elements

Instruments Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

Contents

List of Figures

List of Tables

Package Description

The Instruments package supplies the ‘instruments’ which are used to record the details of the financial instruments underlying ‘deals’.

The key components of the instruments package are ’transactions’ and ’instruments’.

A transaction specifies the flow of some financial asset. It specifies the direction of the flow, the condition under which the flow will occur and the goods being transferred. The assets being transferred by transactions will be ’instruments’.

An instrument specifies a set of transactions, which can be bought or sold as a group. For instance, a bond would include all its coupon and principal payments as transactions.

Thus, the Instruments Model consists of instruments, which contain transactions, which contain instruments, which contain transactions, and so on. This recursive structure bottoms out with ’atomic’ instruments, such as simple cashflows. For AtomicInstruments to perform this role, they must also be able to function as transactions. That is, they must have the information of a transaction, and must implement the Transaction interface.

The specifier part of the package supplies classes for the specification of floating rate cashflows and of series of such cashflows.

1 Interfaces

1.1 AbstractTransaction

AbstractTransaction is intended to be an abstract superclass of classes implementing the Transaction interface. AbstractTransaction supplies a transaction direction (the buySellMultiplier), and behavior allowing the storage of arbitrary supplementary data.

1.1.1 Relationships

	Class	Description	Notes
⇑	Validatable		
⇓	Transaction §1.2		
↓	AbstractTransactionModel §2.1		

⇑:Inherits ⇓:Inherited by ↓:Realized by

1.1.2 Operations

Integer buySellMultiplier() buySellMultiplier

Enumeration {1, -1}.

A transaction represents the transfer of an instrument to or from a specified organization (the ’counterparty’ of the transaction). The buySellMultiplier indicates whether the counterparty of the transaction is buying or selling the instrument. A value of +1 indicates that the counterparty is buying, and a value of -1 indicates selling.

The description of a deal may involve nesting transactions with different counterparties and buySellMultipliers. For example, consider the sale of a bond issued by organization ’X’ to another organization ’Y’. This would be modeled as a transaction that has the bond as its instrument, Y as its counterparty, and a buySellMultiplier of +1. The bond itself would contain a series of transactions representing coupons. These transaction would have X as their counterparty, and buySellMultipliers of -1.

The use of numbers to indicate the direction of the transaction is a convenience to allow the effect of compounding to be worked out by multiplication.

Ultimately, a transaction will always be between two organizations. However, in general we will not know who is on the other side of a transaction. For instance, the coupon on a bond is paid by the issuer, but the receiver of this coupon is whoever owns the bond. For this reason transactions are modeled as having only one organization.

Object dataFor(String dataKey) dataFor
dataKey: String

Return the supplementary data associated with the specified string. For example, dataFor(’counterparty’) would return the counterparty or null if no counterparty is specified.

Typically, supplementary data would be counterparty information, or information about the accounting type of the transaction.

Null addData(String dataKey, Object data) addData
dataKey: String
data: Object

Add a piece of supplementary data, with the specified key. For example, addData('counterparty', anOrganization) would store the organization 'anOrganization' as the transaction's 'counterparty'.

Typically, supplementary data would be counterparty information, or information about the accounting type of the transaction.

Null removeDataFor(String dataKey) removeDataFor
dataKey: String

Remove data keyed by the parameter.

1.2 Transaction

A transaction is used to represent a one way flow of goods to or from a specified party, on a particular date.

An example of this would be a coupon payment on a bond.

The goods being paid or received may be any kind of instrument, including atomic instruments, such as simple amounts of a commodity, or more complex financial instruments, such as a bond.

1.2.1 Relationships

	Class	Description	Notes
⇑	AbstractTransaction §1.1		
⇓	PriceTransaction §1.7		
⇓	AtomicInstrument §1.3		
↓	TransactionModel §2.5		
↔	PriceTransactionModel §2.6	underlying-Transaction 0..1	
↔	ComplexInstrumentModel §2.9	transactionSequence 0..1	

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

1.2.2 Operations

Condition condition() condition

The condition under which the transaction will take place. This transaction will occur as soon as the condition evaluates to true. Typically this condition will be a DateCondition, explicitly specifying the date of the transaction.

Instrument instrument() instrument

An instrument defines the goods being exchanged.

Transaction multiplyTransaction(Number multiplier) multiplyTransaction
multiplier: Number

Return a new transaction equivalent to the original transaction times a multiplier n. Just multiplies the underlying instrument by this number.

SimpleCashflow valueAtPrice(Price price) valueAtPrice
price: Price

This method takes a price as parameter and returns the value of the transaction: A simpleCashflow. This value is calculated by applying the price to the receiver's underlying instrument, using the 'valueAtPrice' method on the instrument.

1.3 AtomicInstrument

An AtomicInstrument is an elementary financial instrument. That is, it is not composed from other simple instruments. An AtomicInstrument is either a SimpleCashflow or a CashflowSpecifier. It can be composed to form more complex instruments.

1.3.1 Relationships

	Class	Description	Notes
⇑	Instrument §1.11		
⇑	Transaction §1.2		
⇓	SimpleCashflow §1.6		
⇓	CashflowSpecifier §1.4		
⇓	CollateralItem §1.5		

⇑:Inherits ⇓:Inherited by

1.3.2 Operations

Commodity commodity() commodity

The commodity being traded.

Date paymentDate() paymentDate

The actual date on which the cashflow occurs.

1.4 CashflowSpecifier

A CashflowSpecifier defines a single cashflow in terms of a principal and interest rate to be applied to it, thus allowing the amount of the cashflow to be calculated. Cashflows are defined in terms of a fixed rate, floating rate and principal payment.

1.4.1 Relationships

	Class	Description	Notes
⇑	AtomicInstrument §1.3		
↓	CashflowSpecifierModel §2.2		
↔	CashflowSeriesSpecifierModel §2.7	paymentSpecification 0..1	

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

1.4.2 Operations

Date startDate() startDate

Start date of the period for which this cashflow applies. The period from startDate to endDate is the period over which the fixed and floating interest rates apply.

Date endDate() endDate

The end date of the period for which the cashflow applies.

Date fixingDate() fixingDate

The date on which the floating rate is fixed. This date must be before the payment date.

String advanceOrArrears() advanceOrArrears

Indicates whether the payment is made in advance or arrears. If payment is in advance then paymentDate = startDate. If arrears, then paymentDate = endDate.

Number principal() principal

The notional principal used to calculate the cashflow amounts.

PaymentRate fixedInterestRate()

fixedInterestRate

The fixed interest rate. This is applied to the principal to calculate the fixed rate flow.

BasicYieldCurveSpecification floatingRateIndex()

floatingRateIndex

The index (rate curve) used to calculate the floating rate payment. For example, LIBOR, PIBOR, US Treasury etc.

