AutoExRights Class

佐原伸

2001年5月22日

1　責任

自動権利処理・自動権利行使／自動権利放棄の要件定義。

2　概要

インザマネーの時権利行使し、アオウトオブマネーの時権利放棄する。ただし、実際には受渡代金を考慮して、受渡代金が正の場合のみ権利行使する。

```
class AutoExRights
types
        public ExRightsStatus = AtTheMoney | InTheMoney | OutOfTheMoney;
        public CallPut = Call | Put;
        public Money2 = ℚ
        inv money ≜ isCorrectDecimalPlace (money, 2);
        public Money = ℕ;
        OptionRec :: key : char*
                     dt : Date
                     amount : ℕ
                     commission : ℕ
                     tax : ℕ
functions
public
        isCorrectDecimalPlace : ℚ × ℕ → 𝔹
        isCorrectDecimalPlace (x, y) ≜
          let f = λ x : ℚ · x × 10 ↑ y in
          f (x) = floor f (x);
public
        getExRightsStatus : CallPut × Money2 × Money
                                →
                            ExRightsStatus
        getExRightsStatus (callPut, SQ, ExRightsPrice) ≜
          let x = SQ − ExRightsPrice,
              y = SQ > ExRightsPrice in
          cases mk- (callPut, x, y) :
            mk- (-, 0, -) → AtTheMoney,
            mk- (Call, -, true), mk- (Put, -, false) → InTheMoney,
            mk- (Call, -, false), mk- (Put, -, true) → OutOfTheMoney
          end;
public
```

$test : () \to ExRightsStatus \times ExRightsStatus \times ExRightsStatus \times ExRightsStatus \times ExRightsStatus$

$test\ () \triangleq$

mk- $(getExRightsStatus\ (\text{CALL}, 10, 10),$
$getExRightsStatus\ (\text{CALL}, 10.1, 10),$
$getExRightsStatus\ (\text{CALL}, 10.1, 11),$
$getExRightsStatus\ (\text{PUT}, 10.1, 10),$
$getExRightsStatus\ (\text{PUT}, 10.1, 11));$

public

$moneyBlc : Money2 \times Money \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to Money$

$moneyBlc\ (SQ, ExRightsPrice, amount, commission, tax) \triangleq$

floor (abs $(SQ - ExRightsPrice) \times amount - commission - tax$)

end *AutoExRights*

Test Suite : vdm.tc

Name	**#Calls**	**Coverage**
AutoExRights‘getExRightsStatus	5	$\surd$
AutoExRights‘isCorrectDecimalPlace	6	$\surd$
AutoExRights‘moneyBlc	1	$\surd$
AutoExRights‘test	1	$\surd$
Total Coverage		**100%**

class $Date$is subclass of $Object$

values

$jdMinusMjd = 2400000.5;$

private

$asString = \mathsf{new} Integer().asString;$

private

$STR = \mathsf{new} String();$

$DayNameList = [\textsc{Mon}, \textsc{Tue}, \textsc{Wed}, \textsc{Thr}, \textsc{Fri}, \textsc{Sat}, \textsc{Sun}]$

types

public $Time = \mathbb{R};$

public $DateTime = \mathbb{R};$

public $TermLength = \mathbb{R};$

public $DayName = \textsc{Mon} \mid \textsc{Tue} \mid \textsc{Wed} \mid \textsc{Thr} \mid \textsc{Fri} \mid \textsc{Sat} \mid \textsc{Sun};$

public $DayNumber = \mathbb{N}$

inv $d \triangleq 0 \leq d \wedge d \leq 6$

instance variables

private $modifiedJulianDate : \mathbb{R} := 0;$

functions

public

$today : () \rightarrow Date$

$today\,() \triangleq$

$\quad fromDate\,(2001, 3, 1);$

public

$lt2 : Date \times Date \rightarrow \mathbb{B}$

$lt2\,(d1, d2) \triangleq$

$\quad mjd\,(d1) < mjd\,(d2);$

public

$gt2 : Date \times Date \rightarrow \mathbb{B}$

$gt2\,(d1, d2) \triangleq$

$\quad mjd\,(d1) > mjd\,(d2);$

public

$le2 : Date \times Date \rightarrow \mathbb{B}$

$le2\,(d1, d2) \triangleq$

$\quad \neg\, gt2\,(d1, d2);$

public

$ge2 : Date \times Date \to \mathbb{B}$
$ge2\,(d1, d2) \triangleq$
$\quad \neg\, lt2\,(d1, d2);$

public

$former : Date \times Date \to \mathbb{B}$
$former\,(d1, d2) \triangleq$
$\quad lt2\,(d1, d2);$

public

$eq2 : Date \times Date \to$
$\quad \mathbb{B}$
$eq2\,(d1, d2) \triangleq$
$\quad mjd\,(d1) = mjd\,(d2);$

