elements

Interest Rates Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

Contents

List of Figures

List of Tables

Package Description

Interest rates represent the time value of money. Interest rates, in the elements model represent a rate that transforms an amount of some currency at one date into an amount of the *same* currency at another date.

Interest rates can be grouped into curves called *yield curves*. A yield curve can derive an interest rate from any date to any date. Yield curves can, therefore, be used to discount a series of future (and past) cashflows to a common date. As a special case, yield curves can be used to discount a set of cashflows to the present, giving a *net present value* or *NPV*.

The interest rate package is considerably more concrete than the basic rates package that it depends upon. For a newcomer, we suggest that you read this package first and then move on to the more abstract basic rates package.

1 Interfaces

1.1 BasicYieldCurveSpecification

Basic yield curves are constructed from more basic rates, converted to interest rates. This interface provides an interface for the specification of these rates.

1.1.1 Relationships

	Class	Description	Notes
⇑	Validatable		
⇑	Identifiable		
↓	BasicYieldCurveSpecification-Model §3.4		
↓	BasicYieldCurveReferenceData-Model §3.3		
↔	BasicYieldCurveReferenceData-Model §3.3	model 0..1	
↔	BasicYieldCurveConstructorSer-vice §4.1	specification	

⇑:Inherits ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

1.1.2 Operations

Currency currency() currency

The currency that this yield curve is for. Return the currency for the interest rates that this curve uses.

DateBasis dateBasis() dateBasis

The date basis for interpolation/extrapolation. Return the date basis that is to be used when calculating date distances for the purposes of extrapolation and interpolation. Individual source rates may use different date bases when provided.

Collection<RateSpecifier> sources() sources

The source rates for this specification. Return the collection of rate specifiers that describe this yield curve.

InterestRateQuotationMethod quotationMethod() quotation-Method

The common quotation method to use when interpolating or extrapolating interest rates. Return the date basis that is to be used when calculating date distances for the purposes of extrapolation and interpolation. Individual source rates may use different date bases when provided.

Reportable validate() validate

Validate this yield curve. A yield curve specification is valid if the currency, date basis and quotation method are supplied and each source rate specifier satisfies the following requirements:

- The logical rate specifier is an InterestRateSpecifier §1.7 with both the from- and to-dates present.
- If the derivation is a basic derivation, then there is also a quotation method.
- The currency of the logical rate specifier is the same currency as the specification's currency.

In addition, warnings should be added wherever:

- The from- and to-periods are not defined, but exact dates are supplied instead. (This usually means a yield curve with a limited life-span.)
- The date basis for the rate specifier is different to the date basis for the curve.

1.2 InterestRateCompoundingFrequency

The rate at which a yield or discount rate compounds. This is a utility interface used to allow interest rates which need compounding conventions to choose between conventions.

1.2.1 Relationships

	Class	Description	Notes
⇑	Comparable		
⇑	ValueSemantics		
↓	InterestRateCompoundingFrequency-Model §3.6		

⇑:Inherits ↓:Realized by

1.2.2 Operations

«Static Method» InterestRateCompoundingFrequency canonical() canonical

The standard quotation frequency. Return a ContinuousCompoundingFrequencyModel §3.7. This model is assumed to use an Actual/Actual date basis.

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical
r: Number The rate (as a yield) to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this quotation frequency into the canonical quotation frequency. Return a yield that would give the same amount of interest as the supplied yield, but using a continuous compounding frequency.

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this the canonical quotation frequency into this quotation frequency. Return a yield in this frequency that would give the same amount of interest as the supplied yield, which has a continuous compounding frequency..

Number discountToYield(Number r, LogicalRateSpecifier specifier) discountToYield
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a discount rate to a yield. Return a yield that would give the same amount of interest as the supplied discount rate, using whatever day count is required.

Number yieldToDiscount(Number r, LogicalRateSpecifier specifier) yieldToDiscount
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield to a discount rate. Return a discount rate that would give the same amount of interest as the supplied yield, using whatever day count is required.

1.3 InterestRatePiece

A specialization of the RatePiece interface for managing interest rates.

1.3.1 Relationships

	Class	Description	Notes
⇑	RatePiece		
↓	InterestRatePieceModel §3.12		

⇑:Inherits ↓:Realized by

1.4 InterestRatePremiumQuotationMethod

Interest rate premiums represent additional margins over some interest rate. The premiums may be used to reflect additional risk, profit margins or internal trading margins.

1.4.1 Relationships

	Class	Description	Notes
⇑	QuotationMethod		
⇑	ValueSemantics		
↓	InterestRateBasisPointsModel §3.5		

⇑:Inherits ↓:Realized by

1.4.2 Operations

Boolean isMargin() isMargin

Is this rate in margin form? Return true.

Boolean isCanonical() isCanonical

Is this the canonical representation? Return true if the rate is a basis point representation, return false otherwise.

String type() type

The type of rate that this quotation method is for. Return "Interest Rate"

Number addPremium(InterestRateQuotationMethod baseQuote, Number base, Number premium) addPremium

baseQuote: InterestRateQuotationMethod The quotation method for the base interest rate.
base: Number The base interest rate.
premium: Number The premium to add.

Add a premium to an interest rate. To add a premium, the quotation methods for the base interest rate and the premium must match. The returned rate uses the same quotation method as the base interest rate.

«Static Method» QuotationMethod canonical() canonical

Canonical quotation method. Return a InterestRateBasisPointsModel §3.5 object.

1.5 InterestRateQuotationMethod

Interest rate quotation methods have two parts: a style, indicating how the number representing the interest rate is to be interpreted, and a frequency, indicating the number of times that the interest rate is compounded over some fixed period.

The canonical representation for an interest rate is in terms of annualized yields.

1.5.1 Relationships

	Class	Description	Notes
⇑	QuotationMethod		
⇑	ValueSemantics		
↓	InterestRateQuotationMethod-Model §3.13		

⇑:Inherits ↓:Realized by

1.5.2 Operations

Boolean isCanonical() isCanonical

Is this the canonical representation? Return true if the rate is a continuously compounding yield, false otherwise.

String type() type

The type of rate that this quotation method is for. Return “Interest Rate”.

InterestRateCompoundingFrequency frequency() frequency

The compounding frequency of this interest rate. Return the compounding frequency of this interest rate.

Boolean isInterestRate() isInterestRate

Is this an interest rate? Interest rates represent quotation methods that represent additions to or subtractions from initial or final amounts, rather than ratios between the initial and final amounts. Yields and discount rates are example interest rates; discount factors are an example of non-interest rates.

Number discountFactor(Number r, LogicalRateSpecifier specifier) discountFactor

r: Number The rate to convert into a discount factor.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate in this quotation format into a discount factor. Return the equivalent discount factor for a rate, r, quoted using this quotation method, running from the from-date to the to-date, using the specifier's date basis to calculate the term in years, t.

If μ is the canonical equivalent of r, then return $e^{-\mu t}$.

«Static Method» QuotationMethod canonical() canonical

Canonical quotation method. Return a YieldQuotationMethodModel §3.18 with a frequency of a ContinuousCompoundingFrequencyModel §3.7. (Continuously compounding yield)

From the definitions of the various interest rate operations and yield curves, it would appear that discount factors might be a better choice of canonical rate. Discount factors, however, suffer from a singularity when the from- and to-dates are equal.

1.6 InterestRateQuote

A rate quote specialized to handle interest rates.

1.6.1 Relationships

	Class	Description	Notes
⇑	RateQuoteModel		
⇑	RateQuote		
↓	InterestRateQuoteModel §3.20		

⇑:Inherits ↓:Realized by

1.6.2 Operations

Number discountFactor(LogicalRateSpecifier specifier) discountFactor
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

The equivalent discount factor. Return the discount factor constructed from the total quote.

1.7 InterestRateSpecifier

Interest rates are used for two purposes. The first use of interest rates is to specify the price of a loan or deposit: the amount of money that is paid for the right to use some other amount of money for a period of time. The second use is to represent the *time value of money*: money in the future is worth less than money now.

These two uses are interrelated. The time value of money can be essentially viewed as a consequence of the fact that money now can be invested and earn interest, which means that there will be slightly more money by the time that future money becomes available.

Interest rates are specified for a commodity and an environment and run between two dates. In addition, interest rates in a particular currency may vary according to the place in which the loan is being made and the party to whom the loan is being made.