This will be a specifier for the desired rate curve. Typically it will be a standard specifier set up as reference data.

Period floatingRateTenor()

floatingRateTenor

The tenor used to determine the floating rate. For example, if the tenor is 6 months and the floatingRateIndex is LIBOR, the floating rate used will be the 6 month LIBOR rate as at the fixing date.

Note that the floating rate tenor can be different from the term from startDate to endDate, although they will often be the same.

Number principalCashflow()

principalCashflow

The amount of principal payment included in the cashflow. For instance, a loan repayment may include interest and some repayment of principal.

PaymentRate floatingInterestRate()

floatingInterestRate

This is the floating interest rate to be used to calculate the floating rate cashflow. After the fixing date it will be a set value. Before the fixing date it will be read off the floatingRateIndex.

SimpleCashflow floatingRateCashflow()

floatingRateCashflow

The actual cashflow associated with the floating rate. This will be a calculated value.

The floatingRateCashflow is calculated using the spot interest rate from the specified index curve for period floatingRateTenor as at the fixing date. This rate is then converted to an appropriate paymentRate. This rate is then applied to the principal for the period from startDate to endDate by using the interestOnPrincipal method on paymentRate.

SimpleCashflow fixedRateCashflow()

fixedRateCashflow

The cashflow generated by the fixedInterestRate. This will be a calculated value.

The fixedRateCashflow is calculated using the interestOnPrincipal() on the fixedInterestRate, with the receiver's principal as parameter to this method.

1.5 CollateralItem

A collateral item is an asset pledged by a borrower that will be given up if a loan is not paid.

With some trades, for example repos, one party to the deal may feel that there is a significant risk that the other party may default on the deal. In these situations, collateral may be provided as extra insurance to the party with the higher credit rating.

1.5.1 Relationships

	Class	Description	Notes
⇑	AtomicInstrument §1.3		
↓	CollateralItemModel §2.3		

⇑:Inherits ↓:Realized by

1.5.2 Operations

Instrument asset() asset

This is the asset being used as collateral. Bonds or shares are typically assets which are used as collateral. Cash is another asset which can be used as collateral.

CommodityHolding faceValue() faceValue

The value of the asset being used as collateral.

Enumeration specialOrGeneral() specialOrGeneral

Enumeration{'special', 'general'}.

Collateral is identified as either 'special' or 'general'. With some trades, a dealer wants to use specific securities (eg. bonds) only as the collateral. 'special' is used to identify this. If any security can be used to make up the collateral in a trade, then this field is 'general'.

Date startDate() startDate

The date from when the asset is being used as collateral.

Double haircut() haircut

Haircuts are employed to deal with two particular issues concerning repo trades. The first, that the two parties to a repo will be of different credit standing, thereby allowing the more creditworthy party to the repo to borrow a greater value than they lend. The second, that the value of securities pledged as collateral will vary with market prices; in order to provide some cushion against these changes in value; thus reducing the number of margin calls to be made.

Date repaymentDate() repaymentDate

The date on which repayment of collateral occurs.

Price marketPrice() marketPrice

When collateral has been obtained from an internal party, then the price at which the collateral has been obtained needs to be recorded.

CommodityHolding marketValue() marketValue

The value of the collateral at the market price. This will be filled in at a later date when revaluation is to be taken into account.

type() type

lendingFee() lendingFee

1.6 SimpleCashflow

A SimpleCashflow is a financial instrument which can be used to create more complex instruments. This instrument is ‘simple’ in the sense that it cannot be broken down any further. This is the minimum amount of information required to define a meaningful instrument. A SimpleCashflow interface returns a cashflow on a given date. An example of this would be some amount of a currency, on a given date. That is 100 USD on 14 January 1999.

1.6.1 Relationships

	Class	Description	Notes
⇑	AtomicInstrument §1.3		
↓	SimpleCashflowModel §2.4		

⇑:Inherits ↓:Realized by

1.6.2 Operations

Number quantity() quantity

The amount of a commodity.

1.7 PriceTransaction

A PriceTransaction is used to model the situation where a price is paid in exchange for another transaction: The ’underlying transaction’.

The consideration paid may be specified either as a ’price’ or as an instrument to be exchanged for the underlying transaction.

A ’price’ specification would be, for example, a price per hundred face value or yield to maturity: A number that is applied to the underlying transaction to calculate the consideration. In this case the instrument method will return this calculated consideration.

In the case where the consideration is specified explicitly, it will be held in the ’instrument’ of the PriceTransaction.

1.7.1 Relationships

	Class	Description	Notes
⇑	Transaction §1.2		
↓	PriceTransactionModel §2.6		

⇑:Inherits ↓:Realized by

1.7.2 Operations

SecurityPrice price() price

Goods are exchanged for some price. The price the goods are exchanged for is recorded here. The price is applied to the underlying transaction to calculate an instrument to be paid as a consideration for the underlying transaction.

Transaction underlyingTransaction() underlying-Transaction

A priceTransaction is used to model a payment which is being made as the consideration for another transaction. This other transaction is the ’underlying-Transaction’ of the priceTransaction.

1.8 CommodityHolding

CommodityHolding represents a simple holding of some commodity. For instance, 100 US dollars, or 20 ounces of gold. This is an amount of a commodity without any concept of date of payment.

1.8.1 Relationships

	Class	Description	Notes
⇑	ValueSemantics		
↓	CommodityHoldingModel §2.8		

⇑:Inherits ↓:Realized by

1.8.2 Operations

Number quantity() quantity

This is the quantity being held.

Commodity commodity() commodity

This is the commodity being held.

1.9 Condition

A condition specifies a set of requirements. The evaluate method will return true if these requirements are satisfied in the current environment, and false otherwise.

Conditions test time, rates and functions against specified values or other rates and functions. For example, a condition may specify that date = 1 Feb 2001; or that the maximum value of a share price over a given period is less than a specified value.

Conditions are used to specify the circumstances under which a transaction will occur or an option will be available.

In the modeling of loans, bonds and simple instruments we only need to be able to specify date conditions. This is all that has been modeled at this stage. Ultimately more complex types of conditions, for example for use with options, will be required..

1.9.1 Relationships

	Class	Description	Notes
⇓	DateCondition §1.10		

⇓:Inherited by

1.9.2 Operations

Boolean evaluate() evaluate

This method tests whether or not the condition is satisfied.

1.10 DateCondition

DateConditions test whether the current date has a particular value. A DateCondition will be true if the current date equals the condition's date parameter.

1.10.1 Relationships

	Class	Description	Notes
⇑	Condition §1.9		
↓	DateConditionModel §2.11		

⇑:Inherits ↓:Realized by

1.10.2 Operations

Date date() date

This method returns the date for the dateCondition. The dateCondition will be true if the current date equals this specified date.

Boolean evaluate() evaluate

Tests whether or not the current date is equal to the date returned by the date method.