public

$leapYear : \mathbb{Z} \to$
$\quad \mathbb{B}$
$leapYear\,(y) \triangleq$
$\quad y \text{ mod } 400 = 0 \vee (y \text{ mod } 100 \neq 0 \wedge y \text{ mod } 4 = 0);$

public

$dayNumber : Date \to$
$\quad DayNumber$
$dayNumber\,(d) \triangleq$
$\quad \text{let } d1 = \text{floor } (mjd\,(d)) \text{ in}$
$\quad (d1 - 4) \text{ mod } 7;$

public

$dateToYmd2 : Date \to \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
$dateToYmd2\,(d) \triangleq$
$\quad \text{mk-}\,(year\,(d), month\,(d), day\,(d));$

public

$weekDay : Date \to DayName$
$weekDay\,(d) \triangleq$
$\quad DayNameList\,(dayNumber\,(d));$

public

$dayOfWeek : DayName \to DayNumber$
$dayOfWeek\,(dayName) \triangleq$
cases $dayName$:
$\textsc{Sun} \to 0,$
$\textsc{Mon} \to 1,$
$\textsc{Tue} \to 2,$
$\textsc{Wed} \to 3,$
$\textsc{Thr} \to 4,$
$\textsc{Fri} \to 5,$
$\textsc{Sat} \to 6$
end;

public

$firstDayOfWeek : \mathbb{Z} \times \mathbb{Z} \times DayName \to Date$
$firstDayOfWeek\,(y, m, dayName) \triangleq$
let $dnum = dayOfWeek\,(dayName),$
$firstDate = firstDayOfMonth\,(y, m),$
$diff = dnum - dayNumber\,(firstDate)$ in
cases true :
$(diff = 0) \to firstDate,$
$(diff > 0) \to firstDate.addDays\,(diff),$
$(diff < 0) \to firstDate.addDays\,((7 + diff) \bmod 7)$
end;

public

$lastDayOfWeek : \mathbb{Z} \times \mathbb{Z} \times DayName \to Date$
$lastDayOfWeek\,(y, m, dayName) \triangleq$
$firstDayOfWeek\,(y, (m + 1), dayName).subtractDays\,(7);$

public

$nthDayOfWeek : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times DayName \rightarrow$
$Date \times \mathbb{B}$
$nthDayOfWeek\,(y, m, n, dayName) \triangleq$
let $firstDate = firstDayOfWeek\,(y, m, dayName)$,
$RESULT = firstDate.addDays\,(7 \times (n - 1))$ in
cases $month\,(RESULT)$:
$(m) \rightarrow$ mk- $(RESULT, \text{true})$,
others $\rightarrow$ mk- $(RESULT, \text{false})$
end;

public
$firstDayOfMonth : \mathbb{Z} \times \mathbb{Z} \rightarrow Date$
$firstDayOfMonth\,(y, m) \triangleq$
$fromDate\,(y, m, 1)$;

public
$lastDayOfMonth : \mathbb{Z} \times \mathbb{Z} \rightarrow Date$
$lastDayOfMonth\,(y, m) \triangleq$
$fromDate\,(y, m + 1, 1).subtractDays\,(1)$;

public
$year : Date \rightarrow \mathbb{Z}$
$year\,(d) \triangleq$
if $monthAux\,(d) < 14$
then $yearAux\,(d) - 4800$
else $yearAux\,(d) - 4799$;

public
$month : Date \rightarrow \mathbb{Z}$
$month\,(d) \triangleq$
if $monthAux\,(d) < 14$
then $monthAux\,(d) - 1$
else $monthAux\,(d) - 13$;

public
$day : Date \rightarrow \mathbb{Z}$
$day\,(d) \triangleq$
$dayOfMonth\,(d)$;

public

$dayOfYear : Date \to \mathbb{Z}$
$dayOfYear\,(d) \triangleq$
  let $theFirstDayOfYear = fromDate\,(year\,(d), 1, 0)$ in
  floor $(mjd\,(d) - mjd\,(theFirstDayOfYear))$;
$dayOfMonth : Date \to \mathbb{Z}$
$dayOfMonth\,(d) \triangleq$
  floor $(dayAsReal\,(d))$;
$dayAsReal : Date \to \mathbb{R}$
$dayAsReal\,(d) \triangleq$
  $dateToYmdAux\,(d) + 122.1 -$
  floor $(365.25 \times yearAux\,(d)) -$ floor $(30.6001 \times monthAux\,(d))$;
$monthAux : Date \to$
  $\mathbb{Z}$
$monthAux\,(d) \triangleq$
  floor $((dateToYmdAux\,(d) + 122.1 -$ floor $(365.25 \times yearAux\,(d)))/30.6001)$;
$dateToYmdAux : Date \to \mathbb{R}$
$dateToYmdAux\,(d) \triangleq$
  let $jd = mjdToJd\,(mjd\,(d))$,
    $century =$ floor $((jd + 32044.9)/36524.25)$ in
  if $jd > 2299160$
  then $jd + 32044.9 + century - century$ div $4 + 0.5$
  else $jd + 32082.9 + 0.5$;
$yearAux : Date \to$
  $\mathbb{Z}$
$yearAux\,(d) \triangleq$
  floor $(dateToYmdAux\,(d)/365.25)$;