1.7.1 Relationships

	Class	Description	Notes
⇑	LogicalRateSpecifier		
↓	InterestRateSpecifierModel §3.21		

⇑:Inherits ↓:Realized by

1.7.2 Operations

Collection<LogicalRateFormalParameter> formalParameters() formalParameters

The possible parameters for this rate specifier. Return the following set of parameters:

Name	Type		Description
commodity	Commodity	discrete	The commodity (currency)
location	Location	discrete	The location which is using this currency
from-date	Date	continuous	The start date of the interest rate.
to-date	Date	continuous	The end date of the interest rate.
from-period	Period	discrete	The start period.
to- period	Period	discrete	The end period.

The from- and to- dates use the LogicalRateDateFormalParameter interface.

Currency currency() currency

The currency that the interest rate is for. Return the currency that is used for this interest rate.

Date fromDate() fromDate

The start date. Return the date at which a notional loan at this interest rate commences.

If fromPeriod() returns a non-nil value, then this date must be equal to the from period added to the current processing date, using the date classifier supplied by the currency. The location does not play any part in the date classifier.

Date toDate() toDate

The end date. Return the date at which a notional loan at this interest rate matures.

If toPeriod() returns a non-nil value, then this date must be equal to the to period added to the current processing date, using the date classifier supplied by the currency. The location does not play any part in the date classifier.

Period fromPeriod() fromPeriod

The start period. If there is no from period, then return nil. Otherwise, return the period from today to when a notional loan in this interest rate commences.

Period toPeriod() toPeriod

The end period. If there is no to period, then return nil. Otherwise, return the period from today to when a notional loan in this interest rate matures.

Location location() location

The location of the interest rate. Return nil if the notional loan for this currency is location-neutral. If the interest rate is for a specific place, return the location.

Collection<LogicalRateActualParameter> actualParameters() actualParameters

The set parameters for this rate specifier. Return a collection of actual parameters matching the parameter name against the operation of the same name. Do not include any actual parameters for operations that return nil.

dateBasis dateBasis() dateBasis

The date basis. Return the date basis for calculating day counts and year counts.

Party party() party

The party to whom a loan is being made. Return nil if the notional loan for this currency is party-neutral (usually meaning that the associated risk is assumed to be the risk associated with the government that controls the currency). If the interest rate is for a loan to a specific party, return the party.

1.8 PointInterestRate

A single interest rate, defined for a single currency and two dates.

1.8.1 Relationships

	Class	Description	Notes
⇑	PointRate		
↓	BasicInterestRateModel §3.1		

⇑:Inherits ↓:Realized by

1.8.2 Operations

RateQuote mid() mid

The mid component. The mid component is calculated by taking the mean of the bid and ask component *discount factors*. The resulting mid component is quoted using the same quotation method as the bid component.

If the bid and ask components are constructed from a number of pieces, the mid pieces are calculated by calculating the discount factors incrementally for the base rate plus each piece and subtracting the resulting total rate to give a new margin.

As an example, using a (annualized, compounding) bid rate of $6.52\% + 100bp + 100bp$ and an ask rate of $6.55\% + 120bp + 110bp$ over a period of 1.5 years, then: The base discount factor is given by $\frac{(1+0.0652)^{-1.5}+(1+0.0655)^{-1.5}}{2} = 0.9094 \equiv 6.53\%$. The next discount factor is given by $\frac{(1+0.0752)^{-1.5}+(1+0.0775)^{-1.5}}{2} = 0.8955 \equiv 7.63\% \equiv +100bp$. The next discount factor is given by $\frac{(1+0.0852)^{-1.5}+(1+0.0885)^{-1.5}}{2} = 0.8826 \equiv 8.68\% \equiv +105bp$. So the mid rate is $6.53\% + 100bp + 105bp$.

Instrument buy(Instrument quantity) buy
quantity: Instrument The quantity to convert.
Raises: RateConversionException

Buy one quantity of an instrument by paying some other quantity of the instrument. See PointRate for an overview of this operation.

If this rate is mine (ie. is from within the system; responds to isMine with true) and the quantity is greater than zero, use the bid rate. Any change of one of the listed characteristics flips from bid to ask. Another change flips back from ask to bid.

The quantity, in this case, must be a SimpleCashflow with a commodity equal to the primary commodity and a date equal to the from- or to-date. Let DF be the discount factor for the appropriate bid or ask rate. Let n be the quantity of the quantity SimpleCashflow. If the supplied quantity has a date equal to the to-date, then return a SimpleCashflow with the same commodity as the primary commodity, the from-date and $n \times DF$ as the quantity. If the supplied quantity has a date equal to the from-date, then return a SimpleCashflow with the same commodity as the primary commodity, the to-date and n/DF as the quantity.

Instrument sell(Instrument quantity) sell
quantity: Instrument The quantity to convert.
Raises: RateConversionException

Sell one quantity of an instrument by paying some other quantity of the instrument. See PointRate for an overview of this operation.

If this rate is mine (ie. is from within the system; responds to isMine with true) and the quantity is greater than zero, use the ask rate. Any change of one of the listed characteristics flips from ask to bid. Another change flips back from bid to ask.

The quantity, in this case, must be a SimpleCashflow with a commodity equal to the primary commodity and a date equal to the from- or to-date. Let DF be the discount factor for the appropriate bid or ask rate. Let n be the quantity of the quantity SimpleCashflow. If the supplied quantity has a date equal to the to-date, then return a SimpleCashflow with the same commodity as the primary commodity, the from-date and $n \times DF$ as the quantity. If the supplied quantity has a date equal to the from-date, then return a SimpleCashflow with the same commodity as the primary commodity, the to-date and n/DF as the quantity.

1.9 YieldCurve

A yield curve is a specialization of a rate curve for interest rates. Yield curves are currency-specific and, essentially, return the discount factor for that currency between two dates; the from- and to-dates.

Although the from- and to-dates appear to be completely independent parameters, they actually reflect the density nature of interest rates. The discount factors between dates are transitive in nature. If we have three dates $d_1 < d_2 < d_3$ and discount factors D_{12}, D_{23} and D_{13} between the various dates, then $D_{13} = D_{12}D_{23}$.

The 30-day day count bases can lead to counter-intuitive interest rates as a result of variations in day counts. See the YearModelActual class for further discussion of this problem. Using the example of a 30/360 date basis and d_1 equal to 31-Mar-1997, d_2 equal to 1-Apr-1997 and d_3 equal to 31-Jul-1997, then the term in years between d_1 and d_3 is $120/360$, the term in years between d_1 and d_2 is $1/360$ and the term in years between d_2 and d_3 is $120/360$. If a level interest rate is used, then $D_{12} = (1+i)^{-\frac{1}{360}}$, $D_{23} = (1+i)^{-\frac{120}{360}}$ and $D_{13} = (1+i)^{-\frac{120}{360}}$. Clearly, $D_{13} \neq D_{12}D_{23}$.

However, interest rates are usually constructed from sets of interest rate points. In the example above, the interest rates could come from a curve constructed from a point which has 10% from 31-Mar-1997 to 1-Apr-1997 and 10% from 1-Apr-1997 to 31-Jul-1997. The actual interest rate for 31-Mar-1997 to 31-Jul-1997 is then calculated by combining the discount factors from the two periods and converting the resulting discount factor into an interest rate of 10.06%.

Yield curves are always assumed to initially calculate interest rates between a specified date, the *origin date*, and some other date. The origin date is denoted by d_0. Interest rates between two arbitrary dates, d_1 and d_2 are calculated by computing discount factors: $D_{12} = D_{02}/D_{01}$.

1.9.1 Relationships

	Class	Description	Notes
⇑	RateCurve		
⇓	BasicYieldCurve §1.10		

⇑:Inherits ⇓:Inherited by

1.9.2 Operations

Collection<LogicalRateFormalParameter> formalParameters() formalParameters

The parameters of the curve. Return the currency and date basis formal parameters. See the InterestRateSpecifier §1.7 interface for details.

Date originDate() originDate

The origin date from which this curve is constructed. Return the date from which all interpolated interest rates and discount factors are computed.

InterestRate originTo(Date toDate) originTo

toDate: Date The date to which this interest rate runs.

The interest rate from the origin date to another date. Return the interest rate running from the origin date to the supplied toDate.

InterestRate fromTo(Date from, Date to) fromTo

from: Date

to: Date

The interest rate between two dates.

If the interest rate from the origin date to the from date has a discount factor of D_{01} and the interest rate from the origin date to the toDate has a discount factor of D_{02} then return the interest rate that has a discount factor of D_{02}/D_{01}.