1.11 Instrument

An Instrument is used to record the financial details of deals. The financial details are the underlying cashflows, including information about timing, amount, commodity and the organization paying or receiving the cashflow.

An instrument specifies a collection of transactions which in turn specify the exchange of instruments. This hierarchy of containment bottoms out with atomic instruments, which are ’elementary’, in the sense that they are not composed of simpler instruments.

1.11.1 Relationships

	Class	Description	Notes
⇑	Validatable		
⇓	InstrumentWithMultiplier §1.13		
⇓	InstrumentWithSpecifier §1.14		
⇓	CashflowSeriesSpecifier §1.12		
⇓	AtomicInstrument §1.3		
↓	ComplexInstrumentModel §2.9		
↔	InstrumentWithMultiplierModel §2.12	unitInstrument 0..n	
↔	TransactionModel §2.5	instrument 0..1	
↔	CollateralItemModel §2.3	security	
↔	CollateralItemModel §2.3	asset	

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

1.11.2 Operations

Collection<Transaction> transactionSequence() transactionSequence

This returns a collection of transactions which are the cashflows associated with a deal.

Collection<Transaction> cashflowsAtDate(Date aDate) cashflowsAtDate
aDate: Date

This returns the cashflows in the transactionSequence that occur on and after the given date. It will need to check the conditions on transactions to test any date restrictions.

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number

Return a given multiple of the receiver.

SimpleCashflow valueAtPrice(Price price) valueAtPrice

price: Price

Return the value of the receiver at the specified price.

1.12 CashflowSeriesSpecifier

A cashflow series specifier encodes a series of cashflows. It is capable of generating and returning the cashflows it defines. These cashflows can be defined in terms of fixed interest rates, floating rates etc. The sequence can be of any finite length, or may be infinite (perpetuity).

A CashflowSeriesSpecifier will be used to encode, for example, the repayments on a loan, the dividends on an equity, or the series of coupons for a bond or floating rate note. By specifying an infinite series of cashflows it can be used to model perpetual bonds and equities.

1.12.1 Relationships

	Class	Description	Notes
⇑	Instrument §1.11		
↓	CashflowSeriesSpecifierModel §2.7		
↔	InstrumentWithSpecifierModel §2.10	specifier 0..1	

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

1.12.2 Operations

Party counterparty() counterparty

Who is paying/receiving these cashflows to/from us?

RepeatedPeriod paymentPeriod() paymentPeriod

This is a repeated period defining the frequency of the cashflows. This repeated period is applied to the paymentRate which holds data to generate a series of payments (cashflowSpecifiers).

CashflowSpecifier paymentSpecification() paymentSpecification

This specifies how the cashflows are to be calculated for the cashflowSeries.

This cashflow specifier will have the dates for the first cashflow in the series. The subsequent dates are calculated by applying the repeated period to these initial dates.

Double paymentGrowthRate() paymentGrowthRate

This is the annual rate at which payments grow. It is most relevant for equities.

String amortizationType()

amortization-Type

Are the cashflows based on level principal or level payments? The latter is only meaningful in the case of fixed rate cashflows. It is specifically used for level-repayment loans.

This operation is included as a convenient method of specifying a standard pattern of payments. In general, of course, a cashflow series may not follow either of these forms, in which case this operation will return nil.

The possible return values for this method are ’levelPrincipal’ and ’levelPayment’. These values are of type ’string’. If the development language offers string interning, then this should be used.

Date endDate()

endDate

This is the end date for the cashflowSeries. It is not relevant in the case of perpetuities.

Boolean perpetuity()

perpetuity

Does the cashflow series ever terminate? If so then this method returns false.

Commodity commodity()

commodity

Return the commodity (currency) in which this series of cashflows is denominated.

Collection<Transaction> generateCashflows()

generateCash-flows

This method will return the full series of cashflows specified by the CashflowSpecifier. If the cashflow series is a perpetuity then we will return, say, 10 cashflows. The first 9 cashflows will be the calculated cashflows. The 10th will be a new cashflowSeriesSpecifier, specifying the subsequent transactions.

The cashflows will be CashflowSpecifierModels. These will be generated by starting with the paymentSpecification, and applying the paymentPeriod (which is a repeatedPeriod) to its dates, and applying the paymentGrowthRate to its principal. Note that the DateClassifier used to specify holidays for the period calculations should be taken from the location of the receiver’s commodity
(ie. this.commodity.location.classifier).

Collection<Transaction> generateFirstCashflows(Integer numberOfCashflows)

generateFirst-Cashflows

numberOfCashflows: Integer

This method takes an integer, N, as its parameter, and returns N+1 transactions. These transactions contain the first N cashflows generated by the specifier. The last transaction contains a new CashflowSeriesSpecifier encoding the ’tail’ of the series.

1.13 InstrumentWithMultiplier

An InstrumentWithMultiplier allows an instrument to be specified as a multiple (the ’multiplier’) of another instrument (the ’unitInstrument’).

For example, consider a trade of 1 million of bonds which have a lot size of 1000. This can be represented by an instrumentWithMultiplier with the required bond as its unitInstrument, and a multiplier of 1000.

1.13.1 Relationships

	Class	Description	Notes
⇑	Instrument §1.11		
↓	InstrumentWithMultiplierModel §2.12		

⇑:Inherits ↓:Realized by

1.13.2 Operations

Number multiplier() multiplier

The multiplier is used to identify the number of times the unit instrument will be multiplied by to give the amount of an instrument being traded within a deal.

This can be an integer, fraction or a dollar value depending on the type of instrument being traded.

For example, a bond issue with a lot size of 1000 (being the unitInstrument), will trade for a face value of $1,000,000 when the mulitplier is 1000.

Instrument unitInstrument() unitInstrument

An InstrumentWithMultiplier specifies a financial instrument as a multiple of another instrument. This latter instrument is the ’unitInstrument’ of the InstrumentWithMultiplier.

SimpleCashflow valueAtPrice(Price price) valueAtPrice
price: Price

Calculate the value of the unitInstrument at the specified price, and multiply by the receiver's multiplier.

1.14 InstrumentWithSpecifier

A specifier is designed to be a summary of information about cashflows. There are occasions when all the cashflows for a trade are sufficiently standard and regular to be algorithmically generated. In this case a cashflowSeriesSpecifier can be used to record the information required to generate the cashflows.

1.14.1 Relationships

	Class	Description	Notes
⇑	Instrument §1.11		
↓	InstrumentWithSpecifierModel §2.10		

⇑:Inherits ↓:Realized by

1.14.2 Operations

CashflowSeriesSpecifier specifier() specifier

This contains information 'specifying' a cashflow which can be used to generate a series of cashflows over the life of a deal. For example, a bond deal can have fixed coupon payments associated with it. If the same things differ between each coupon then a specifier can be used to construct the series of cashflows for a particular bond issue.