public

$addDays2 : Date \times \mathbb{Z} \to Date$
$addDays2\,(d, n) \triangleq$
  $d.addDays\,(n)$;

public

$subtractDate : Date \times Date \to \mathbb{Z}$
$subtractDate\,(d1, d2) \triangleq$
  floor $(mjd\,(d1) - mjd\,(d2))$;

public

$subtractDays2 : Date \times \mathbb{Z} \to Date$
$subtractDays2\,(d, n) \triangleq$
$d.subtractDays\,(n);$

public

$length : Date \times Date \to TermLength$
$length\,(d1, d2) \triangleq$
$subtractDate\,(d1, d2)$post cases true :
$(former\,(d1, d2)) \to RESULT \geq 0,$
others $\to RESULT \leq 0$
end;

public

$find : Date \times TermLength \to Date$
$find\,(d, l) \triangleq$
$addDays2\,(d, l)$post $length\,(d, RESULT) = l;$

public

$within : Date \times Date \times TermLength \to \mathbb{B}$
$within\,(d1, d2, l) \triangleq$
$length\,(d1, d2) \leq l;$

public

$mjdToJd : \mathbb{R} \to \mathbb{R}$
$mjdToJd\,(r) \triangleq$
$r + jdMinusMjd;$

public

$jdToMjd : \mathbb{R} \to \mathbb{R}$
$jdToMjd\,(r) \triangleq$
$r - jdMinusMjd;$

public

$ymdToYear : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \to$
$\mathbb{R}$
$ymdToYear\,(y, m, d) \triangleq$
$y + (m - 1)/12 + (d - 1)/365.25;$

public

$date2string : Date \to$ char*
$date2string\,(d) \triangleq$
$d.dateToYmdString\,();$

public

$string2date : \text{char}^* \to [Date]$

$string2date\,(s) \triangleq$

let mk-$(status, d) = fromString\,(s)$ in

if $status =$ true

then d

else nil ;

public

$time2string : Time \to \text{char}^*$

$time2string\,(t) \triangleq$

is not yet specified;

public

$string2time : \text{char}^* \to [Time]$

$string2time\,(s) \triangleq$

is not yet specified

operations

public

$mjd : Date \xrightarrow{o} \mathbb{R}$

$mjd\,(d) \triangleq$

(return $d.getMjd\,()$

);

public

$fromDate : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \xrightarrow{o} Date$

$fromDate\,(y, m, d) \triangleq$

let $[year, month] =$ if $(m > 2)$

then $[y + 4800, m + 1]$

else $[y + 4799, m + 13]$,

$century = year$ div 100,

$cc =$ if $(ymdToYear\,(y, m, d) > 1582.78)$

then $century$ div $4 - century - 32167$

else -32205 in

(dcl $dt : Date := \text{new}Date()$;

$\mathrm{dt}setMjd(\mathsf{floor}\ (365.25\times year)+\mathsf{floor}\ (30.6001\times month)+$
$d + cc - 0.5 - jdMinusMjd)$;
return dt
);
public
$fromString : \mathsf{char}^* \xrightarrow{o}$
$\mathbb{B} \times Date$
$fromString\,(yyyymmdd) \triangleq$
(if $\neg\, STR.isDigits\,(yyyymmdd)$
then return $(\mathsf{mk\text{-}}(\mathsf{false}, today\,()))$;
let $ymd = STR.asInteger\,(yyyymmdd)$,
$y = ymd\ \mathsf{div}\ 10000$,
$mmdd = ymd\ \mathsf{mod}\ 10000$,
$m = mmdd\ \mathsf{div}\ 100$,
$d = mmdd\ \mathsf{mod}\ 100$ in
if $fromDate\,(y, m, d).dateToYmdString\,() = yyyymmdd$
then return $\mathsf{mk\text{-}}(\mathsf{true}, fromDate\,(y, m, d))$
else return $\mathsf{mk\text{-}}(\mathsf{false}, fromDate\,(y, m, d))$
);
public
$isDateString : \mathsf{char}^* \xrightarrow{o}$
$\mathbb{B}$
$isDateString\,(yyyymmdd) \triangleq$
return $fromString\,(yyyymmdd).\#1$;
public
$fromMjd : \mathbb{R} \xrightarrow{o}$
$Date$
$fromMjd\,(modieiedJd) \triangleq$
(dcl $dt : Date := \mathsf{new}Date()$;
$\mathrm{dt}setMjd(modieiedJd)$;
return dt
);
public

