Advanced Java™ Globalization

Charles Hornig

Globalization Architect
IBM Corporation
http://www.ibm.com

TS-2873

Make Your Code Work Globally

‘All of the time’, not ‘most of the time’

Learn how to make your Java™ code work where your customers work—
All around the world, all of the time.

Agenda

Our Box of Tools

Global Problem Areas

Summary

Agenda

Our Box of Tools

- Java Platform, Standard Edition (Java SE platform)
- ICU4J
- Unicode

Global Problem Areas

Summary

Java SE Platform

- Globalization continues to improve
- Use the best tools you can!

Version	Features
Java 2 Platform, Standard Edition (J2SE™ platform) 1.4	`java.nio.charset.*` classes `java.util.Currency` class Unicode 3.0 Thai and Hindi locales
J2SE platform 5.0	Supplementary characters Unicode 4.0 Vietnamese locale
Java SE platform 6	Pluggable locale data Unicode normalization

Source: http://java.sun.com

ICU4J

International components for Unicode for Java technology

- Library provides Unicode and global support
  - Enhanced versions of Java SE platform functions
  - Functions not present in Java SE platform
- Open source with permissive license
- Provides new global support quickly

Source: http://www.icu-project.org

Unicode

"I'm using Java SE platform, so my code supports Unicode."

- Unicode support in Java SE platform has evolved
- Some Unicode functionality takes extra effort

Agenda

Our Box of Tools

Problem Areas

Summary

Agenda

Problem Areas

- **Characters**
- Sorting and searching
- User-visible persistent storage
- Case conversion
- Time zones
- Character encodings
- Currency

Characters

What is a ‘character’, anyway?

Code Unit	Code Point	
`\u0041`	U+0041 LATIN CAPITAL LETTER A	A
`\ud802`	n/a (high surrogate)	
`\ud802\udd00`	U+10900 PHOENICIAN LETTER ALF	
`\u0041\u0300`	U+0041 LATIN CAPITAL LETTER A U+0300 COMBINING GRAVE ACCENT	À

Example: Trim Leading Letters

```
// Fails for supplementary or combining characters
public static trim_bad1(String string) {
    char ch;
    for (int i = 0;
            i < string.length();
            i += 1) {
        ch = string.charAt(i);
        if (!Character.isLetter(ch))
            break;
        }
    return string.substring(i);
}
```

Example: Trim Leading Letters

```
// Fails for combining characters
public static trim_bad2(String string) {
    int ch;
    for (int i = 0;
            i < string.length();
            i += Character.charCount(ch)) {
        int ch = string.codePointAt(i);
        if (!Character.isLetter(ch))
            break;
    }
    return string.substring(i);
}
```

Example: Trim Leading Letters

```
// Correct
public static trim_good(String string) {
    BreakIterator iter =
        BreakIterator.getCharacterInstance();
    iter.setText(string);
    for (int i = iter.first();
            i != BreakIterator.DONE;
            i = iter.next()) {
        int ch = string.codePointAt(i);
        if (!Character.isLetter(ch))
            break;
    }
    if (i == BreakIterator.DONE)
        return "";
    else
        return string.substring(i);
}
```

Agenda

Problem Areas

- Characters
- **Sorting and searching**
- User-visible persistent storage
- Case conversion
- Time zones
- Character encodings
- Currency

Searching and Sorting

- Use **`java.lang.String`** methods
  - Exact comparison with ASCII string
  - Searching for ASCII punctuation
- Use **`java.text.Collator`** methods
  - Ordering strings for display
  - Comparisons of non-ASCII strings
- Use **`com.ibm.icu.text.StringSearch`** methods
  - Searching using **`Collator`** for comparisons

Agenda

Problem Areas

- Characters
- Sorting and searching
- **User-visible persistent storage**
- Case conversion
- Time zones
- Character encodings
- Currency

User-Visible Persistent Storage

- Storing persistent data that must also be human-readable
- Can't use binary representation
- Can't use correct local format
- Choose a format that is:
  - Locale-independent
  - Readable
  - Standard