Boolean useSpecifier() useSpecifier

If the specifier is used to construct the cashflows, then true is returned, otherwise false is returned. This operation is necessary because there are cases when the specifier is merely used to generate a default series of cashflows the details of which are then modified. In this situation we still want to record the specifier, since it records useful information, even though it is not sufficient to fully define the cashflows.

For example, a loan with regular fixed repayments would be modeled as an instrumentWithSpecifier, and the specifier would generate all the cashflows (useSpecifier would be true). If the same loan had an extra one-off principal repayment, then useSpecifier would be false.

Collection defaultTransactionSequence() defaultTransactionSequence

defaultTransactionSequence returns the sequence of transactions defined by the specifier. This will be the result of sending the transactionSequence method to the specifier.

Commodity commodity() commodity

An instrument with specifier will have a single commodity. This method returns the commodity.

Party counterparty() counterparty

Return the counterparty of the instrumentWithSpecifier.

2 Classes

2.1 AbstractTransactionModel

AbstractTransactionModel realizes the AbstractTransaction interface. It is intended to be used as an abstract superclass of classes implementing the Transaction interface. It supplies its subclasses with services required for specifying transaction direction, and named supplementary data.

2.1.1 Relationships

	Class	Description	Notes
↑	AbstractTransaction §1.1		
⇓	SimpleCashflowModel §2.4		
⇓	TransactionModel §2.5		
⇓	CashflowSpecifierModel §2.2		
⇓	CollateralItemModel §2.3		

⇓:Inherited by ↑:Realizes

2.1.2 Attributes

buySellMultiplier: Integer

supplementaryData: Dictionary The supplementaryData attribute holds a dictionary used to store optional supplementary data, keyed by a descriptive string. It will hold such things as counterparty, if relevant.

2.1.3 Operations

Object dataFor(String dataKey) dataFor
dataKey: String

If the supplementaryData is null, then return null. If supplementary data is a dictionary, then return the value keyed by the dataKey, if any. If there is no such key in the dictionary, return null.

Null addData(String dataKey, Object data) addData
dataKey: String
data: Object

If the supplementaryData attribute is a dictionary, then add the data keyed by the dataKey. If the supplementaryData attribute is null, then set it to be a dictionary, and add the data as above.

2.2 CashflowSpecifierModel

CashflowSpecifierModel implements the CashflowSpecifier interface and provides attributes for all relevant data. It allows a single cashflow to be specified.

2.2.1 Relationships

	Class	Description	Notes
⇑	AbstractTransactionModel §2.1		
↑	CashflowSpecifier §1.4		
↔	Period	floatingRateTenor 0..1	→
↔	BasicYieldCurveSpecification	floatingRateIndex 1..1	→
↔	Commodity	commodity 1..1	→

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

2.2.2 Attributes

startDate: Date

endDate: Date

fixingDate: Date

advanceOrArrears: String This attribute records whether the payment is in advance or arrears. This may be somewhat redundant information, since the payment will be in advance if paymentDate = start date and in arrears if payment date = end date.

principal: Number

principalCashflow: Number

fixedInterestRate: PaymentRate

floatingInterestRate: PaymentRate

paymentDate: Date

2.2.3 Operations

Reportable validate()

validate

BasicInterestRate floatingInterestRate()

floatingInterestRate

If the floatingInterestRate attribute has been set, then return its value. If the floating rate has not been fixed, then query the floatingRateIndex for the interest rate applicable from the fixing date for the period floatingRateTenor.

Boolean floatingRateSet()

floatingRateSet

This boolean flag records whether or not the floating rate for this cashflow has been set. This will happen at the rate fixing date.

Instrument instrument()

instrument

CashflowSpecifierModel is both an instrument and a transaction, so the instrument method should just return self.

Condition condition()

condition

This method will return a DateCondition with date equal to the receiver's paymentDate.

CashflowSpecifierModel multiply(Number multiplier)
multiplier: Number

multiply

Return a copy of the receiver with all instance variables the same, except principal and principalPayment, which should be multiplied by the parameter (multiplier).

Instrument multiplyInstrument(Number multiplier)
multiplier: Number

multiplyInstrument

Return an instrument equal to the receiver with all instance variables the same, except principal and principalPayment, which should be multiplied by the parameter (multiplier).

This can be done using the multiply method, and type-casting the result to be of type instrument.

Transaction multiplyTransaction()

multiplyTransaction

Return an instrument equal to the receiver with all instance variables the same, except principal and principalPayment, which should be multiplied by the parameter (multiplier).

This can be done using the multiply method, and type-casting the result to be of type Transaction

Collection transactionSequence()

transactionSequence

Return a collection of transactions containing only the receiver.

Collection cashflowsAtDate(Date date)
date: Date

cashflowsAtDate

If the receiver's paymentDate is the same as or later than the methods date parameter, then return a collection of transactions containing the receiver. Otherwise return an empty collection.

2.3 CollateralItemModel

CollateralItemModel realizes the CollateralItem interface

2.3.1 Relationships

	Class	Description	Notes
$\Uparrow$	AbstractTransactionModel §2.1		
$\uparrow$	CollateralItem §1.5		
$\leftrightarrow$	Instrument §1.11	security	$\rightarrow$
$\leftrightarrow$	Instrument §1.11	asset	$\rightarrow$
$\leftrightarrow$	Price	marketPrice	$\rightarrow$
$\leftrightarrow$	PaymentRate	lendingFee	$\rightarrow$

$\Uparrow$:Inherits $\uparrow$:Realizes $\leftrightarrow$:Association $\rightarrow$:Navigable ◇:Aggregate ◆:Composite

2.3.2 Attributes

specialOrGeneral: Enumeration = general Enumeration{'special', 'general'}.

Collateral is identified as either 'special' or 'general'. With some trades, a dealer wants to use specific securities only as the collateral. 'special' is used to identify this. If any security can be used to make up the collateral against a trade, then this field is 'general'.

startDate: Date

haircut: Double

repaymentDate: Date

type: Enumeration Enumeration{'repo', 'buy-sell back', 'securities lending'}.

This identifies the type of repo being traded. Repos have different characteristics which can be determined by the type of repo.

2.3.3 Operations

PaymentRate lendingFee() lendingFee

Securities lending transactions are where special securities are lent out for a fee, (the lending fee) and collateral is held as security in case of counterparty default.

This lending fee is calculated according to the type of general collateral being held as security.

In the case where the collateral is securities, then let x be the lending fee expressed as basis points per annum,

lending fee $= xb.p. \times$ face value $\times marketvalueofsecurityatend \times$ term

Or, if the collateral being held is cash, then the lending fee can be expressed as an interest rate.

2.4 SimpleCashflowModel

SimpleCashflowModel realizes the SimpleCashflow interface.