If the origin date and the toDate are equal, then the returned discount factor will always be 1. At this point, it will not be possible to calculate a unique interest rate. In this case, the interest rate is estimated to be the interest rate derived from the interest rate from the fromDate - 1 calendar day to the toDate — the overnight rate.

PointRate value(Collection<LogicalRateActualParameter> parameters) value

parameters: Collection<LogicalRateActualParameter>

Raises: RateSpecificationException

Get the value at some point on the curve. The supplied parameters must fix the from-date and to-date parameters. If specified, the supplied parameters must match the currency and date basis of the yield curve; raise a RateSpecificationException exception if the parameters do not match.

Return the result of the fromTo() operation, using the from-date and to-date from the supplied parameters as the arguments.

1.10 BasicYieldCurve

A basic rate curve that is a yield curve. The curve interpolation and extrapolation machinery is used to construct a curve of interest rates or discount factors running from the origin date to other dates.

1.10.1 Relationships

	Class	Description	Notes
⇑	BasicRateCurve		
⇑	YieldCurve §1.9		
↓	BasicYieldCurveModel §3.2		

⇑:Inherits ↓:Realized by

2 Service Interfaces

2.1 YieldCurveConstructor

A yield curve constructor is used to build a yield curve from an assortment of interest rates, bond prices, FRA rates and anything else that can be converted into an implied interest rate.

Once collected, these rates can be interpolated and extrapolated into a suitable curve. Since some of the source rates may need to use a partially built yield curve to imply their own equivalent interest rates, construction may be a complex, iterative process.

2.1.1 Relationships

	Class	Description	Notes
⇑	RateConstructor		
↓	BasicYieldCurveConstructorService §4.1		

⇑:Inherits ↓:Realized by

3 Classes

3.1 BasicInterestRateModel

A concrete implementation of the PointInterestRate interface. This class holds components that implement the InterestRateQuote §1.6 interface.

3.1.1 Relationships

	Class	Description	Notes
⇑	BasicPointRateModel		
↑	BasicPointRate		
↑	PointInterestRate §1.8		

⇑:Inherits ↑:Realizes

3.2 BasicYieldCurveModel

An implementation of the BasicYieldCurve interface using the BasicRateCurveModel as a basis. The elements of the curve are restricted to interest rates, with a consistent date basis.

3.2.1 Relationships

	Class	Description	Notes
⇑	BasicRateCurveModel		
↑	BasicYieldCurve §1.10		

⇑:Inherits ↑:Realizes

3.2.2 Attributes

originDate: Date The origin date for this yield curve.

3.3 BasicYieldCurveReferenceDataModel

An implementation of the BasicYieldCurveSpecification interface that can be managed as a piece of reference data. The interface is realized by holding an actual model object and delegating to that model.

3.3.1 Relationships

	Class	Description	Notes
⇑	ReferenceDataModel		
↑	BasicYieldCurveSpecification §1.1		
↔	BasicYieldCurveSpecification §1.1	model	→

⇑:Inherits ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.4 BasicYieldCurveSpecificationModel

A concrete model of the BasicYieldCurveSpecification §1.1 interface. The various parameters which make up the specification are implemented as attributes and an aggregation of point rates.

3.4.1 Relationships

	Class	Description	Notes
↑	BasicYieldCurveSpecification §1.1		
↔	RateFunctionSpecifier	points	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.4.2 Attributes

currency: Currency The currency of the yield curve.

dateBasis: DateBasis The common date basis for the yield curve.

quotationMethod: InterestRateQuotationMethod The common quotation method for the yield curve.

identifier: String The name of the specification.

3.4.3 Operations

Collection<RateSpecifier> sources() sources

The source rates for this specification.
Return the associated points.

3.5 InterestRateBasisPointsModel

A concrete implementation of the InterestRatePremiumQuotationMethod interface. This class represents an interest rate premium in terms of basis points. Basis points represent $\pm 0.01\%$ difference in a quoted interest rate. The frequency of the basis point addition can vary independently of that of the base interest rate.

3.5.1 Relationships

	Class	Description	Notes
↑	InterestRatePremiumQuotation-Method §1.4		

↑:Realizes

3.5.2 Attributes

frequency: InterestRateCompoundingFrequency The compounding frequency for this premium.

3.5.3 Operations

Number parse(InputStream stream, Boolean loose, LogicalRateSpecifier specifier) parse
stream: InputStream The stream to read the value from.
loose: Boolean Perform loose parsing. The default value is true.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.
Raises: ParseException

Read in a text description of a rate and turn it into an appropriately quoted rate. Read a + or - sign, an integer, p, and a trailing ‘bp’. If loose is true, then the sign and trailing ‘bp’ are optional. Return $p/10000$.

printRate(OutputStream stream, Number rate, Boolean loose, LogicalRateSpecifier specifier) printRate
stream: OutputStream The stream to print onto.
rate: Number The rate to print.

loose: Boolean Print the rate in loose form. The default value is false.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Print a rate piece on an output stream. Print a + or - sign, $rate \times 10000$ rounded to the nearest integer and a trailing 'bp'. Zero has a preceding + sign.

Number addPremium(InterestRateQuotationMethod baseQuote, Number base, Number premium) addPremium
baseQuote: InterestRateQuotationMethod The quotation method for the base interest rate.
base: Number The base interest rate.
premium: Number The premium to add.

Add a premium to an interest rate.

If the baseQuote is not in interest rate form (ie. it is a discount factor) then convert the base amount into the canonical quotation method for an interest rate.

Convert the premium amount into a representation with the same compounding frequency as the baseQuote. Return the sum of the base and premium amounts (suitably converted) in the form of the base quote.

As an example, suppose that the base rate is 10% with a biannual compounding rate. Suppose that this premium is 50 in basis point format, with a continuous compounding rate. Convert the premium into the biannual frequency $2((1 + e^{0.005} - 1)^{\frac{1}{2}} - 1) = 0.005006$. Return the sum 10.5006%.

3.6 InterestRateCompoundingFrequencyModel

An abstract class that implements the InterestRateCompoundingFrequency interface. Subclasses contain the various compounding conventions.

3.6.1 Relationships

	Class	Description	Notes
↑	InterestRateCompoundingFrequency §1.2		
⇓	SimpleCompoundingFrequencyModel §3.11		
⇓	ContinuousCompoundingFrequencyModel §3.7		
⇓	DiscreteCompoundingFrequencyModel §3.8		

⇓:Inherited by ↑:Realizes

3.6.2 Operations

Boolean equal(Comparable arg) equal
arg: Comparable The object to compare against.

Equality test. Return true if the two compounding frequencies are of the same class, false otherwise.

3.7 ContinuousCompoundingFrequencyModel

Interest is compounded continuously.

3.7.1 Relationships

	Class	Description	Notes
⇑	InterestRateCompoundingFrequencyModel §3.6		

⇑:Inherits

3.7.2 Operations

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this quotation frequency into the canonical quotation frequency.

Let t be term in years from the from-date to the to-date, using the specifier's date basis. Let t' be term in years from the from-date to the to-date, using an Actual/Actual date basis.

Return $\frac{t}{t'}r$.[2]

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this the canonical quotation frequency into this quotation frequency.

Let t be term in years from the from-date to the to-date, using the specifier's date basis. Let t' be term in years from the from-date to the to-date, using an Actual/Actual date basis. Return $\frac{t'}{t}r$.[2]

Number discountToYield(Number r, LogicalRateSpecifier specifier) discountToYield
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a discount rate to a yield.

Return r

Derivation: $y = d$ since, $\lim_{n\to\infty}(1+y/n)^n(1-d/n)^n = 1$ from $PV(1+y) = FV$ and $FV(1-d) = PV$.

Number yieldToDiscount(Number r, LogicalRateSpecifier specifier) yieldToDiscount
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield to a discount rate.

Return r

Derivation: $d = y$ since, $\lim_{n\to\infty}(1+y/n)^n(1-d/n)^n = 1$ from $PV(1+y) = FV$ and $FV(1-d) = PV$.

3.8 DiscreteCompoundingFrequencyModel

This class models rates which compound over some discrete compounding rate, eg. annually, bi-annually or quarterly. Two subclasses implement the behavior in slightly different ways, so as to allow level or actual period calculations.

