Investor Class

CLASS ‘Investor’:

Create field ‘INVESTMENT’ of type Integer

Create field ‘gain’ of type Double

Create field ‘UUID’ of type String

Create field ‘bonds’ of type ArrayList<Bond> ?

METHOD ‘purchase’ <bond>: //////////

Add <bond> to ‘bonds’ collection

Set ‘fraction’ in <bond> to result of INVESTMENT/bond.getPrice

Set ‘purchaseDate’ in <bond> to Date ?

bond.setPurchaseDate(new SimpleDateFormat("yyyy-MM-dd").format(new Date()));

END METHOD.

METHOD ‘getBonds’: ///////////

Return bonds

END METHOD.

END CLASS.

Brain Class

CLASS ‘Brain’:

Create object field ‘investor’ of type Investor

Create collections field ‘bonds’ of type ?

METHOD ‘createBond’ <name, price, term, coupon, frequency>: ///////////

Add bond to ‘bonds’ collection with <name, price, term, coupon, frequency> ?

END METHOD.

METHOD ‘printBonds’: ////////////

Print as output all bonds in collection ‘bonds’

END METHOD.

METHOD ‘sumPayout’: ///////////

?Create local variable ‘sum’ of type Double, initialise to 0

FOR EACH Bond from ‘bonds’ in ‘investor’:

sum += investor.payoutF(bond)

Return sum

END METHOD.

METHOD ‘discountedPayout’ <bond, r>: ///////////

Create local variable ‘sum’ of type Double, initialise as 0

FOR i = 1; i < bond.getTerm; increment i:

sum += (investor.getINVESTMENT()\*bond.getCoupon()/Math.pow(1 + r, i))

sum += ((investor.getINVESTMENT()\*bond.getCoupon() + investor.getINVESTMENT())/Math.pow(1 + r, bond.getTerm()));

Return sum

END METHOD.

METHOD ‘macaulayD’ <bond, rate>: ////////////

Create local variable ‘temp’ of type Double, initialise as 0

FOR i = 1; i < bond.getTerm; increment i:

temp += (i\*bond.getCoupon()\*100/Math.pow(1 + rate,i))

ENDFOR.

Return result of (temp + ((bond.getTerm()\*100)/Math.pow(1 + rate,bond.getTerm()))) / discountedPayout(bond, rate)

END METHOD.

METHOD ‘iRR’ <bond, r>: ////////////

Initialise local variable ‘rate’, of type Double, as <r>

Initialise local variable ‘price’ as result of ‘getPrice’ in ‘bond’

Initialise local variable ‘i’, of type Integer, as 0

WHILE i < 50000:

IF result of ‘discountedPayout(bond, rate)’ < price THEN:

rate -= 0.00001

ELSE:

rate += 0.00001

ENDIF.

Increment i

ENDWHILE.

Return rate

END METHOD.

METHOD ‘getBonds’: //////////////

Return bonds

END METHOD.

END CLASS.

Pseudocode as OCL

In Investor:

update purchase(Bond bond): void

post: bonds.add(bond)

&& bond.setFraction(INVESTMENT/bond.getPrice())

&& bond.setPurchaseDate(new Date)

query(?) getBonds(): ArrayList<Bond>

post: return bonds

\*I don’t know when to use update or query\*

In Brain:

createBond(String name, double price, int term, double coupon, double frequency): void

post: bonds.add(new Bond(name, price, term, coupon, frequency))

printBonds(): void

post:

bonds->forAll(bond | print bond)

sumPayout(): int

post:

int sum = 0

investor.getBonds()->forAll(bond | sum += investor.payoutF(bond))

return sum

discountedPayout(Bond bond, double r): double

double sum = 0

for (int i = 1; i < bond.getTerm(); ++i){

sum += (investor.getINVESTMENT()\*bond.getCoupon()/Math.pow(1 + r, i));

}

sum += ((investor.getINVESTMENT()\*bond.getCoupon() + investor.getINVESTMENT())/Math.pow(1 + r, bond.getTerm()));

return sum;

}

public double macaulayD(Bond bond, double rate){

double temp = 0;

for (int i = 1; i <= bond.getTerm(); i++){

temp += (i\*bond.getCoupon()\*100/Math.pow(1 + rate,i));

}

return (temp + ((bond.getTerm()\*100)/Math.pow(1 + rate,bond.getTerm()))) / discountedPayout(bond, rate);

}

public double irr (Bond bond, double r) {

double rate = r;

double price = bond.getPrice();

int i = 0;

while( i < 50000) {

if (discountedPayout(bond,rate) < price) {

rate -= 0.00001;} else { rate += 0.00001; }

i++;

}

return rate;

}

public ArrayList<Bond> getBonds() {

return bonds;

}

Payment Class

CLASS ‘Payment’:

Create field ‘amount’ of type Double, initialise as zero

Create field ‘time’ of type Double, initialise as zero

CONSTRUCTOR METHOD ‘Payment’ <investedSum, ID>:

Set field ‘investedSum’ to <investedSum>

Set field ‘ID’ to <ID>

END CONSTRUCTOR METHOD.

METHOD ‘discount’:

END METHOD.

METHOD ‘totalAmount’:

END METHOD.

END CLASS.