$dateToYmd : () \xrightarrow{o} \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$
$dateToYmd\,() \triangleq$
return mk- $(year\,(\text{self}), month\,(\text{self}), day\,(\text{self}))$;

public

$dateToYmdString : () \xrightarrow{o} \text{char}^*$
$dateToYmdString\,() \triangleq$
(let $y = year\,(\text{self})$,
$m = month\,(\text{self})$,
$d = day\,(\text{self})$ in
cases true:
$(m < 10 \wedge d < 10) \rightarrow$ return $(asString\,(y)^\frown$"0"$^\frown asString\,(m)^\frown$ "0" $^\frown asString\,(d))$,
$(m < 10) \rightarrow$ return $(asString\,(y)^\frown$"0"$^\frown asString\,(m)^\frown$ $asString\,(d))$,
$(d < 10) \rightarrow$ return $(asString\,(y)^\frown asString\,(m)^\frown$"0"$^\frown$ $asString\,(d))$,
others $\rightarrow$ return $(asString\,(y)^\frown asString\,(m)^\frown asString\,(d))$
end
);

public

$lt : Date \xrightarrow{o} \mathbb{B}$
$lt\,(d) \triangleq$
return $mjd\,(\text{self}) < mjd\,(d)$;

public

$gt : Date \xrightarrow{o} \mathbb{B}$
$gt\,(d) \triangleq$
return $mjd\,(\text{self}) > mjd\,(d)$;

public

$le : Date \xrightarrow{o} \mathbb{B}$
$le\,(d) \triangleq$
return $\neg\, gt\,(d)$;

public

$ge : Date \xrightarrow{o} \mathbb{B}$
$ge\,(d) \triangleq$
return $\neg\, lt\,(d)$;

```
public
        eq : Date →o
            B
        eq (d) △
          return (mjd (self) = mjd (d));
public
        addDays : Z →o Date
        addDays (n) △
          (   dcl dt : Date := newDate();
              dtsetMjd(mjd (self) + n) ;
              return dt
          );
public
        subtractDays : Z →o Date
        subtractDays (n) △
          (   dcl dt : Date := newDate();
              dtsetMjd(mjd (self) − n) ;
              return dt
          );
        setMjd : R →o ()
        setMjd (d) △
          modifiedJulianDate := d;
        getMjd : () →o R
        getMjd () △
          return modifiedJulianDate
end Date
```

Test Suite : vdm.tc

Name	#Calls	Coverage
Date‘addDays2	0	0%
Date‘date2string	0	0%
Date‘dateToYmd2	0	0%
Date‘dateToYmdAux	23	85%
Date‘day	1	√
Date‘dayAsReal	1	√

Name	#Calls	Coverage
Date‘dayNumber	3	√
Date‘dayOfMonth	1	√
Date‘dayOfWeek	3	62%
Date‘dayOfYear	0	0%
Date‘eq2	0	0%
Date‘find	0	0%
Date‘firstDayOfMonth	3	√
Date‘firstDayOfWeek	3	65%
Date‘former	0	0%
Date‘ge2	0	0%
Date‘gt2	0	0%
Date‘jdToMjd	0	0%
Date‘lastDayOfMonth	0	0%
Date‘lastDayOfWeek	0	0%
Date‘le2	0	0%
Date‘leapYear	0	0%
Date‘length	0	0%
Date‘lt2	0	0%
Date‘mjdToJd	23	√
Date‘month	4	68%
Date‘monthAux	10	√
Date‘nthDayOfWeek	3	88%
Date‘string2date	0	0%
Date‘string2time	0	0%
Date‘subtractDate	0	0%
Date‘subtractDays2	0	0%
Date‘time2string	0	0%
Date‘today	0	0%
Date‘weekDay	0	0%
Date‘within	0	0%
Date‘year	1	68%
Date‘yearAux	12	√
Date‘ymdToYear	3	√

Name	**#Calls**	**Coverage**
Date‘addDays	6	√
Date‘dateToYmd	1	√
Date‘dateToYmdString	0	0%
Date‘eq	0	0%
Date‘fromDate	3	86%
Date‘fromMjd	0	0%
Date‘fromString	0	0%
Date‘ge	0	0%
Date‘getMjd	32	√
Date‘gt	0	0%
Date‘isDateString	0	0%
Date‘le	0	0%
Date‘lt	0	0%
Date‘mjd	32	√
Date‘setMjd	9	√
Date‘subtractDays	0	0%
Total Coverage		**40%**