3.8.1 Relationships

	Class	Description	Notes
⇑	InterestRateCompoundingFrequency-Model §3.6		
⇓	DiscreteLevelCompoundingFrequen-cyModel §3.10		
⇓	DiscreteActualCompoundingFrequen-cyModel §3.9		

⇑:Inherits ⇓:Inherited by

3.9 DiscreteActualCompoundingFrequencyModel

A compounding frequency where the interest is compounded on the basis of an actual sequence of days. The associated RepeatedPeriod gives the mechanism for calculating the interest rate. This frequency is the most general frequency and should be used with care.

In the various operation definitions, the following terms are used:

Let $\{d_1, \ldots, d_n\}$ be the sequence of payment dates generated by using period to generate dates by adding to the from-date supplied by the quotation method's specifier. The sequence is terminated by the last date at or beyond the to-date. Let $\{t_1 = d_2 - d_1, \ldots, t_{n-1} = d_n - d_{n-1}\}$ be the series of terms in years, derived from the specifier's date basis.

Where the repeated period is too short, the repeated period is applied again, continuing on from the final date of the repeated period until a complete set of dates is constructed. Let the sequence of total repeated period dates be $\{p_1, \ldots, p_m\}$ where each p_i is the start date of the period. Let $\{y_1 = p_2 - p_1, \ldots, y_{m-1} = pm - p_{m-1}\}$ be the series of terms in years, derived from the specifier's date basis. Let y'_i be the total period for the term t_i; i.e. the y_j for the $p_j \to p_{j+1}$ into which d_i falls. Further, let $f_i = t_i/y'_i$, the fraction of the interest rate period.

Finally, set $t_t = d_t - d_{n-1}$ where d_t is the to-date and t_t is the term in years, calculated from the specifier's date basis.

Let t be the term in years from the from-date to the to-date, derived from the specifier's date basis. Let t' be the term in years from the from-date to the to-date, derived from an Actual/Actual date basis.

As an common example, the repeated period defined by 3 units of 6 months rolled forward, a date basis of Actual/360, a from-date of 2-Jan-1999 and a to-date of 8-Jan-2002. The d_i dates are: { 2-Jan-1999, 2-Jul-1999, 3-Jan-2000, 2-Jul-2000, 2-Jan-2001, 2-Jul-2001, 2-Jan-2002, 2-Jul-2002 }. The t_i terms are: { 181/360, 185/360, 181/360, 184/360, 181/360, 184/360, 181/360 }. The p_i dates are: {

2-Jan-1999, 2-Jul-2000, 2-Jan-2002, 2-Jul-2003 }. The y_i terms are: { 547/360, 549/360, 546/360 } and the f_i terms are { 181/547, 185/547, 181/547, 184/549, 181/549, 184/549, 181/546 }, $t_t = 6/360$

The Actual/Actual term, $t' = 364/365 + 366/366 + 365/365 + 8/365 = 1102/365$

3.9.1 Relationships

	Class	Description	Notes
⇑	DiscreteCompoundingFrequency-Model §3.8		
↔	RepeatedPeriod	period	→

⇑:Inherits ↔:Association →:Navigable ◊:Aggregate ♦:Composite

3.9.2 Operations

Boolean equal(Comparable arg) equal

arg: Comparable The object to compare against.

Equality test. Return true if the two compounding frequencies are of the same class and are associated to equal periods, false otherwise.

Number discountToYield(Number r, LogicalRateSpecifier specifier) discount-ToYield

r: Number The rate (as a yield) to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a discount rate to a yield.

Return the solution to y for the equation

$$\left((1+yf_{n-1})^{\frac{t_t}{t_{n-1}}}\Pi_{i=1}^{n-2}(1+yf_i)\right)\times\left((1-rf_{n-1})^{\frac{t_t}{t_{n-1}}}\Pi_{i=1}^{n-2}(1-rf_i)\right)=1$$

Derivation: This represents the application of a compounding series of interest payments, with the final payment calculated over a fractional period.

Using the common example, if the discount rate is 10% then the equivalent yield is calculated by:

$$(1+y181/546)^{6/181}$$
$$(1+y181/547)(1+y185/547)$$
$$(1+y181/547)(1+y184/549)$$

$$(1 + y181/594)(1 + y184/549)$$

$$\times$$

$$(1 - 0.1181/546)^{6/181}$$

$$(1 - 0.1181/547)(1 - 0.,1185/547)$$

$$(1 - 0.1181/547)(1 - 0.1184/549)$$

$$(1 - 0.1181/594)(1 - 0.1184/549)$$

$$= 1$$

Giving $y = 10.34\%$.

Number yieldToDiscount(Number r, LogicalRateSpecifier specifier) yieldToDiscount
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield to a discount rate.

Return the solution to d for the equation

$$\left((1 + rf_{n-1})^{\frac{t_t}{t_{n-1}}} \Pi_{i=1}^{n-2}(1 + rf_i)\right) \times \left((1 - df_{n-1})^{\frac{t_t}{t_{n-1}}} \Pi_{i=1}^{n-2}(1 - df_i)\right) = 1$$

Derivation: This represents the application of a compounding series of interest payments, with the final payment calculated over a fractional period.

Using the common example, if the yield is 10% then the equivalent discount rate is calculated by:

$$(1 + 0.1181/546)^{6/181}$$

$$(1 + 0.1181/547)(1 + 0.1185/547)$$

$$(1 + 0.1181/547)(1 + 0.1184/549)$$

$$(1 + 0.1181/594)(1 + 0.1184/549)$$

$$\times$$

$$(1 - d181/546)^{6/181}$$

$$(1 - d181/547)(1 - d185/547)$$

$$(1 - d181/547)(1 - d184/549)$$

$$(1 - d181/594)(1 - d184/549)$$

$$= 1$$

Giving $d = 9.68\%$.

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical

r: Number The rate (as a yield) to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this quotation frequency into the canonical quotation frequency.

Return

$$\frac{1}{t'} \ln \left((1 + r f_{n-1})^{\frac{t_t}{t_{n-1}}} \Pi_{i=1}^{n-2} (1 + r f_i) \right)$$

Derivation: This represents the application of a compounding series of interest payments, with the final payment calculated over a fractional period. The result is set equal to $e^{\mu t'}$

Using the common example, if the yield is 10% then the equivalent canonical yield is calculated by:

$$i = \frac{365}{1102} \ln \begin{pmatrix} (1 + 0.1181/546)^{6/181} \\ (1 + 0.1181/547)(1 + 0.,1185/547) \\ (1 + 0.1181/547)(1 + 0.1184/549) \\ (1 + 0.1181/549)(1 + 0.1184/549) \end{pmatrix}$$

Giving $i = 6.55\%$ Note that, since the example period runs over about 1.5 years, the quoted yield is about $10\%/1.5$.

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical

r: Number The rate (as a yield) to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this the canonical quotation frequency into this quotation frequency.

Return the solution to y for

$$(1 + y f_{n-1})^{\frac{t_t}{t_{n-1}}} \Pi_{i=1}^{n-2} (1 + y f_i) = e^{rt'}$$

Derivation: This represents the application of a compounding series of interest payments, with the final payment calculated over a fractional period. The result is set equal to $e^{\mu t'}$

Using the common example, if the canonical yield is 10% then the equivalent yield is calculated by:

$$\begin{pmatrix} (1+y181/546)^{6/181} \\ (1+y181/547)(1+y185/547) \\ (1+y181/547)(1+y184/549) \\ (1+y181/549)(1+y184/549) \end{pmatrix} = e^{0.1\frac{1102}{365}}$$

Giving $y = 15.40\%$. Note that, since the example period runs over about 1.5 years, the quoted yield is about $1.5 \times 10\%$.

3.10 DiscreteLevelCompoundingFrequencyModel

A variant of interest rate compounding where the compounding frequency is expressed in terms of a number of divisions within a year.

3.10.1 Relationships

	Class	Description	Notes
⇑	DiscreteCompoundingFrequency-Model §3.8		

⇑:Inherits

3.10.2 Attributes

frequency: Number = 1 The number of times a year that interest is paid, leading to a compounding frequency. Values of 1, 2, 4, 6 or 12 are common. However, other values, fractional values and values less than 1 are all possible.

3.10.3 Operations

Boolean equal(Comparable arg) equal
arg: Comparable The object to compare against.

Equality test. Return true if the two compounding frequencies are of the same class and have the same frequency attributes, false otherwise.

Number discountToYield(Number r, LogicalRateSpecifier specifier) discount-ToYield
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a discount rate to a yield.