Example: Export to .properties File

```
// Writes localized representation which cannot be parsed
public static void toCsv_bad
    (PrintWriter out, String prop, double val)
{
    out.format("%s: %g", prop, val);
}

// Correct
public static void toCsv_good
    (PrintWriter out, String prop, double val)
{
     out.format("%s: %s", prop, String.valueOf(val);
}
```

Agenda

Problem Areas

- Characters
- Sorting and searching
- User-visible persistent storage
- **Case conversion**
- Time zones
- Character encodings
- Currency

Case Conversion

- Don’t do it
- If you must, do it all at once
- Respect language rules

Example: Check for Suffix

```
// Inefficient; doesn't respect locale or Unicode rules
public static boolean endsWithIgnoreCase_bad
    (String string, String suffix)
{
    return string.toLowerCase()
                 .endsWith(suffix.toLowerCase());
}
```

Example: Check for Suffix

```
// Doesn't respect locale or Unicode rules
public static boolean endsWithIgnoreCase_unsafe
    (String string, String suffix)
{
    int toffset = (string.length() - suffix.length());
    return string.regionMatches(true, toffset, suffix, 0,
                                suffix.length());
}
```

Example: Check for Suffix

```
import com.ibm.icu.text.BreakIterator;
import com.ibm.icu.text.StringSearch;

// Correct
public static boolean endsWithIgnoreCase_safe
    (String string, String suffix)
{
    StringSearch ss = new StringSearch(suffix, string);
    ss.getCollator().setStrength(Collator.SECONDARY);
    ss.setOverlapping(true);
    BreakIterator i = BreakIterator.getCharacterInstance();
    ss.setBreakIterator(i);
    if (ss.last() == StringSearch.DONE)
        return false;
    else
        return ((ss.getMatchStart() + ss.getMatchLength())
                == string.length());
}
```

Example: Convert String to Upper Case Array

```
// Fails for German sharp s (ß)
public static char[] toArray_bad(String string) {
    char[] chars = string.toCharArray();
    for (int i = 0; i < chars.length; i++)
        chars[i] = Character.toUpperCase(chars[i]};
    return chars;
}

// Correct
public static char[] toArray_good(String string) {
    String upper = string.toUpperCase();
    return upper.toCharArray();
}
```

Example: Convert String to Upper Case Array in Client Locale

```
public static char[] toArray_remote
    (String string, Locale locale)
{
    String upper = string.toUpperCase(locale);
    return upper.toCharArray();
}
```

Agenda

Problem Areas

- Characters
- Sorting and searching
- User-visible persistent storage
- Case conversion
- **Time zones**
- Character encodings
- Currency

Time Zones

- Presenting times
- Interpreting time strings
- Storing time strings
- Time zone vs. time zone offset

Example: Time String for Log

```
// Localized date is not parseable
public static String logTime_bad1() {
    DateFormat fmt = new DateFormat.getDateTimeInstance();
    return fmt.format(new Date());
}
// Fails if log read in different time zone
public static String logTime_bad2() {
    SimpleDateFormat fmt
        = new SimpleDateFormat("yyyy-MM-dd'THH:mm:ss");
    return fmt.format(new Date());
}
// Correct
public static String logTime_good() {
    SimpleDateFormat fmt
        = new SimpleDateFormat("yyyy-MM-dd'THH:mm:ssZ");
    return fmt.format(new Date());
}
```

Agenda

Problem Areas

- Characters
- Sorting and searching
- User-visible persistent storage
- Case conversion
- Time zones
- **Character encodings**
- Currency

Character Encodings

- Use `java.nio.charset` classes
- Do Java Native Interface (JNI™) conversions on the Java technology side

Example: Convert String to/from Named Encoding

```
// Corrupts data on errors
public static byte[] toCodePage_bad(String charset,
                                    String string) {
    return string.getBytes(charset);
}

// Fails to detect corrupt data
public static String fromCodePage_bad(String charset,
                                      byte[] bytes) {
    return new String(bytes, charset);
}
```

