­­EECS 132 introduction to JAVA

Name: Ming Zhuang

Test file of the Project 1

* Class Asset
  1. Constructor & getName & getCostBasis

The values initialized in the constructor should be returned.

> Asset myAsset = new Asset("House", 25000.0)

> myAsset.getName()

"House"

> myAsset.getCostBasis()

25000.0

Correct.

* 1. setName & getName

change a certain name, we should get exactly the same return String.

> Asset myAsset = new Asset(House, 25000.0)

> myAsset.setName("apartment")

> myAsset.getName()

"apartment"

Correct.

* 1. setDescription & getDescrption

give your asset some description and get the same String back

> Asset myAsset = new Asset(House, 25000.0)

> myAsset.setDescription("beautiful one")

> myAsset.getDescription()

"beautiful one"

Correct.

* 1. setCostBasis & getCostBasis

change the costBasis , we should get exactly the same number back

> Asset myAsset = new Asset(House, 25000.0)

> myAsset.setCostBasis(50000.0)

> myAsset.getCostBasis()

50000.0

Correct.

* 1. setCurrentPrice & getCurrentPrice
* change the current price, and we should get exactly the same number back

> Asset myAsset = new Asset(House, 25000.0)

> myAsset.setCurrentPrice(99999.9)

> myAsset.getCurrentPrice()

99999.9

Correct.

* 1. setCapitalGains & getCaptialGains
* change the current price, and we should get exactly the same number back

> Asset myAsset = new Asset(House, 25000.0)

> myAsset.setCapitalGains(66666.6)

> myAsset.getCapitalGains()

66666.6

Correct.

* Class Bond
  1. Constructor & getName & getPrincipal & getInterestRate

Initialize the name, the principal and the interest rate and get them back exactly the same; check if current price equals principal; check if cost basis is 0.

> Bond myBond = new Bond("first", 2000, 0.35)

> myBond.getPrincipal()

2000

> myBond.getInterestRate()

0.35

> myBond.getName()

"first"

> myBond.getCurrentPrice()

2000.0

> myBond.getCostBasis()

0.0

Correct.

* 1. setInterestRate & getInterestRate

set the interest rate and then get it back

> Bond myBond = new Bond("first", 2000, 0.35)

> myBond.setInterestRate(0.084)

> myBond.getInterestRate()

0.084

Correct.

* 1. payInterest

set the interest rate to 0.1 and the principal to 1000. We should get 0.1 \* 1000 = 100 back.

> Bond myBond = new Bond("first", 1000,0.1)

> myBond.payInterest()

100.0

Correct.

* 1. Buy & getNumberOwned

The original cost basis is 0. And the make the current price equals principal which is 1000. So that after the method runs, it should return current price which is 1000, the cost basis will be 1000, and the number owned is 1.

> Bond myBond = new Bond("first", 1000,0.1)

> myBond.buy()

1000.0

> myBond.getNumberOwned()

1

> myBond.getCostBasis()

1000.0

Correct.

* 1. Sell

First create a new class. Then test it, since the number owner is 0, we should get 0.0

> Bond myBond = new Bond("first", 1000, 0.1)

> myBond.sell()

0.0

Correct.

Then we run method buy twice. Then the cost basis is 2000, and the current price is 1000, and the number owned is 2.

Change the price from 1000 to 1500

After we run sell once,

the current price is 1500, which will be returned

the cost basis will become (2000- (2000 / 2) = 1000

and the capital gain will be 0 + (1500 - 1000) = 500. Buy at 1000, sell at 1500, I get 500. Make sense.

the number owned will become 2 – 1 = 1

> Bond myBond = new Bond("first", 1000, 0.1)

> myBond.buy()

1000.0

> myBond.buy()

1000.0

> myBond.getNumberOwned()

2

> myBond.getCostBasis()

2000.0

> myBond.getCapitalGains()

0.0

> myBond.setCurrentPrice(1500)

> myBond.sell()

1500.0

> myBond.getNumberOwned()

1

> myBond.getCostBasis()

1000.0

> myBond.getCapitalGains()

500.0

Correct.