2.4.1 Relationships

	Class	Description	Notes
⇑	AbstractTransactionModel §2.1		
↑	SimpleCashflow §1.6		
↔	Commodity	commodity 1..1	→

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

2.4.2 Attributes

quantity: Number The amount of a commodity.

paymentDate: Date The actual date on which the cashflow occurs.

2.4.3 Operations

validate() validate

Instrument instrument() instrument

The receiver is both a transaction and an instrument. Return the receiver.

Collection transactionSequence() transactionSequence

The receiver is both an instrument and a transaction. Return a collection of transactions containing only the receiver.

Condition condition() condition

Return a DateCondition with date equal to the receiver's paymentDate.

SimpleCashflowModel multiply() multiply

Return a copy of the receiver with all instance variables the same, except the quantity, which should be multiplied by the parameter (multiplier).

Transaction multiplyTransaction() multiplyTransaction

Return a transaction equal to the receiver with all instance variables the same, except the quantity, which should be multiplied by the parameter (multiplier).

This can be done using the multiply method, and type-casting the result to be of type Transaction.

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number

Return an instrument equal to the receiver with all instance variables the same, except the quantity, which should be multiplied by the parameter (multiplier).

This can be done using the multiply method, and type-casting the result to be of type instrument.

2.5 TransactionModel

TransactionModel realizes the transaction interface.

A transaction is used to represent a one way flow of goods to or from a specified party, on a particular date.

The goods being transferred will always be an instrument of some kind. There are several types of instrument including ’atomic’ and ’complex’. Atomic instruments are simple flows of a commodity. Complex instruments have multiple cash-flows, and include, for example, bonds.

Examples of transactions include: 1) A coupon payment on a bond would be a transaction with the bond issuer as its counterparty and the coupon payment as its instrument (an atomic instrument).

2) The purchase of a bond would be a transaction with the counterparty being the seller of the bond and the bond as its instrument (a complex instrument).

2.5.1 Relationships

	Class	Description	Notes
$\Uparrow$	AbstractTransactionModel §2.1		
$\uparrow$	Transaction §1.2		
$\Downarrow$	PriceTransactionModel §2.6		
$\leftrightarrow$	Instrument §1.11	instrument 1..1	$\rightarrow$

$\Uparrow$:Inherits $\Downarrow$:Inherited by $\uparrow$:Realizes $\leftrightarrow$:Association $\rightarrow$:Navigable ◊:Aggregate ♦:Composite

2.5.2 Attributes

condition: Condition This condition will specify the circumstances under which the transaction will occur. Typically it will be a DateCondition, specifying the settlement date for the transaction.

2.5.3 Operations

Reportable validate() validate

Transaction multiplyTransaction(Number multiplier) multiplyTransaction
multiplier: Number

Return a transactionModel with instrument equal to the receiver's instrument multipled by the specified number. All other attributes and associations of the receiver are unchanged.

2.6 PriceTransactionModel

This class realizes the PriceTransaction interface. This class is used to record the details of transactions in which a pair of goods are exchanged between parties, at a particular price.

2.6.1 Relationships

	Class	Description	Notes
$\Uparrow$	TransactionModel §2.5		
$\uparrow$	PriceTransaction §1.7		
$\leftrightarrow$	Price	price 0..1	$\rightarrow$
$\leftrightarrow$	Transaction §1.2	underlying-Transaction 1..1	$\rightarrow$

$\Uparrow$:Inherits $\uparrow$:Realizes $\leftrightarrow$:Association $\rightarrow$:Navigable ◊:Aggregate ♦:Composite

2.6.2 Operations

Instrument instrument() instrument

This method will return the instrument paid for the underlying transaction. This will either be calculated by applying a price to the underlying transaction, or will be stored explicitly as the receiver's associated instrument. In the former case the calculated consideration can be cached as the receiver's instrument. In the latter case, the receiver's 'price' will be null. The price is applied to the underlying transaction using the 'valueAtPrice' method.

Transaction multiplyTransaction(Number multiplier) multiplyTransaction
multiplier: Number

Return a TransactionAtPriceModel with instrument equal to the receiver's instrument and priceInstrument multipled by the specified number. All other attributes and associations of the receiver are unchanged.

2.7 CashflowSeriesSpecifierModel

CashflowSeriesSpecifierModel implements the CashflowSeriesSpecifier interface

2.7.1 Relationships

	Class	Description	Notes
$\uparrow$	CashflowSeriesSpecifier §1.12		
$\leftrightarrow$	CashflowSpecifier §1.4	paymentSpecification 1..1	$\rightarrow$
$\leftrightarrow$	RepeatedPeriod	paymentPeriod 1..1	$\rightarrow$
$\leftrightarrow$	Party	counterparty 1..1	$\rightarrow$

$\uparrow$:Realizes $\leftrightarrow$:Association $\rightarrow$:Navigable ◊:Aggregate ⧫:Composite

2.7.2 Attributes

paymentGrowthRate: Number

amortizationType: String

endDate: Date

perpetuity: Boolean

2.7.3 Operations

Reportable validate() validate

Commodity commodity() commodity

Return the commodity on the paymentSpecification.
Return the commodity of the payment specifier.

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number
Return a copy of the receiver with all instance variables the same, except the paymentSpecification which should be set to the original payment specification multiplied by the parameter (multiplier).

Collection<Transaction> transactionSequence() transactionSequence
Return the result of the generateCashflows method.

Collection<Transaction> cashflowsAtDate(Date date) cashflowsAtDate
date: Date
Return the cashflows (generated by the receiver) that occur on or after the specified date.

2.8 CommodityHoldingModel

CommodityHoldingModel realizes the CommodityHoldingModelInterface.

2.8.1 Relationships

	Class	Description	Notes
↑	CommodityHolding §1.8		

↑:Realizes

2.8.2 Attributes

quantity: Number

commodity: Commodity

2.9 ComplexInstrumentModel

A ComplexInstrumentModel is an instrument constructed from a set of specifed transactions. The ComplexInstrumentModel will hold an explicit set of transactions in its transactionSequence.

2.9.1 Relationships

	Class	Description	Notes
↑	Instrument §1.11		
⇓	InstrumentWithSpecifierModel §2.10		
↔	Transaction §1.2	transactionSequence 1..n	→

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

2.9.2 Operations

Reportable validate() validate

The validation would be defined by the business rules, and hence is not documented.

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number

Return a copy of the receiver with all instance variables the same, except the transactionSequence multiplied by the parameter.

Collection<Transaction> transactionSequence() transactionSequence

On ComplexInstrumentModel, the transactionSequence is explicitly stored as an associated collection of Transactions. This method just returns this collection.

Collection<Transaction> cashflowsAtDate(Number multiplier) cashflowsAtDate
multiplier: Number

The cashflowsAtDate() method on complexInstrumentModel will just select those transactions in the transactionSequence which can occur on or after the specified date.