Let n be the level payment frequency. Return $\frac{r}{1-\frac{r}{n}}$.

Derivation: $(1+y/n)^n(1-d/n)^n = 1$ from $PV(1+y) = FV$ and $FV(1-d) = PV$.

Number yieldToDiscount(Number r, LogicalRateSpecifier specifier) yieldToDiscount

r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield to a discount rate.

Let n be the level payment frequency. Return $\frac{r}{1+\frac{r}{n}}$.

Derivation: $y = d$ since, $(1+y/n)^n(1-d/n)^n = 1$ from $PV(1+y) = FV$ and $FV(1-d) = PV$.

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical

r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this quotation frequency into the canonical quotation frequency.

Let t be term in years from the from-date to the to-date, using the specifier's date basis. Let t' be term in years from the from-date to the to-date, using an Actual/Actual date basis. Return $\frac{nt}{t'}\ln(1+\frac{r}{n})$, where n is the frequency.

Derivation: $e^{\mu t'} = (1+\frac{r}{n})^{nt}$

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical

r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this the canonical quotation frequency into this quotation frequency.

Let t be term in years from the from-date to the to-date, using the specifier's date basis. Let t' be term in years from the from-date to the to-date, using an Actual/Actual date basis. Return $n(e^{\frac{rt'}{nt}}-1)$, where n is the frequency.

Derivation: $e^{\mu t'} = (1+\frac{r}{n})^{nt}$

3.11 SimpleCompoundingFrequencyModel

Interest is calculated at a simple rate.

3.11.1 Relationships

	Class	Description	Notes
⇑	InterestRateCompoundingFrequency-Model §3.6		

⇑:Inherits

3.11.2 Operations

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this quotation frequency into the canonical quotation frequency.

Let t be the term in years from the from-date to the to-date of the rate specifier of the quotation, using the specifier's date basis. Let t' be the term in years from the from-date to the to-date using an Actual/Actual date basis. Return $\frac{1}{t'}\ln(1+rt)$

Derivation: $e^{\mu t'} = (1+rt)$.[1]

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield using this the canonical quotation frequency into this quotation frequency.

Let t be the term in years from the from-date to the to-date of the rate specifier of the quotation, using the specifiers's date basis. Let t' be the term in years from the from-date to the to-date using an Actual/Actual date basis. Return $\frac{e^{rt'}-1}{t}$

Derivation: $e)^{rt'} = (1+it)$.[1]

Number discountToYield(Number r, LogicalRateSpecifier specifier) discount-ToYield
r: Number The rate (as a yield) to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a discount rate to a yield.

Let t be the term in years, derived from the quotation method's rate specifier, from and to dates and date basis. Return $\frac{r}{1-rt}$

Derivation: $(1+yt)(1-dt) = 1$ from $PV(1+yt) = FV$ and $FV(1-dt) = PV$.

Number yieldToDiscount(Number r, LogicalRateSpecifier specifier) yieldToDiscount

r: Number The rate (as a yield) to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a yield to a discount rate.

Let t be the term in years, derived from the quotation method's rate specifier, from and to dates and date basis. Return $\frac{r}{1+rt}$

Derivation: $(1+yt)(1-dt) = 1$ from $PV(1+yt) = FV$ and $FV(1-dt) = PV$.

3.12 InterestRatePieceModel

A concrete implementation of the InterestRatePiece interface. Quotation methods used by this subclass must implement the InterestRateQuotationMethod §1.5 interface.

3.12.1 Relationships

	Class	Description	Notes
⇑	RatePieceModel		
↑	InterestRatePiece §1.3		

⇑:Inherits ↑:Realizes

3.13 InterestRateQuotationMethodModel

A concrete implementation of the InterestRateQuotationMethod interface. This is an abstract class. Subclasses encode the various possibilities available.

3.13.1 Relationships

	Class	Description	Notes
↑	InterestRateQuotationMethod §1.5		
⇓	InterestRateAbstractYieldModel §3.15		
⇓	DiscountFactorQuotationMethod-Model §3.14		

⇓:Inherited by ↑:Realizes

3.13.2 Operations

Boolean isMargin() isMargin

Is this rate in margin form? Return false.

3.14 DiscountFactorQuotationMethodModel

A discount factor simply represents the ratio of two amounts at different times. The discount factor is expressed in terms of the ratio of the amount at the from date, a_1 to the amount at the to date a_2 (i.e a_1/a_2). This is usually expressed as a simple number less than 1. For display purposes, discount factors are usually displayed with 4 decimals of accuracy.

3.14.1 Relationships

	Class	Description	Notes
⇑	InterestRateQuotationMethod-Model §3.13		

⇑:Inherits

3.14.2 Operations

Number parse(InputStream stream, Boolean loose, LogicalRateSpecifier specifier) parse

stream: InputStream The stream to read the value from.
loose: Boolean Perform loose parsing. The default value is true.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.
Raises: ParseException

Read in a text description of a rate and turn it into an appropriately quoted rate. If loose is true, then read an arbitrary real number from the stream. If loose is false, then read a real number from the stream with the following format: `#.0000` with any number of digits before the decimal point.

printRate(OutputStream stream, Number rate, Boolean loose, LogicalRateSpecifier specifier) printRate
stream: OutputStream The stream to print onto.
rate: Number The rate to print.
loose: Boolean Print the rate in loose form. The default value is false.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Print a rate piece on an output stream. If loose is true then print rate in the most accurate formal for the class of rate. If loose is false, then print rate using the `#.0000` format.

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical
r: Number The rate to convert into canonical form.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate in this quotation format to the canonical quotation format.

Let t' be the term in years between the from date and to-date using an Actual/Actual date basis Return

$$-\frac{1}{t'} \ln r$$

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical
r: Number The rate to convert into canonical form.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate from the canonical quotation format to this quotation format.

Let t' be the term in years between the from date and to date according to the specifier's from and to dates and an Actual/Actual basis. Return

$$e^{-rt'}$$

Number discountFactor(Number r, LogicalRateSpecifier specifier) discountFactor
r: Number The rate to convert into a discount factor.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate in this quotation format into a discount factor. Return r.

InterestRateCompoundingFrequency frequency() frequency

The compounding frequency of this interest rate. Return a continuous compounding frequency.[1]

Boolean isInterestRate() isInterestRate

Is this an interest rate? Return false.

3.15 InterestRateAbstractYieldModel

An abstract model for yield-like quotation methods. Yield-like quotation methods quote as an interest or discount rate with some sort of payment frequency. Although always treated, externally, as percentages, the internal representation of a yield-like rate is as an ordinary number; 0.07 rather than 7 for 7%.

3.15.1 Relationships

	Class	Description	Notes
⇑	InterestRateQuotationMethodModel §3.13		
⇓	YieldQuotationMethodModel §3.18		
⇓	DiscountRateQuotationMethodModel §3.16		

⇑:Inherits ⇓:Inherited by

3.15.2 Attributes

frequency: InterestRateCompoundingFrequency = InterestRateCompoundingFrequency.canonical()

The compounding frequency to use.

3.15.3 Operations

Number parse(InputStream stream, Boolean loose, LogicalRateSpecifier specifier) parse

stream: InputStream The stream to read the value from.

loose: Boolean Perform loose parsing. The default value is true.

[1] The justification for this compounding frequency is that discount factors are often regarded as interpolating using exponential interpolation in yield curves.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.
Raises: ParseException

Read in a text description of a rate and turn it into an appropriately quoted rate. Yield-like rates are assumed to be percentages. If loose is true, then read an arbitrary real number, r, from the stream, with an optional trailing `%` symbol. If loose is false, then read a real number from the stream with the following format: `[+-]#.00%` with any number of digits before the decimal point. Return $r/100$.

printRate(OutputStream stream, Number rate, Boolean loose, LogicalRateSpecifier specifier) printRate
stream: OutputStream The stream to print onto.
rate: Number The rate to print.
loose: Boolean Print the rate in loose form. The default value is false.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Print a rate piece on an output stream. Let r be $rate/100$. If loose is true then print r in the most accurate formal for the class of rate. If loose is false, then print r using the `#.00%` format.

Number asCanonical(Number r, LogicalRateSpecifier specifier) asCanonical
r: Number The rate to convert into canonical form.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate in this quotation format to the canonical quotation format. First get the yield equivalent, by setting $y = this.toYield(r)$. Return the value of $frequency.asCanonical(this, r)$.