* Class Equity
  1. Constructor & getSymbol

Initialize the name, symbol and the current price and get them back exactly the same.

Note that the cost basis should always be 0.

> Equity myEquity = new Equity("holyEquity", '!', 3000.0)

> myEquity.getName()

"holyEquity"

> myEquity.getSymbol()

'!'

> myEquity.getCurrentPrice()

3000.0

> myEquity.getCostBasis()

0.0

Correct.

* 1. setNumberShares & getNumberShares

change the numberShares and get it back exactly the same.

> Equity myEquity = new Equity("holyEquity", '!', 3000.0)

> myEquity.setNumberShares(0.5)

> myEquity.getNumberShares()

0.5

Correct.

* Class MutualFund
  1. Constructor

Set the initial value. Should get them back exactly the same.

> MutualFund myMutualFund = new MutualFund("BigFund", '@', 3003.3)

> myMutualFund.getName()

"BigFund"

> myMutualFund.getSymbol()

'@'

> myMutualFund.getCurrentPrice()

3003.3

Correct.

* 1. setLoad & getLoad

set the load and then get it back exactly as it was set.

> MutualFund myMutualFund = new MutualFund("BigFund", '@', 3003.3)

> myMutualFund.setLoad(0.9)

> myMutualFund.getLoad()

0.9

Correct.

* 1. Buy

Make the load 10%(0.1). Set number of shares own 0. Set current price 100.0.

Then we give a invention of money of 1000.0.

Accroding to number of shares owned = money invented x (100% - load) / current price,the numberShares should become 9. The costBasis should be 1000.0.

And the return value is money invented, which is 1000.0

> MutualFund myMutualFund = new MutualFund("smallFund", '?', 100.0)

> myMutualFund.setLoad(0.1)

> myMutualFund.setNumberShares(0.0)

> myMutualFund.buy(1000.0)

1000.0

> myMutualFund.getCostBasis()

1000.0

> myMutualFund.getNumberShares()

9.0

Correct.

Then test the situation when the input is 0.0 or negative values.

> MutualFund myMutualFund = new MutualFund("smallFund", '?', 100.0)

> myMutualFund.buy(0.0)

0.0

> myMutualFund.buy(-1000.0)

0.0

Correct.

* 1. Sell

We use the last buy scenario. First test when the input money withdrawn is 0 or negative value.

We should both get 0.

> MutualFund myMutualFund = new MutualFund("myFund", '$', 1000.0)

> myMutualFund.sell(0.0)

0.0

> myMutualFund.sell(-1020.0)

0.0

Correct.

Then we test the situation that the moneyWithdrawn is larger than the current value(price times number)

We should get 0.

> MutualFund myMutualFund = new MutualFund("myFund", '$', 1000.0)

> myMutualFund.sell(1020.0)

0.0

Correct.

Then we resume the last scenario when we finished buying.

> MutualFund myMutualFund = new MutualFund("myFund", '$', 100.0)

> myMutualFund.setLoad(0.1)

> myMutualFund.setNumberShares(0.0)

> myMutualFund.buy(1000.0)

1000.0

> myMutualFund.getCostBasis()

1000.0

> myMutualFund.getNumberShares()

9.0

Then we go for the selling.

I set the moneyWithdrawn 600; we will get return 600.0;

the number of shares will be (0.9 – 600 / 100) = 3.

And the costBasis will be (100 \* (3 / 9)) = 333.33333…;

The capital gains is (0 + (600 – (1000 – 333.333…))) is -66.666… which means that you loss 66.666… unit of money.

> myMutualFund.sell(600.0)

600.0

> myMutualFund.getNumberShares()

3.0

> myMutualFund.getCostBasis()

333.33333333333333

> myMutualFund.getCapitalGains()

-66.666666666666674

Correct.

* Class Stock
  1. Constructor

Set the name, symbol, and the current price.

Get them back. See if it’s the same.

