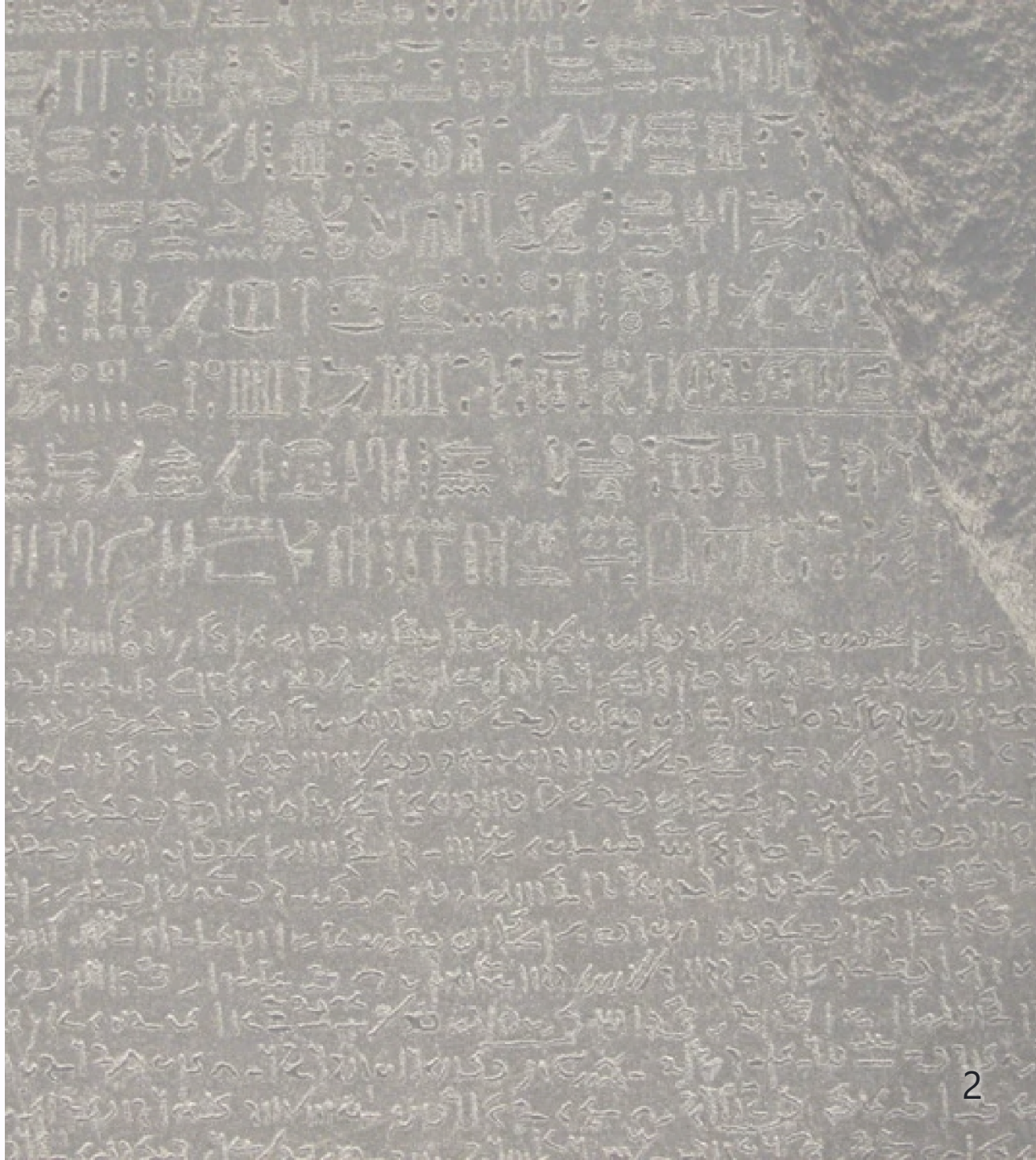
What a Character

Unicode Support in Java

Sualeh Fatehi

Unicode with Java



Java Tries to Cope

So,

- For compatibility with Java 1.0, a `char` is still 16 bits
- Java uses **surrogate pairs** for characters outside the BMP
- Java 5 APIs allow for `int` code points instead of surrogate pairs

Use the Character Class

```
int cp; // some value assigned...
if (Character.isLetter(cp))
// ...
```

INSTEAD OF

```
char ch; // some value assigned...
if ((ch >= 'a' && ch <= 'z') ||
    (ch >= 'A' && ch <= 'Z'))
// ...
```

Use the Character Class

```
int cp; // some value assigned...
if (Character.isDigit(cp))
// ...
```

INSTEAD OF

```
char ch; // some value assigned...
if (ch >= '0' && ch <= '9')
// ...
```

Use the Character Class: But Carefully?

```
// 'LATIN SMALL LETTER SHARP S' - ß
char germanChar = 'ß';
char germanCharUpper = Character
                       .toUpperCase(germanChar);
```

Result:

`germanCharUpper` is 'ß' (but we expect "SS")

(No exception is thrown, and no conversion is done!)

Use `int` Instead of `char` in Java

Many `Character` static methods take `int` code points

- `boolean isDigit(int codePoint)`
- `int toLowerCase(int codePoint)`

As do some `String` methods like

- `int indexOf(int ch)`
- `new String(int[] codePoints, int offset, int count)`

Iterate Over Code Points

Iterate over code points using `String.codePoints()` streams

```
"text".codePoints().forEach(System.out::print)
```

```
"text".codePoints().toArray()
```

Surrogates and Code Points

`Character` static methods allow conversions from surrogate pairs to code points

- `char[] toChars(int codePoint)`
- `boolean isSurrogatePair(char high, char low)`
- `int codePointAt(char[] a, int index)`

Beware of Breakage

- Some `String` methods, such as `substring(...)` and `length()` do not understand surrogates
- `StringBuilder` `delete(...)` method may not work as intended

Normalize Text

- Normalize text for comparison and sorting
- Java supports all the Unicode normalized forms
- Use the Normalizer class

For example, the normalized decomposition of "schön" is "scho\u0308n"

U+0308 is a 'COMBINING DIAERESIS', or ◌̈

Java Support For Unicode

Java Version	Unicode Version
JDK 1.0	Unicode 1.1
JDK 1.1, 1.2	Unicode 2.0
J2SE 5.0 *	Unicode 4.0
Java SE 8	Unicode 6.2
Java SE 11	Unicode 10.0
Java SE 17	Unicode 13.0

* Supplementary characters assigned in Unicode 3.1

Code Examples

Slides and all code examples are on GitHub

https://github.com/**sualeh/What-a-Character**