Number fromCanonical(Number r, LogicalRateSpecifier specifier) fromCanonical
r: Number The rate to convert into canonical form.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate from the canonical quotation format to this quotation format. First get the yield equivalent, by setting $y = frequency.fromCanonical(this, r)$. The return $this.fromYield(this, r)$.

Number asYield(Number r, LogicalRateSpecifier specifier) asYield
r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this

rate.

Convert a rate quoted in this quotation convention into an equivalent yield. Dependent on subclass.

Number fromYield(Number r, LogicalRateSpecifier specifier) fromYield
r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted as a yield into this quotation convention. Dependent on subclass.

InterestRateCompoundingFrequency frequency() frequency

The compounding frequency of this interest rate. Return the frequency attribute.

3.16 DiscountRateQuotationMethodModel

An interest rate quoted as a discount rate; the proportion of the final amount subtracted from the final amount to form the initial amount.

3.16.1 Relationships

	Class	Description	Notes
⇑	InterestRateAbstractYieldModel §3.15		
⇓	HundredMinusDiscountQuotation-Method §3.17		

⇑:Inherits ⇓:Inherited by

3.16.2 Operations

Number asYield(Number r, LogicalRateSpecifier specifier) asYield
r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted in this quotation convention into an equivalent yield. Return $frequency.discountToYield(this, r)$

Number fromYield(Number r, LogicalRateSpecifier specifier) fromYield
r: Number The rate to convert.

specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted as a yield into this quotation convention. Return $frequency.yieldToDiscount(this, r)$.

3.17 HundredMinusDiscountQuotationMethod

A variant of a discount rate where the amount quoted is in terms of a hundred percent minus the actual discount rate.

3.17.1 Relationships

	Class	Description	Notes
⇑	DiscountRateQuotationMethod-Model §3.16		

⇑:Inherits

3.17.2 Operations

Number asYield(Number r, LogicalRateSpecifier specifier) asYield

r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted in this quotation convention into an equivalent yield. Return $frequency.discountToYield(this, 1 - r)$

Number fromYield(Number r, LogicalRateSpecifier specifier) fromYield

r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted as a yield into this quotation convention. Return $1 - frequency.yieldToDiscount(this, r)$.

3.18 YieldQuotationMethodModel

An interest rate quoted as a yield; the proportion of the initial amount added to form the final amount.

3.18.1 Relationships

	Class	Description	Notes
⇑	InterestRateAbstractYieldModel §3.15		
⇓	HundredMinusYieldQuotation-MethodModel §3.19		

⇑:Inherits ⇓:Inherited by

3.19 HundredMinusYieldQuotationMethodModel

A variant of a yield where the amount quoted is in terms of a hundred percent minus the actual yield.

3.19.1 Relationships

	Class	Description	Notes
⇑	YieldQuotationMethodModel §3.18		

⇑:Inherits

3.19.2 Operations

Number asYield(Number r, LogicalRateSpecifier specifier) asYield
r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.

Convert a rate quoted in this quotation convention into an equivalent yield. Return $1 - r$.

Number fromYield(Number r, LogicalRateSpecifier specifier) fromYield
r: Number The rate to convert.
specifier: LogicalRateSpecifier The specifier to use when interpreting this rate.
Convert a rate quoted as a yield into this quotation convention. Return $1 - r$.

3.20 InterestRateQuoteModel

A concrete implementation of the InterestRateQuote interface. Instances of this model are only composed from InterestRatePiece §1.3 components.

3.20.1 Relationships

	Class	Description	Notes
⇑	RateQuoteModel		
↑	InterestRateQuote §1.6		

⇑:Inherits ↑:Realizes

3.21 InterestRateSpecifierModel

A concrete model of an InterestRateSpecifier §1.7. The various parameters are implemented as attributes.

At the moment, only currencies are accepted as suitable commodities. In the future, this may be generalized to more complex commodities.

3.21.1 Relationships

	Class	Description	Notes
↑	InterestRateSpecifier §1.7		
↔	Period	fromPeriod 0..1	→
↔	Period	toPeriod 0..1	→
↔	Location	location 0..1	→
↔	Party	party 0..1	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

3.21.2 Attributes

currency: Currency The currency of the interest rate.

fromDate: Date The start date for the interest rate.

If this specifier has an associated from period, then the start date is calculated by adding the associated from period to the current processing date, with the currency's date classifier. The date classifier for any associated location is not used.

toDate: Date The end date for the interest rate.

If this specifier has an associated from period, then the start date is calculated by adding the associated to period to the current processing date, with the currency's date classifier. The date classifier for any associated location is not used.

dateBasis: DateBasis The date basis for day- and year-count calculations.

3.21.3 Operations

Date fromDate() fromDate

The start date. If the fromDate attribute is nil and there is an associated from date Period, then generate the from date. Return the fromDate attribute.

Date toDate() toDate

The end date. If the toDate attribute is nil and there is an associated to date Period, then generate the to date. Return the toDate attribute.

Period fromPeriod() fromPeriod

The start period. Return the associated from period, if there is one. Otherwise return nil.

Period toPeriod() toPeriod

The end period. Return the associated to period, if there is one. Otherwise return nil.

Location location() location

The location of the interest rate. Return the associated location, if there is one. Otherwise, return nil.

Party party() party

The party to whom a loan is being made. Return the associated party, if there is one. Otherwise, return nil.

4 Services

4.1 BasicYieldCurveConstructorService

An abstract class for providing the basis for a basic yield curve constructor. The basic model assumes that a series of basic interest rates are accumulated and fed into a yield curve constructor. Subclasses provide the requisite construction machinery.

4.1.1 Relationships

	Class	Description	Notes
↑	YieldCurveConstructor §2.1		
⇓	LinearInterestRateCurveConstructor-Component §4.2		
↔	BasicYieldCurveSpecification §1.1	specification	→

⇓:Inherited by ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.1.2 Operations

OrderedCollection<RateSpecifier> sources() sources

The source rates. Return the sources from the associated specification.

Rate construct(OrderedCollection<Rate> sources) construct
sources: OrderedCollection<Rate>
Raises: RateConstructorException

Build the yield curve. Construction of curves is specified by subclasses.

LogicalRateSpecifier result() result

The output rate specification. Return a logical rate specification consisting of an InterestRateSpecifier §1.7 constraining the currency to the currency of the associated specification.

4.2 LinearInterestRateCurveConstructorComponent

Construct a yield curve by linearly interpolating between interest rates and flattening the curve outside the domain of the supplied interest rate points.

This component has been included as an example of yield curve construction. Other construction methodologies are, of course, possible and encouraged.

4.2.1 Relationships

	Class	Description	Notes
⇑	BasicYieldCurveConstructorService §4.1		

⇑:Inherits

4.2.2 Operations

Rate construct(OrderedCollection<Rate> sources) construct
sources: OrderedCollection<Rate>
Raises: RateConstructorException

Build the rate.

1. Set the origin date to be the current processing date. Divide the supplied rates into two groups, those with from-dates on or after the origin date and those with from-dates before the origin date.
2. Sort the rates with from-dates after the origin date into to-date order.
3. Remove any rates that have ≤ 0 days difference in to-dates from the previous date, according to the date basis supplied by the associated specification.
4. Choose the first rate, r_0, and use this as an initial flat estimate for the yield curve, converted to the specification's quotation method and date basis.

Then, for each rate $r_i : i \geq 1$:

1. Use the current yield curve to estimate F_{0i} the discount factor from the origin date to the from-date of r_i.
2. Calculate $DF_i = F_{0i}D_i$, where D_i is the discount factor from the from-date to the to-date of r_i and DF_i, therefore is the discount factor from the origin date to the to-date of r_i.
3. Convert DF_i into c_i using the common quotation method required by the specification.
4. Construct the linear polynomial

$$c_{i-1} + \frac{c_i - c_{i-1}}{t_i - t_{i-1}}(t - t_{i-1})$$

to interpolate between the two points, where t_j is the to-date of the jth rate and $t_2 - t_1$ is the term in years between the two dates, calculated according to the common date basis for this yield curve.

The same process as is outlined above can be used to construct a curve for rates before the origin date, working backwards from the origin date.

An example linear construction is shown in figure 1.