2.10 InstrumentWithSpecifierModel

The InstrumentWithSpecifierModel realizes the InstrumentWithSpecifier interface.

2.10.1 Relationships

	Class	Description	Notes
⇑	ComplexInstrumentModel §2.9		
↑	InstrumentWithSpecifier §1.14		
↔	CashflowSeriesSpecifier §1.12	specifier 1..1	→

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

2.10.2 Attributes

useSpecifer: Boolean

2.10.3 Operations

Commodity commodity() commodity

Return the commodity of the associated specifier.

Party counterparty() counterparty

Return the counterparty of the specifier.

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number

Return a copy of the receiver with all instance variables the same except the transactionSequence and the specifier, which should both be multiplied by the parameter.

Collection<Transaction> transactionSequence() transactionSequence

Return the associated transactionSequence. If the associated specifier is set to ’useSpecifier’ then the transactionSequence will just cache the results of the transactionSequence method on the specifier.

Collection<Transaction> cashflowsAtDate(Number multiplier) cashflowsAtDate
multiplier: Number

Return those transactions in the transactionSequence which can occur on or after the specified date.

2.11 DateConditionModel

2.11.1 Relationships

	Class	Description	Notes
↑	DateCondition §1.10		

↑:Realizes

2.11.2 Attributes

date: Date

2.12 InstrumentWithMultiplierModel

The InstrumentWithMultiplierModel realizes the InstrumentWithMultiplier interface. InstrumentWithMultiplierModel thus also implements the Instrument interface. Each method on an InstrumentWithMultiplier will return the appropriate multiple of the value returned by the same method on the associated unitInstrument.

2.12.1 Relationships

	Class	Description	Notes
↑	InstrumentWithMultiplier §1.13		
↔	Instrument §1.11	unitInstrument 1..1	→

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

2.12.2 Attributes

multiplier: Number The multiplier is used to identify the number of times the unit instrument will be multiplied by to give the amount of an instrument being traded within a deal. This can be an integer, fraction or a dollar value depending on the type of instrument being traded.

2.12.3 Operations

Reportable validate() validate

Instrument multiplyInstrument(Number multiplier) multiplyInstrument
multiplier: Number

Return a copy of the receiver with the same unitInstrument, but with a multiplier multiplied by the parameter.

Collection<Transaction> transactionSequence()

transactionSequence

The transactionSequence method on InstrumentWithMultiplier will return the transactionSequence of the associated unitInstrument, multiplied by the specified multiplier.

Collection<Transaction> cashflowsAtDate()

cashflowsAtDate

The cashflowsAtDate() method on InstrumentWithMultiplier will return the cashflowsAtDate() of the associated unitInstrument, multiplied by the specified multiplier.

3 Associations

Table 1: Instruments— Associations

Association Role	Class	Card.	Notes
specifier			
	CashflowSeriesSpecifier §1.12	1..1	→
	InstrumentWithSpecifierModel §2.10	0..1	
price			
	Price	0..1	→
	PriceTransactionModel §2.6	0..1	
paymentSpecification			
	CashflowSpecifier §1.4	1..1	→
	CashflowSeriesSpecifierModel §2.7	0..1	
paymentPeriod			
	RepeatedPeriod	1..1	→
	CashflowSeriesSpecifierModel §2.7		
floatingRateTenor			
	Period	0..1	→
	CashflowSpecifierModel §2.2	0..n	
floatingRateIndex			
	BasicYieldCurveSpecification	1..1	→
	CashflowSpecifierModel §2.2	0..n	
commodity			
	Commodity	1..1	→
	CashflowSpecifierModel §2.2	0..n	

Table 1: . . . continued

Association Role	Class	Card.	Notes
unitInstrument			
	Instrument §1.11	1..1	→
	InstrumentWithMultiplierModel §2.12	0..n	
commodity			
	Commodity	1..1	→
	SimpleCashflowModel §2.4	0..n	
counterparty			
	Party	1..1	→
	CashflowSeriesSpecifierModel §2.7	0..n	
instrument			
	Instrument §1.11	1..1	→
	TransactionModel §2.5	0..1	
underlyingTransaction			
	Transaction §1.2	1..1	→
	PriceTransactionModel §2.6	0..1	
transactionSequence			
	Transaction §1.2	1..n	→
	ComplexInstrumentModel §2.9	0..1	
security			
	Instrument §1.11		→
	CollateralItemModel §2.3		
asset			
	Instrument §1.11		→
	CollateralItemModel §2.3		
marketPrice			
	Price		→
	CollateralItemModel §2.3		
lendingFee			
	PaymentRate		→
	CollateralItemModel §2.3		

→:Navigable ◊:Aggregate ♦:Composite

3.1 specifier

Role: *Navigable* CashflowSeriesSpecifier, 1..1.
Role: InstrumentWithSpecifierModel, 0..1.

3.2 price

Role: *Navigable* Price, 0..1.
Role: PriceTransactionModel, 0..1.

3.3 paymentSpecification

Role: *Navigable* CashflowSpecifier, 1..1.
Role: CashflowSeriesSpecifierModel, 0..1.

3.4 paymentPeriod

Role: *Navigable* RepeatedPeriod, 1..1.
Role: CashflowSeriesSpecifierModel.

3.5 floatingRateTenor

Role: *Navigable* Period, 0..1.
Role: CashflowSpecifierModel, 0..n.

3.6 floatingRateIndex

Role: *Navigable* BasicYieldCurveSpecification, 1..1.
Role: CashflowSpecifierModel, 0..n.

3.7 commodity

Role: *Navigable* Commodity, 1..1.
Role: CashflowSpecifierModel, 0..n.

3.8 unitInstrument

Role: *Navigable* Instrument, 1..1.
Role: InstrumentWithMultiplierModel, 0..n.

3.9 commodity

Role: *Navigable* Commodity, 1..1.
Role: SimpleCashflowModel, 0..n.

3.10 counterparty

Role: *Navigable* Party, 1..1.
Role: CashflowSeriesSpecifierModel, 0..n.

3.11 instrument

Role: *Navigable* Instrument, 1..1.
Role: TransactionModel, 0..1.

3.12 underlyingTransaction

Role: *Navigable* Transaction, 1..1.
Role: PriceTransactionModel, 0..1.

3.13 transactionSequence

Role: *Navigable* Transaction, 1..n.
Role: ComplexInstrumentModel, 0..1.

3.14 security

Role: *Navigable* Instrument.
Role: CollateralItemModel.

3.15 asset

Role: *Navigable* Instrument.
Role: CollateralItemModel.

3.16 marketPrice

Role: *Navigable* Price.
Role: CollateralItemModel.

3.17 lendingFee

Role: *Navigable* PaymentRate.
Role: CollateralItemModel.