> Stock myStock = new Stock("HeyStock", '@', 1000.3)

> MyStock.getName()

"HeyStock"

> MyStock.getSymbol()

'@'

> MyStock.getCurrentPrice()

1000.3

Correct.

* 1. Buy

Input: numberShares = 5, currentPrice = 1000.0, commission = 2000.0

The return should be 5 \* 1000 + 2000 = 7000

Number of shares become 5.0

Costbasis should be 7000.0

> Stock myStock = new Stock("HeyStock", '@', 1000.0)

> myStock.getNumberShares()

0.0

> myStock.getCostBasis()

0.0

> myStock.buy(5, 2000.0)

7000.0

> myStock.getNumberShares()

5.0

> myStock.getCostBasis()

7000.0

Correct.

* 1. Sell

First test the situation that number of selling is bigger than the number of stock which is currently owned.

We should get 0.

> Stock myStock = new Stock("HeyStock", '@', 1000.0)

> myStock.buy(5, 2000.0)

7000.0

> myStock.sell(7, 800.0)

0.0

Correct.

Then test normal situation.

First buy.

Input: numberShares = 5, currentPrice = 1000.0, commission = 100.0

Return should be 5 \* 1000 + 100 = 5100

costBasis = 5100, numberShares = 5, capitalGains = 0

> Stock myStock = new Stock("HeyStock", '@', 1000.0)

> myStock.buy(5, 100.0)

5100.0

> myStock.getCostBasis()

5100.0

> myStock.getNumberShares()

5.0

> myStock.getCapitalGains()

0.0

Then sell 1 stock, with the commission 100.

Return will be (1\*1000 - 100) = 900

Number of shares will be 4

costBasis will be 4/5 \* 5100 = 4080

decreased costBasis = 1020

capitalGains = 900 – 1020 = -120

> Stock myStock = new Stock("HeyStock", '@', 1000.0)

> myStock.buy(5, 100.0)

5100.0

> myStock.getCostBasis()

5100.0

> myStock.getNumberShares()

5.0

> myStock.getCapitalGains()

0.0

> myStock.sell(1, 100.0)

900.0

> myStock.getNumberShares()

4.0

> myStock.getCostBasis()

4080.0

> myStock.getCapitalGains()

-120.0

Correct.

* 1. Split

Try either the numerator or the denominator is 0.

Return should be 0.

> Stock myStock = new Stock ("NASDAQ", 'g', 10)

> myStock.buy(2,0)

20.0

> myStock.split(0,2)

0.0

> myStock.split(2,0)

0.0

Correct.

If these is no fractional value. Only numberOfShares change. And the return of the method is 0.

> Stock myStock = new Stock ("NASDAQ", 'g', 10)

> myStock.buy(2,0)

20.0

> myStock.getNumberShares()

2.0

> myStock.getCostBasis()

20.0

> myStock.getCapitalGains()

0.0

> myStock.split(2,1)

0.0

> myStock.getNumberShares()

4.0

> myStock.getCostBasis()

20.0

> myStock.getCapitalGains()

0.0

Correct.

If there is fractional value,

Price per share is 100, buy 3 shares. Check the number of shares is 3, costbasis is 300, and capital gains is 0.

Then split (1,2), 3 \* (1/2) is 1.5, we sell 0.5.

NumberShares = 1.5 – 0.5 = 1.0

costBasis = 300 \* ( 1 / 1.5 ) = 200

CaptialGains = 0 + ( 0.5 \* 100 ) – (300 - 200) = -50

(we get 50 cash, but lost stock that worth 100, so we have negative 50 capital gain)

> Stock myStock = new Stock ("w", 's', 100)

> myStock.buy(3,0)

300.0

> myStock.getNumberShares()

3.0

> myStock.getCostBasis()

300.0

> myStock.getCapitalGains()

0.0

> myStock.split(1,2)

50.0

> myStock.getNumberShares()

1.0

> myStock.getCostBasis()

200.0

> myStock.getCapitalGains()

-50.0

Correct.