Origin Date: 12-Mar-2001				
From-Date	To-date	Quotation	Date Basis	Rate
12-Mar-2001	13-Mar-2001	simple yield	Actual/Actual	5.5%
13-Mar-2001	14-Mar-2001	simple yield	Actual/Actual	5.6%
14-Mar-2001	14-Apr-2001	simple yield	30/Actual	5.7%
12-Mar-2001	13-Mar-2001	annually compounded yield	30/Actual	6.0%
Date Basis: 30/Actual				
Quotation Method: annually compounded yield				

The rates converted to the common quotation method give

Quoted	Canonical	Converted
5.5%	5.50%	5.65%
5.6%	5.60%	5.76%
5.7%	5.50%	5.85%
6.0%	5.75%	6.00%

These are then converted into the following segments:

From-date	To-date	DF_{of}	DF_{ft}	DF_{ot}	Rate	Polynomial
12-Mar-2001	13-Mar-2001	1.0000	0.9998	0.9998	5.65%	$5.65 + 0 \times (d -$ 12-Mar-2001$)/(1/365)$
13-Mar-2001	14-Mar-2001	0.9998	0.9998	0.9997	5.70%	$5.65 + 0.05 \times (d -$ 13-Mar-2001$)/(1/365)$
14-Mar-2001	14-Apr-2001	0.9997	0.9953	0.9950	5.84%	$5.70 + 0.14 \times (d -$ 14-Mar-2001$)/(30/365)$
14-Mar-2001	14-Mar-2002	0.9997	0.9441	0.9439	6.00%	$5.84 + 0.16 \times (d -$ 14-Apr-2001$)/(330/365)$

Figure 1: Example Linear Interest Rate Curve Construction

5 Associations

Table 1: Interest Rates— Associations

Association / Role	Class	Card.	Notes
fromPeriod			
from period	Period	0..1	→
rate specifier	InterestRateSpecifierModel §3.21	0..n	
toPeriod			
to period	Period	0..1	→
rate specifier	InterestRateSpecifierModel §3.21	0..n	
location			
location	Location	0..1	→
rate specifier	InterestRateSpecifierModel §3.21	0..n	
party			
party	Party	0..1	→
rate specifier	InterestRateSpecifierModel §3.21	0..n	
period			
period	RepeatedPeriod		→
quotation method	DiscreteActualCompoundingFrequencyModel §3.9		
points			
specifiers	RateFunctionSpecifier		→
yield curve	BasicYieldCurveSpecificationModel §3.4		◊
model			
model	BasicYieldCurveSpecification §1.1		→
reference data	BasicYieldCurveReferenceDataModel §3.3	0..1	
specification			
specification	BasicYieldCurveSpecification §1.1		→
constructor	BasicYieldCurveConstructorService §4.1		

→:Navigable ◊:Aggregate ⧫:Composite

5.1 fromPeriod

Role: from period *Navigable* Period, 0..1.
Role: rate specifier InterestRateSpecifierModel, 0..n.

A logical period for the start interest date, if required.

5.2 toPeriod

Role: to period *Navigable* Period, 0..1.
Role: rate specifier InterestRateSpecifierModel, 0..n.
The end period, if this rate specifier specifies a moving end date, based on a period.

5.3 location

Role: location *Navigable* Location, 0..1.
Role: rate specifier InterestRateSpecifierModel, 0..n.
An associated location for the interest rate, for interest rates quoted in non-local currencies.

5.4 party

Role: party *Navigable* Party, 0..1.
Role: rate specifier InterestRateSpecifierModel, 0..n.
The party for whom the interest rate is being quoted.

5.5 period

Role: period *Navigable* RepeatedPeriod.
Role: quotation method DiscreteActualCompoundingFrequencyModel.
The period and frequency over which the interest is calculated.

5.6 points

Role: specifiers *Navigable* RateFunctionSpecifier.
Role: yield curve *Aggregate* BasicYieldCurveSpecificationModel.
The points that make up the curve specification.

5.7 model

Role: model *Navigable* BasicYieldCurveSpecification.
Role: reference data BasicYieldCurveReferenceDataModel, 0..1.
The yield curve specification that acts as the underlying model for the reference data.

5.8 specification

Role: specification *Navigable* BasicYieldCurveSpecification.
Role: constructor BasicYieldCurveConstructorService.
The specification used to build the yield curve.

