JSR 354

Java Money and Currency API

Spec lead – Victor Grazi – Credit Suisse
Summary

• This JSR will provide a money and currency API for Java, targeted at all users of currencies and monetary amounts in Java.

• The API will provide support for standard ISO-4217 and custom currencies, and a representation of a monetary amount.

• It will support currency arithmetic, even across different currencies, and will support foreign currency exchange.

• Additionally, implementation details surrounding serialization and thread safety are to be considered.
Why is this needed?

- Monetary values are a key feature of many applications.
- The existing java.util.Currency class is strictly a structure used for representing ISO-4217 standard currencies.
- No standard value type to represent a monetary amount.
- No support for currency arithmetic.
Challenges

• Keep it simple - Remember the most common use case - adding currency values e.g. in an e-commerce app.
• Performance - how to support low latency applications, such as High Frequency Trading apps
• Precision - There can potentially be differing precisions specified for arithmetic, currency exchange and formatting
• Formatting - Requirement should be to use existing Number Formats. However there is no existing format for representing for example Indian Rupees: which might look something like this: 12,34,00,000
• Natural language support for non-decimal valuations for example Lakhs and Crores.
• 1 Lakh = 100,000, 1 Crore = 100 Lakh. (12,34,56,000.21 is written 12 Crore, 34 Lakh, 56 Thousand Rupees and 21 Paise)
• Support for non-standard rounding rules
Non-standard rounding rules

• It is a big world, and each country has its regulations and cultural nuances for expressing currencies in natural language, rounding policies, groupings, etc.

• For example, in Argentina rounding is prescribed for the third digit after the decimal point:
  - If the third digit is 2 or less, change it to 0 or drop it.
  - If the third digit is between 3 and 7, change it to 5.
  - If the third digit is 8 or more, add one to the second digit and drop the third digit or change it to 0.

Argentina Rounding Examples

<table>
<thead>
<tr>
<th>Original Number</th>
<th>Rounded</th>
<th>Notes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>123.452</td>
<td>123.45</td>
<td>third digit&lt;3</td>
<td>round down</td>
</tr>
<tr>
<td>123.456</td>
<td>123.455</td>
<td>3&lt;=third digit&lt;=7</td>
<td>change to 5</td>
</tr>
<tr>
<td>123.459</td>
<td>123.46</td>
<td>third digit&gt;=8</td>
<td>round up</td>
</tr>
</tbody>
</table>

Switzerland uses a similar rounding strategy.
Risks

Plan – “Money and Currency” is a formidable and dynamic category with regional dependencies, requiring programmers, business users, international accountants, and attorneys.

Risk - Incomplete or outdated spec
Mitigation- Resulting API should be flexible, let application developer change implementation as needed.

Plan - Large expert group to tackle each area.
Risk - Attrition and incomplete specification will require maintenance patches.
Mitigation - Keep an active team after the release. The API can function standalone, and so the risk can be mitigated by releasing standalone patches.

Plan - There is also a Java dependency on JDK NumberFormat.
Risk - NumberFormat may require coordinated modifications.
Mitigation – Coordinate with JDK release; supply NumberFormats that augment the JDK classes.
Initial Expert Group

- Credit Suisse
- Goldman Sachs
- Stephen Colebourne
- Ben Evans
- Werner Keil
Supporting this JSR

• Credit Suisse
• Caxton Associates
• Goldman Sachs
• JP Morgan/Chase
• London Java Community
• Stephen Colebourne
• Werner Keil
Schedule

• Targeted to Java 9
• With back-port to previous versions
Important Links

• Prezi of this presentation: http://prezi.com/no48uqcsyhjy/jsr-354/

• The JSR: http://jcp.org/en/jsr/summary?id=354

• java.net project: http://java.net/projects/javamoney/
At long last!

World class money and Currency Support in Java