Show me (se/ee) the Money!

Better Financial API Support for the Java Platform

Werner Keil

Princeton, New Jersey

September 10th, 2009

The Problem

- No real standardization for Financial APIs
- Closed Source Vendors may come and go or simply abandon products leaving potentially Thousands of customers without support overnight
    - See Microsoft Money as the most infamous Financial Software example

Financial Crisis

- Crisis was caused by greed and speculation
- But also to a large extent misinformation
- Lack of compatibility between “hostile” markets and competitors
- No real Open Standards aside from very few
  - E.g. SWIFT or SEPA, etc.
- No Transparency

CATMedia

Java – The 90s

- Early examples by industry visionaries showed what could have been done back in 1996/7 !!
    - JUnit samples using Money and Currency by Beck and Gamma However, even those luminaries could nor or did not want to “shine” outside their own context and the use case of unit tests only.
    - Roedy Green from Canadian Mind Products has provided not only great books and articles, he actually said to have offered some advise and contributions, but Sun or JavaSoft at this time did not accept them.

Java – Money Class by JUnit

```
public class Money {
  private double amount;
  private String currency;
  public Money(double amount, String currency) {
        this.amount = amount;
        this.currency = currency;
  }
  public double getAmount() {
        return this.amount;
  }
  public String getCurrency() {
        return this.currency;
  }
  public Money add(Money oMoney) {
        return new Money(this.amount +
        oMoney.getAmount(),this.currency);
  }
}
```

Java Currency

- The class java.util.Currency is best known to most
- It was added with 1.4
- Primarily offering the “View” element of common MVC paradigm.
- No value (“Model”) taken into consideration. That is left entirely to developers where BigDecimal is usually recommended due to precision.

Java Currency (2)

- The class is the lowest common denominator of ISO 4217 definitions for currency handling
- Most likely not in a separate JSR (if so only that for SE 1.4)
- Inconsistent method signature
  - getSymbol() vs.
  - getCurrencyCode()
  - → what other than the **Currency code** would a Currency class use? Or why not call the other method getCurrencySymbol() ?

Time and Money (DDD)

- Another example to showcase concepts of Domain Driven Design by Eric Evans
- It is said to still work by SF users, but its limitation to Dollar and reduce it to a showcase and vehicle for their books and trainings, far from a really useful framework
- Evans was offered to join EG at least by JSR-310 but has turned down the offer

Solutions today

Handling currency calculations in Java business application

- March 29, 2009 — Venkat
  Recently I saw a weird floating issue in Java application which made our currency calculation wrong.

Solutions today (2)

- *Can you guess what would be the output of the code below?*

```
System.out.println(38.0 - 26.6);
```

- *Sun recommends to use* BigDecimal for currency calculation in Java. For example above code can be changed as below:

```
System.out.println(BigDecimal.valueOf(38.0).subtract(BigDeci
mal.valueOf(26.6)));
```

- Is this pretty or efficient ?!

Possible Alternatives

- ICU4J (Eclipse)
- Apache Commons Money / JodaMoney
- Grails Currency
- Unit Framework by JSR-275
- Eclipse Financial Platform
  - Formerly known as OFMP
- Commercial/Product-based implementations
  - MoneyDance
  - JFire (Eclipse RCP Open Source CRM)

With Support for Conversion,...

- ICU4J (Eclipse)
- Apache Commons Money / JodaMoney
- Grails Currency
- Unit Framework by JSR-275
- Eclipse Financial Platform
  - Formerly known as OFMP
- Commercial/Product-based implementations
  - MoneyDance
  - JFire (Eclipse RCP Open Source CRM)