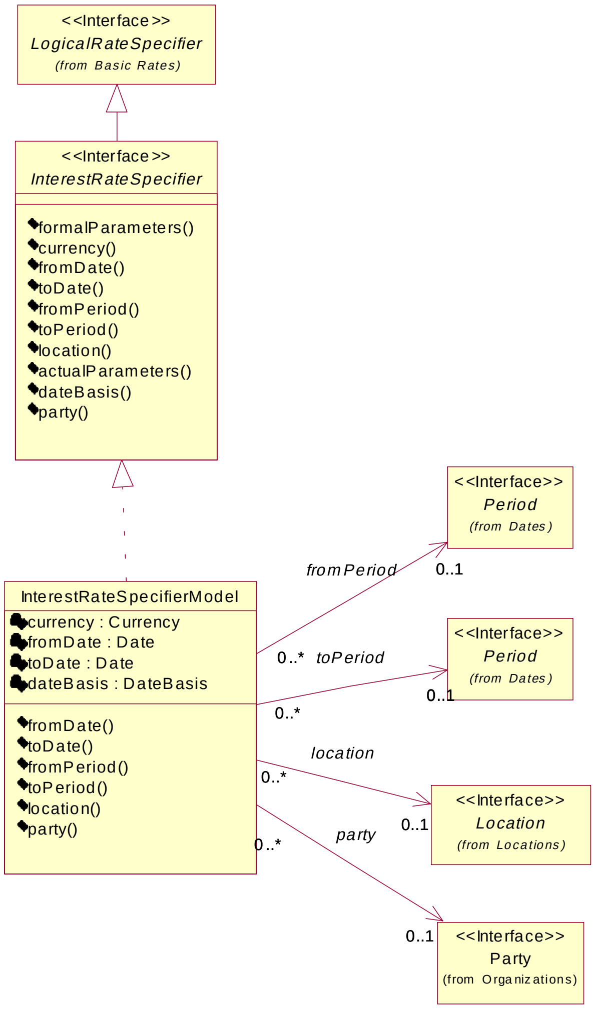

Figure 2: Class Diagram— Rate Specification1

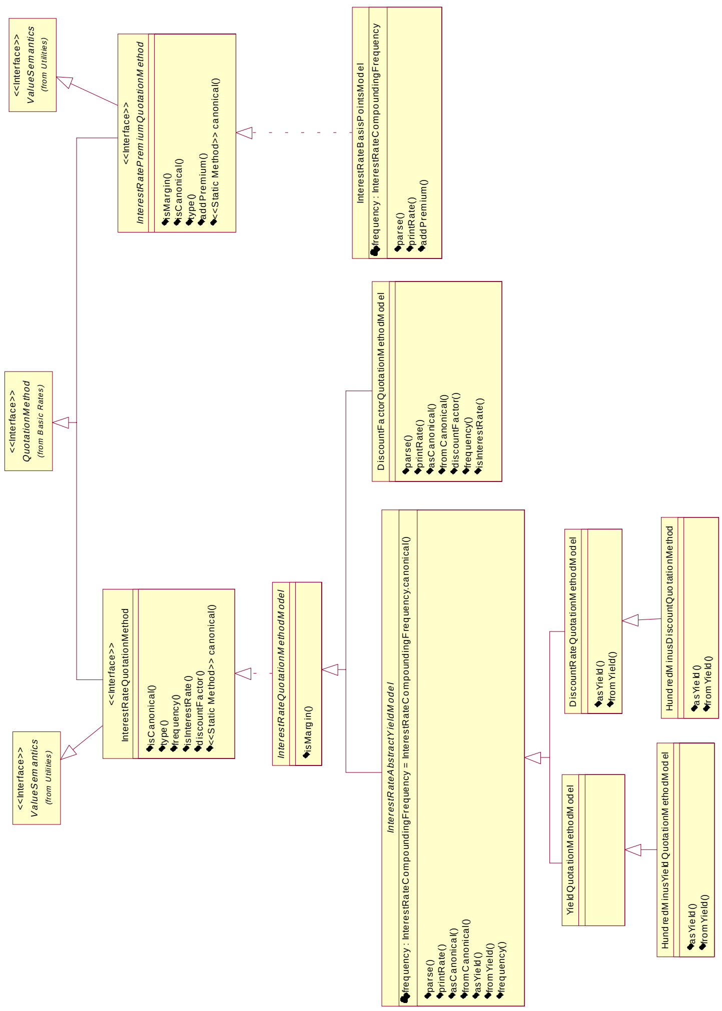

Figure 3: Class Diagram— Rate Specification2

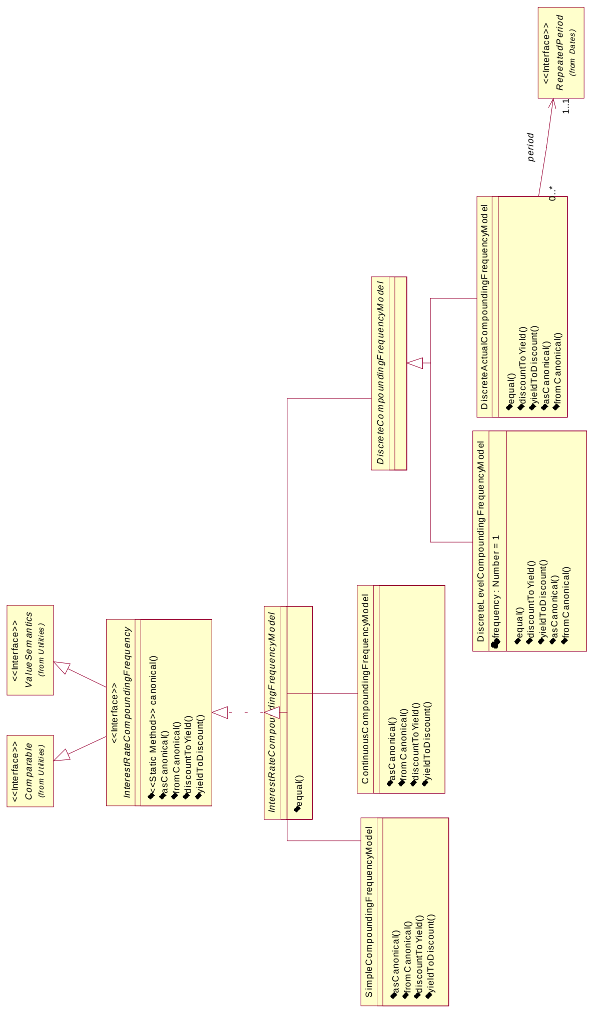

Figure 4: Class Diagram— Rate Specification3

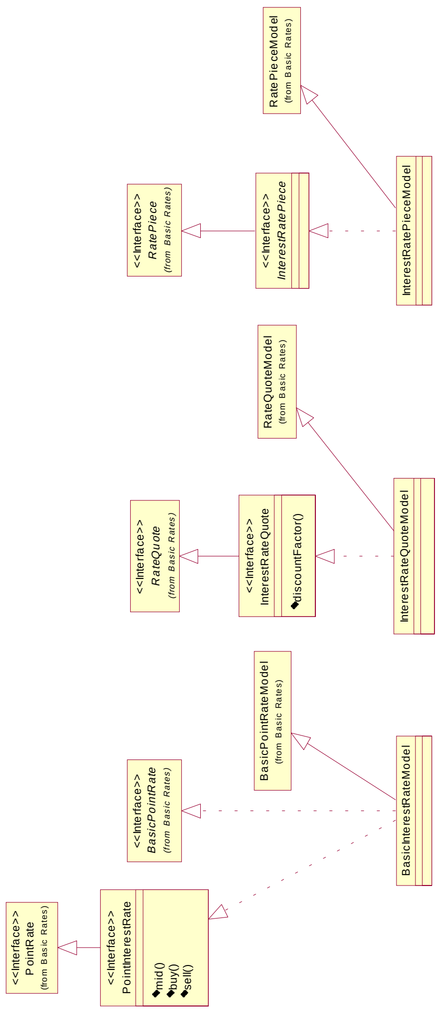

Figure 5: Class Diagram— Point Rates

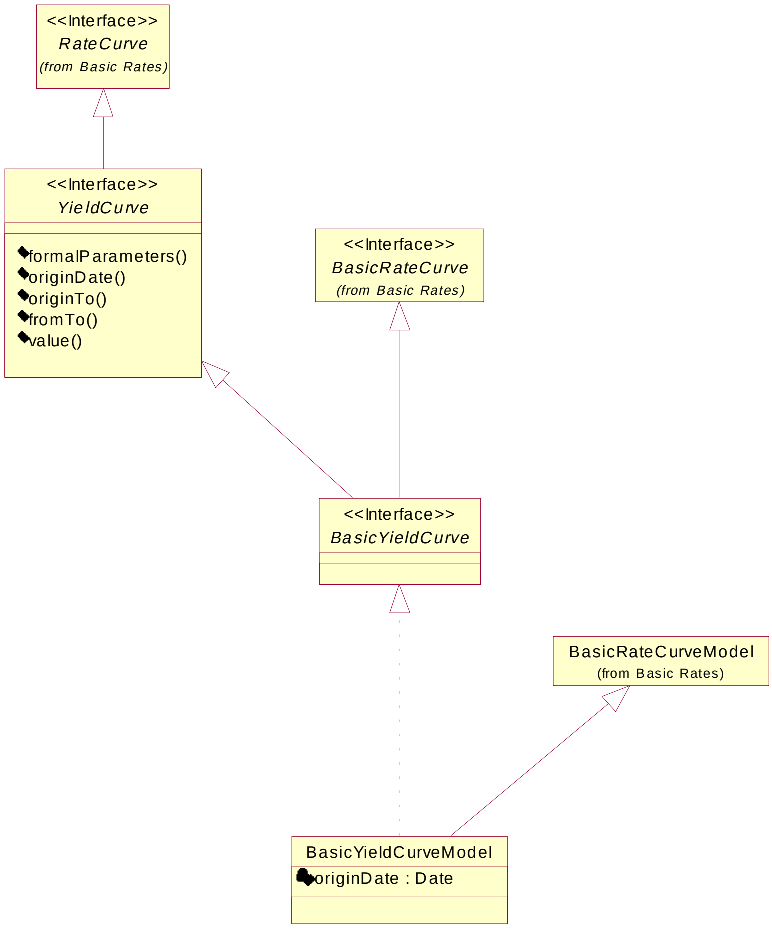

Figure 6: Class Diagram— Yield Curves1

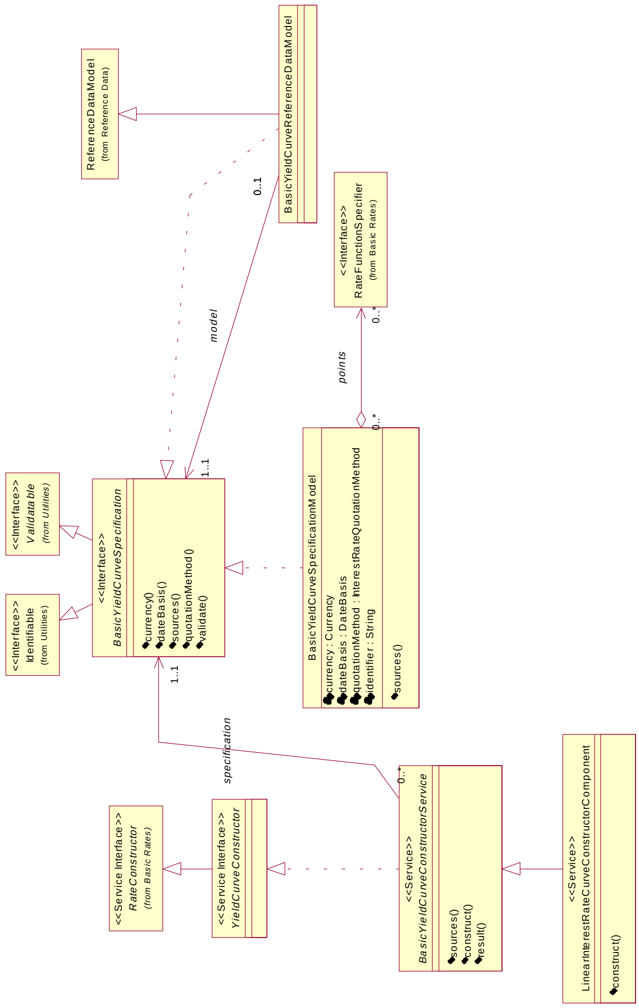

Figure 7: Class Diagram— Yield Curves2

References

[1] Michael Sherris. *Money and Captial Markets*. Allen and Unwin, 1991.

[2] Robert Steiner. *Mastering Financial Calculations*. Pitman Publishing, 1998.