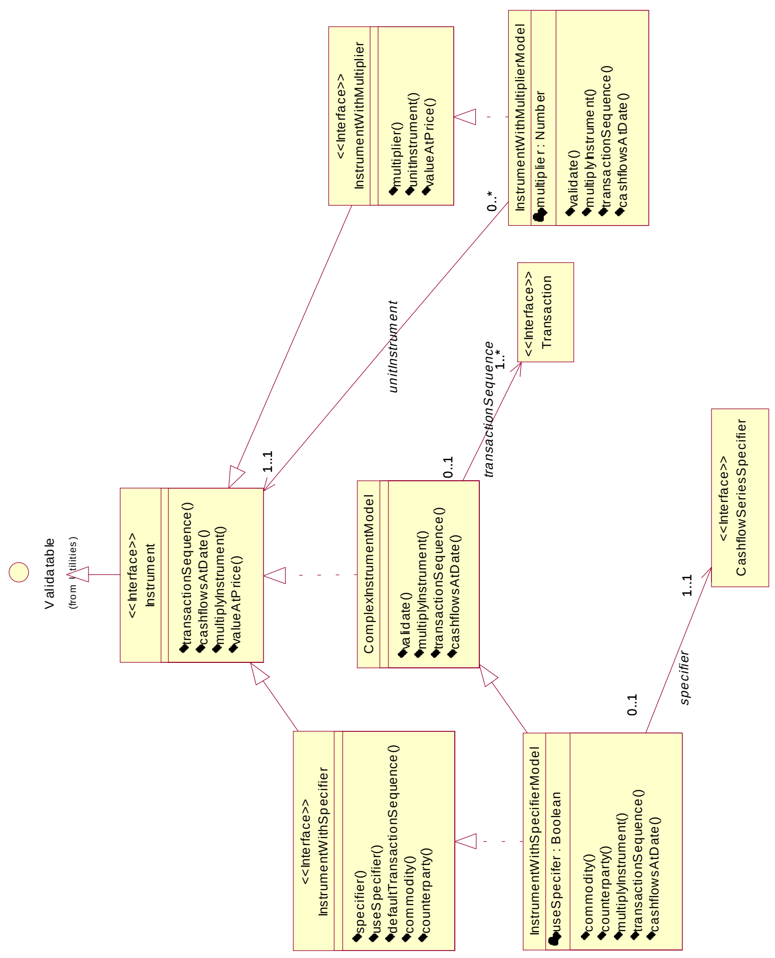

Figure 1: Class Diagram— Instruments

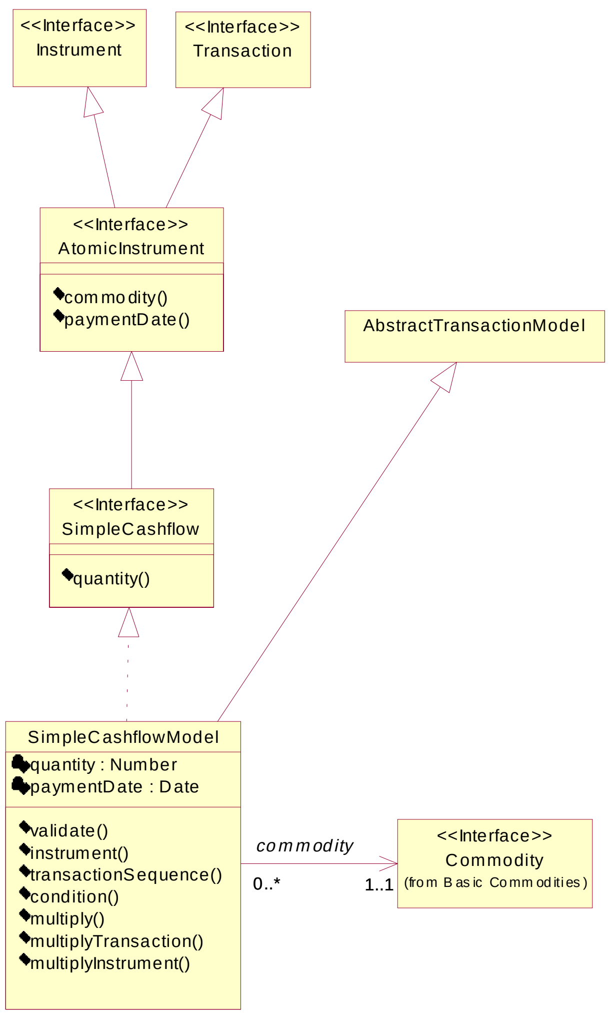

Figure 2: Class Diagram— AtomicInstruments

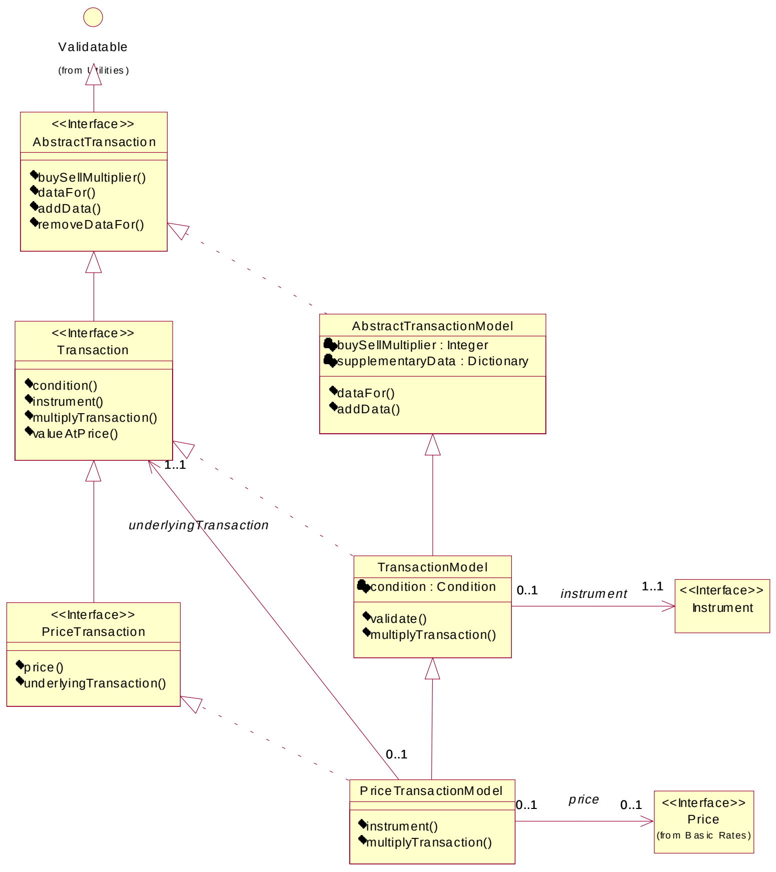

Figure 3: Class Diagram— Transactions

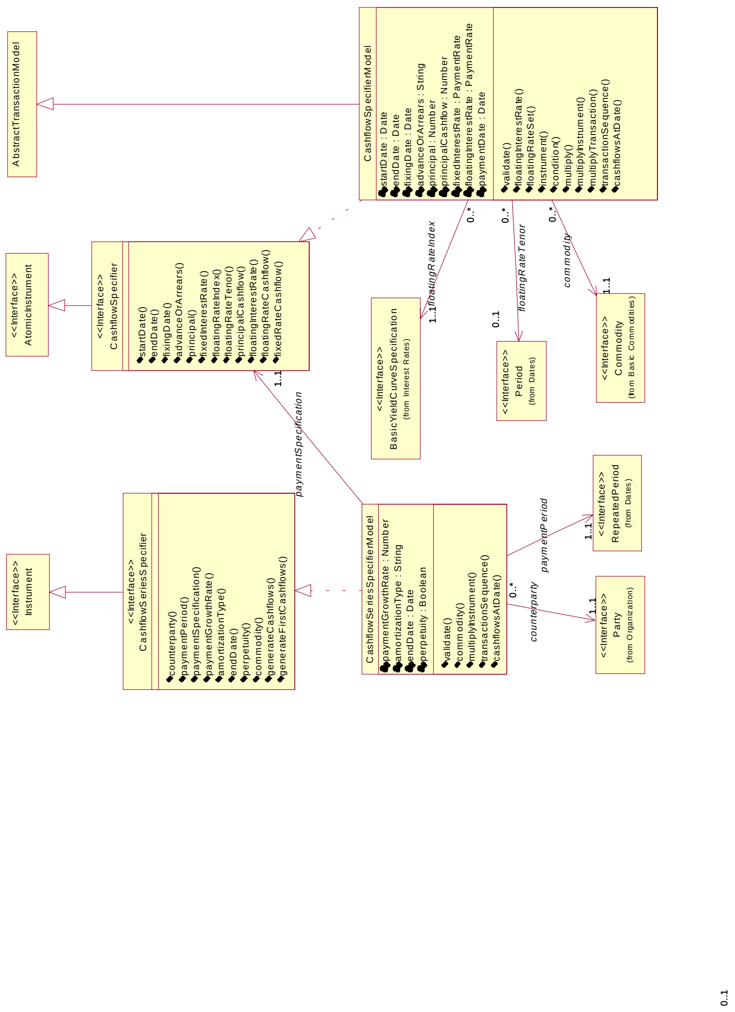