METHOD ‘Destination’ <name, coordinates>:

Set field ‘name’ to <name>

Set field ‘coordinates’ to <coordinates>

END METHOD.

METHOD ‘getCoordinates’:

Return coordinates

END METHOD.

METHOD ‘getName’:

Return name

END METHOD.

END CLASS.

Aeroplane Class

CLASS ‘Aeroplane’:

Create String field ‘name’

Create Coordinates field ‘coordinates’

Create int field ‘speed’

Create int field ‘totalDistance’

Create int field ‘singleJourneyDistance’

Create int field ‘repairDistance’

Create int field ‘hour’

Create int field ‘numberOfRepairs’

Create int field ‘day’

METHOD ‘Aeroplane’ <name, coordinates, speed, totalDistance, repairDistance>:

Set field ‘name’ to < name >

Set field ‘coordinates’ to < coordinates >

Set field ‘speed’ to < speed >

Set field ‘totalDistance’ to < totalDistance >

Set field ‘repairDistance’ to < repairDistance >

END METHOD.

METHOD ‘singleFlight’ <startDestination, aeroplane, endDestination>:

Set field ‘singleJourneyDistance’ = 0

WHILE result of ‘getValueX’ through ‘getCoordinates’ in <aeroplane> !=

result of ‘getValueX’ through ‘getCoordinates’ in <endDestination> OR

result of ‘getValueY’ through ‘getCoordinates’ in <aeroplane> !=

result of ‘getValueY’ through ‘getCoordinates’ in <endDestination> :

IF result of ‘getValueX’ through ‘getCoordinates’ in <aeroplane> <

result of ‘getValueX’ through ‘getCoordinates’ in <endDestination> THEN:

Set ‘distanceRemainingX’ = result of ‘getValueX’ through ‘getCoordinates’ in <endDestination>

* result of ‘getValueX’ through ‘getCoordinates’ in <aeroplane>

IF ‘distanceRemainingX’ < ‘speed’ THEN:

Set ‘setValueX’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueX’ + ‘distanceRemainingX’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘distanceRemainingX’

ELSE:

Set ‘setValueX’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueX’ + ‘speed’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘speed’

ENDIF.

ELSE IF result of ‘getValueX’ through ‘getCoordinates’ in <aeroplane> >

result of ‘getValueX’ through ‘getCoordinates’ in <endDestination> THEN:

Set ‘distanceRemainingX’ = result of ‘getValueX’ through ‘getCoordinates’ in <aeroplane> - result of ‘getValueX’ through ‘getCoordinates’ in <endDestination>

IF ‘distanceRemainingX’ < ‘speed’ THEN:

Set ‘setValueX’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueX’ - ‘distanceRemainingX’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘distanceRemainingX’

ELSE:

Set ‘setValueX’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueX’ - ‘speed’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘speed’

ENDIF.

ENDIF.

IF result of ‘getValueY’ through ‘getCoordinates’ in <aeroplane> <

result of ‘getValueY’ through ‘getCoordinates’ in <endDestination> THEN:

Set ‘distanceRemainingY’ = result of ‘getValueY’ through ‘getCoordinates’ in <endDestination>

* result of ‘getValueY’ through ‘getCoordinates’ in <aeroplane>

IF ‘distanceRemainingY’ < ‘speed’ THEN:

Set ‘setValueY’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueY’ + ‘distanceRemainingY’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘distanceRemainingY’

ELSE:

Set ‘setValueY’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueY’ + ‘speed’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘speed’

ENDIF.

ELSE IF result of ‘getValueY’ through ‘getCoordinates’ in <aeroplane> >

result of ‘getValueY’ through ‘getCoordinates’ in <endDestination> THEN:

Set ‘distanceRemainingY’ = result of ‘getValueY’ through ‘getCoordinates’ in <aeroplane> - result of ‘getValueY’ through ‘getCoordinates’ in <endDestination>

IF ‘distanceRemainingY’ < ‘speed’ THEN:

Set ‘setValueY’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueY’ - ‘distanceRemainingY’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘distanceRemainingY’

ELSE:

Set ‘setValueY’ through ‘getCoordinates’ in <aeroplane>

= ‘getValueY’ - ‘speed’

Set ‘singleJourneyDistance’ = ‘singleJourneyDistance’ + ‘speed’

ENDIF.

ENDIF.

Set ‘totalDistance’ = ‘totalDistance’ + ‘singleJourneyDistance’

ENDWHILE.

Return singleJourneyDistance

END METHOD.

METHOD ‘getTotalDistance’:

Return totalDistance

END METHOD.

METHOD ‘getCoordinates’:

Return coordinates

END METHOD.

METHOD ‘getName’:

Return name

END METHOD.

METHOD ‘getRepairDistance’:

Return repairDistance

END METHOD.

METHOD ‘getNumberOfRepairs’:

Return numberOfRepairs

END METHOD.

METHOD ‘getDay’:

Return day

END METHOD.

METHOD ‘setDay’ <day>:

Set field ‘day’ to <day>

END METHOD.

METHOD ‘setTotalDistance’ <totalDistance>:

Set field ‘totalDistance’ to <totalDistance>

END METHOD.

METHOD ‘setNumberOfRepairs’ <numberOfRepairs>:

Set field ‘numberOfRepairs’ to <numberOfRepairs>

END METHOD.

END CLASS.