JSR-275

Case Study: Monetary System

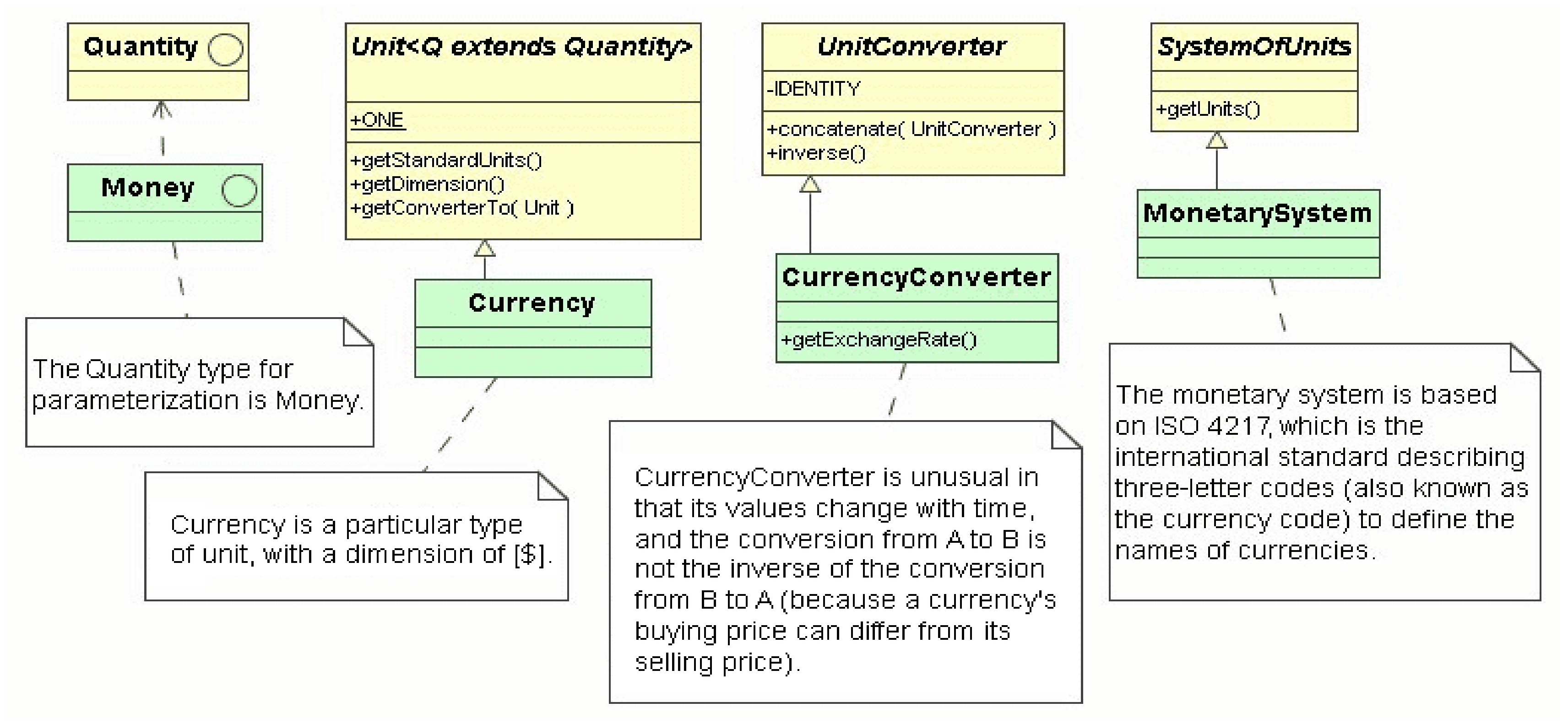
Monetary systems are not subject to JSR-275, but this illustrates, how easily the framework can be extended to non physical or scientific quantities.

Such extension can be valuable by leveraging the framework's capabilities (formatting, conversion,…)

and applying its usefulness beyond what java.util.Currency now provides

JSR-275

Currency Conversion Classes



The Quantity type for parameterization is Money.

Currency is a particular type of unit, with a dimension of [$].

CurrencyConverter is unusual in that its values change with time, and the conversion from A to B is not the inverse of the conversion from B to A (because a currency's buying price can differ from its selling price).

The monetary system is based on ISO 4217, which is the international standard describing three-letter codes (also known as the currency code) to define the names of currencies.

Let me show you the money…

DEMO

Let‘s talk

Q & A