Figure 4: Class Diagram— Specifiers

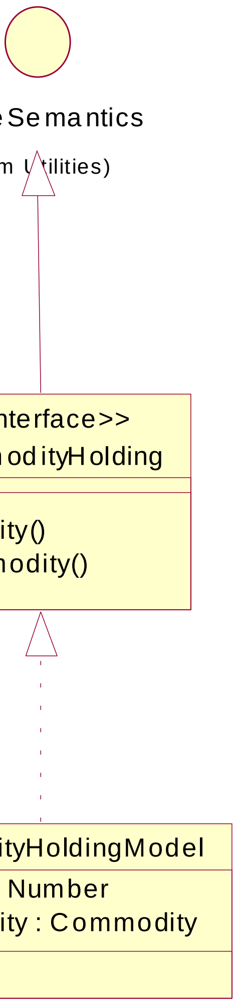

Figure 5: Class Diagram— Holdings

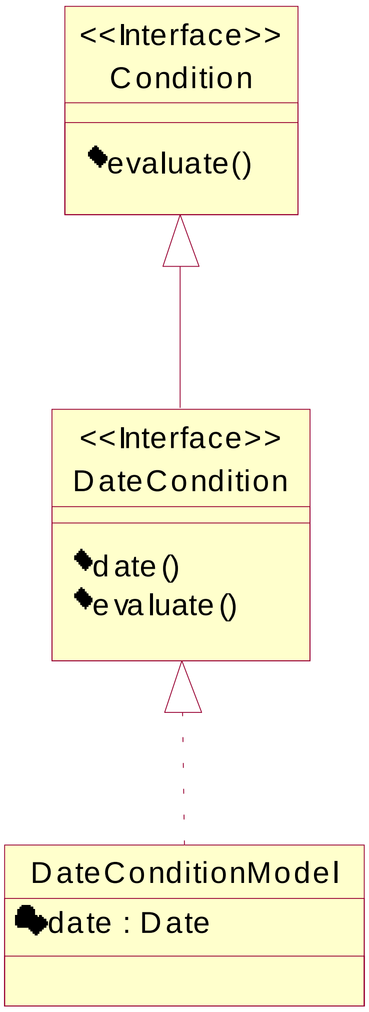

Figure 6: Class Diagram— Conditions

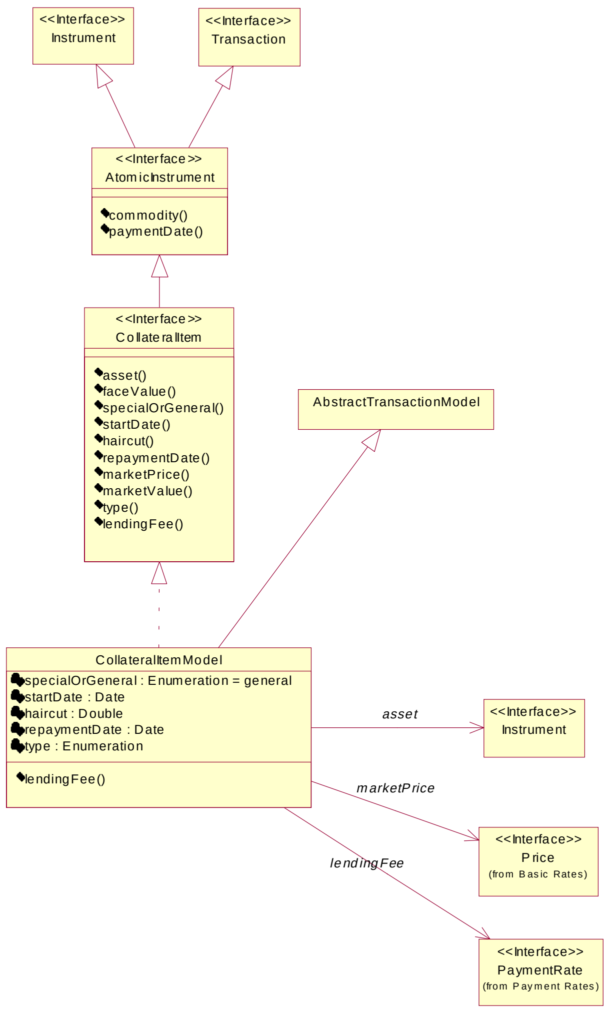

Figure 7: Class Diagram— Collateral

References