д.)



Отличие между UTF-8/16/32

	Ј	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 — А	2DE0–2DFF	5.1
Cyrillic Extended-B	Расширенная кириллица — В	A640–A69F	5.1
Cyrillic Extended-C	Расширенная кириллица — С	1C80–1C8F	9.0

[Википедия](#)

Как в Java хранятся строки?

```
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.

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**,
принимаящий 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

```
src/java.base/share/classes/java/lang/System.java
@@ -699,6 +699,9 @@ public static native void arraycopy(Object src, int srcPos,
699 699      * <td>User's home directory</td></tr>
700 700      * <tr><th scope="row">{@systemProperty user.dir}</th>
701 701      * <td>User's current working directory</td></tr>
702 +      * <tr><th scope="row">{@systemProperty native.encoding}</th>
703 +      * <td>Character encoding name derived from the host environment and/or
704 +      * the user's settings. Setting this system property has no effect.</td></tr>
702 705      * </tbody>
703 706      * </table>
704 707      * <p>
```

↑
Перезаписывание этого параметра
никакого эффекта не принесёт.

JDK-18: новое поведение

java.nio.file.Files



UTF-8

Charset.defaultCharset()

-Dfile.encoding



Пусто или "UTF-8"



"COMPAT"



-Dnative.encoding



Системная локаль

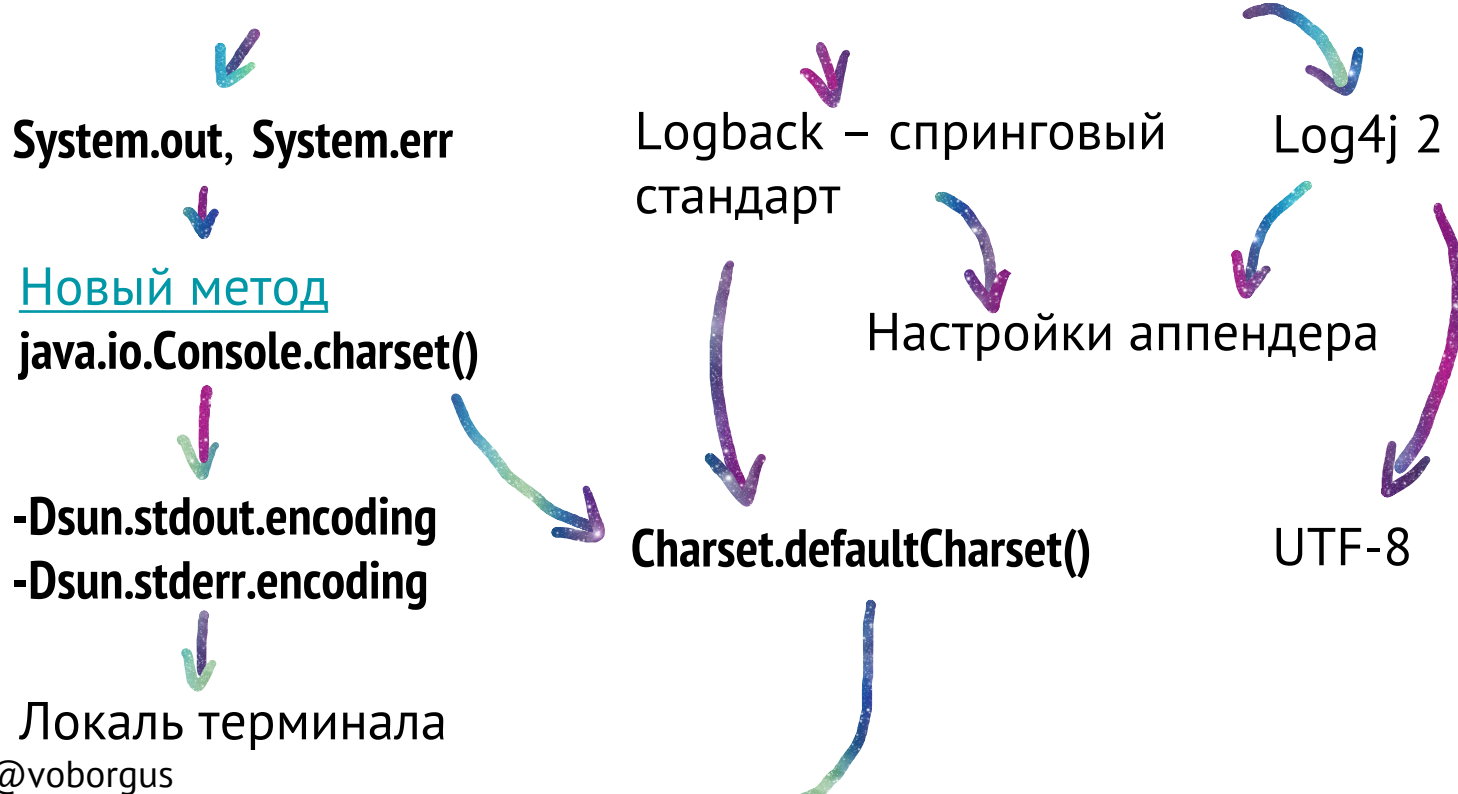
InputStreamReader/Writer
FileReader/Writer
PrintStream/Writer
Formatter
Scanner
String.getBytes()

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

UTF-8

Reader, Scanner

**Локаль
терминала***

`System.console().readLine()`

UTF-16**

`main(String[] args)`

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**

Что хорошего получили



UTF-8

по-умолчанию



-Dfile.encoding

стандартизация

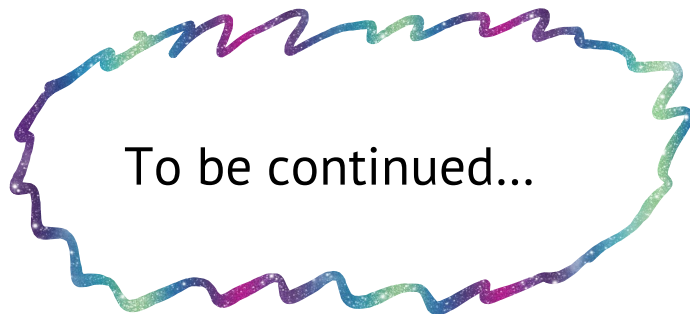


-Dnative.encoding

появление

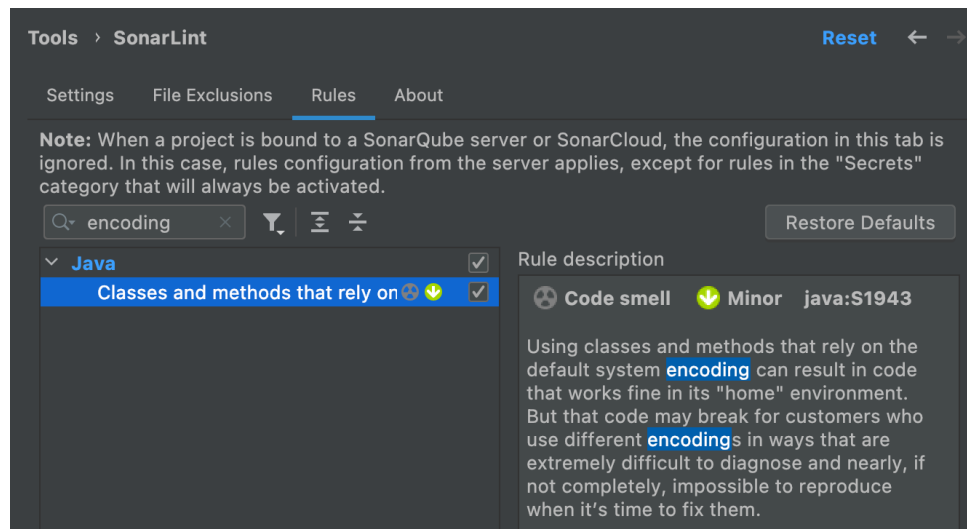
Какие, как говорится, зоны развития

- 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:

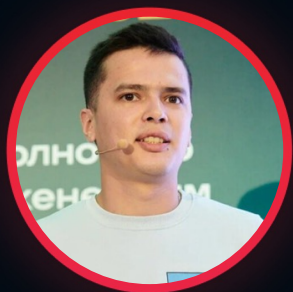
- Запустить с **files.encoding** и прогнать автотесты
- Поставить галочку в Sonarlint
- [RSPEC-1943](#) в SonarQube
- [Rewrite Recipe](#)
- [Forbidden APIs](#)
- [Error Prone](#)



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

JP-400 или UTF-8 кодировка по-умолчанию



Дмитрий
Сугробов

@voborgus

Joker<?>