Example: Append String to Text File in Named Encoding

```
// Corrupts data on errors
public static void toFile_bad(String charset,
                              String filename,
                              String string) {
    FileOutputStream stream
        = new FileOutputStream(filename, true);
    OutputStreamWriter writer
        = new OutputStreamWriter(stream, charset);
    writer.write(string, 0, string.length());
    writer.close();
}
```

Example: Convert String to Named Encoding

```
// Correct
public static byte[] toCodePage_good(String charset,
                                     String string) {
    Charset cs = Charset.forName(charset);
    CharsetEncoder coder = cs.newEncoder();
    ByteBuffer bytebuf
        = coder.encode(CharBuffer.wrap(string));
    byte[] bytes = new byte[bytebuf.limit()];
    bytebuf.get(bytes);
    return bytes;
}
```

Example: Convert String From Named Encoding

```
// Correct
public static String fromCodePage_good(String charset,
                                        byte[] bytes) {
    Charset cs = Charset.forName(charset);
    CharsetDecoder coder = cs.newDecoder();
    CharBuffer charbuf
        = coder.decode(ByteBuffer.wrap(bytes));
    return charbuf.toString();
}
```

Example: Append String to Text File in Named Encoding

```
// Correct
public static void toFile_good(String filename,
                               String string) {
    Charset cs = Charset.forName(charset);
    CharsetEncoder coder = cs.newEncoder();
    FileOutputStream stream
        = new FileOutputStream(filename, true);
    OutputStreamWriter writer
        = new OutputStreamWriter(stream, coder);
    writer.write(string, 0, string.length());
    writer.close();
}
```

Example: Native Method to Call getenv()

```
// Fails for non-ASCII data
extern "C"
jstring Java_sample_getenv(JNIEnv* env,
                           jobject obj,
                           jstring key) {
    char const* pKey = env->getStringUTFChars(env, key 0);
    char* pVal = getenv(pKey);
    env->releaseStringUTFChars(env, key, pKey);
    return env->newStringUTF(env, pVal);
 }
```

Example: Native Method to Call getenv()

```
// Java: Native method to call getenv()
// Note:
//  Unsafe conversion functions used here for clarity
//  Applications should use the java.nio.charset.* classes
class sample {
    native String cgetenv(String key);
    String getenv(String key) {
        byte[] bkey = key.getBytes();
        byte[] bval = cgetenv(bkey);
        return new String(bval);
    }
}
```

Example: Native Method to Call getenv()

```
// C++: Native method to call getenv()
extern "C"
jbytearray Java_sample_cgetenv(JNIEnv* env,
                               jobject obj,
                               jbyteArray key) {
    char const* pKey
        = env->GetByteArrayElements(env, key 0);
    char* pVal = getenv((char const*)pKey);
    env->ReleaseByteArrayElements(env, key, pKey);
    jsize len = strlen(pVal);
    jbyteArray val = env->NewByteArray(env, len);
    env->SetByteArrayRegion(env, val, 0, len,
                            (jbyte*)pVal);
    return val;
 }
```

Agenda

Problem Areas

- Characters
- Sorting and searching
- User-visible persistent storage
- Case conversion
- Time zones
- Character encodings
- **Currency**

Currencies

- Default currency for locale is a trap
- Always track currencies explicitly
- Use ISO 4217 codes in persistent storage

Agenda

Our Box of Tools

Problem Areas

Summary

Summary

- It's possible to write global code
- It takes a little extra effort
- It's worth it

For More Information

- Java SE platform
  - http://java.sun.com/javase/technologies/core/basic/intl
- ICU4J
  - http://www.icu-project.org
- Unicode
  - http://www.unicode.org
- IBM globalization
  - http://www.ibm.com/software/globalization

Q&A

Charles Hornig
chornig@us.ibm.com

Advanced Java™ Globalization

Charles Hornig

Globalization Architect
IBM Corporation
http://www.ibm.com

TS-2873