* Class CashAsset
  1. Constructor & getSavingsRate & getLoanRate & getLoanLimit & getOverdraftPenalty

Initialize the values and get them back exactly the same.

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.getSavingsRate()

0.3

> newAccount.getLoanRate()

0.4

> newAccount.getLoanLimit()

1000.0

> newAccount.getOverdraftPenalty()

20.0

Correct.

* 1. setSavingsRate & getSavingsRate

set the savings rate and then get it back exactly the same.

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.setSavingsRate(0.2)

> newAccount.getSavingsRate()

0.2

Correct.

* 1. setLoanRate & getLoanRate

set the loan rate and get it back exactly the same

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.setLoanRate(0.9)

> newAccount.getLoanRate()

0.9

Correct.

* 1. setLoanLimit & getLoanLimit

set the loan limit and get it back exactly the same

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.setLoanLimit(9999.99)

> newAccount.getLoanLimit()

9999.99

Correct.

* 1. setOverdraftPenalty & getOverdraftPenalty

set the overdraft penalty and get it back exactly the same

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.setOverdraftPenalty(1.0)

> newAccount.getOverdraftPenalty()

1.0

Correct.

* 1. deposit & getBalance

first make sure the balance is 0.

Make a 10.0 deposit

Check if the balance is 10.0 now

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.getBalance()

0.0

> newAccount.deposit(10.0)

> newAccount.getBalance()

10.0

Correct.

* 1. withdraw & getBalance

Test fail case. First make the boolean false and amount > balance.

Balance do not change.

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.deposit(10.0)

> newAccount.getBalance()

10.0

> newAccount.withdraw(11.0, false)

False

> newAccount.getBalance()

10.0

Correct.

Then test the normal case.

Deposit 10.0 withdraw 5.0

Return should be true and the balance should become 5.0

> CashAsset newAccount = new CashAsset(0.3, 0.4, 1000.0, 20.0)

> newAccount.deposit(10.0)

> newAccount.getBalance()

10.0

> newAccount.withdraw(5.0, true)

true

> newAccount.getBalance()

5.0

Correct.

* 1. processDay & getAccruedInterest

if the account balance is positive

make the saving interest rate 3.65

then accrued interest should be 10 \* ( 3.65 / 365 ) = 0.1

> CashAsset newAccount = new CashAsset(3.65, 6.56, 1000.0, 20.0)

> newAccount.deposit(10.0)

> newAccount.getBalance()

10.0

> newAccount.processDay()

> newAccount.getAccruedInterest()

0.1

Correct.

If the account balance is negative 100.

Make the loan interest rate 7.3

Then the accrued interest rate should be 100 \* ( 7.3 / 365 ) = 2

> CashAsset newAccount = new CashAsset (4, 7.3, 0, 200.0)

> newAccount.withdraw(100, true)

true

> newAccount.getBalance()

-100.0

> newAccount.getAccruedInterest()

0.0

> newAccount.processDay()

> newAccount.getAccruedInterest()

-1.97

* 1. processMonth & getAccruedInterest & getBalance

set overdraft penalty 200. We overdraw 100 from an empty account to make it -100.

Run on processDay, make accruedInterest -1.97, and numberOverdrafted 1.

So accruedInterest will be cleared, balance = (-100 + -1.97 + -200 ) = -301.97

> CashAsset newAccount = new CashAsset (4, 7.3, 0, 200.0)

> newAccount.withdraw(100, true)

true

> newAccount.getBalance()

-100.0

> newAccount.getAccruedInterest()

0.0

> newAccount.processDay()

> newAccount.getAccruedInterest()

-1.97

> newAccount.processMonth()

> newAccount.getAccruedInterest()

0.0

> newAccount.getBalance()

-301.97

Correct.

* 1. setAccruedInterest & getAccruedInterest

set the accrued interest and get it back

> CashAsset newAccount = new CashAsset(3.65, 7.3, 1000.0, 20.0)

> newAccount.setAccruedInterest(20.0)

> newAccount.getAccruedInterest()

20.0

Correct.

* Class Customer
  1. Constructor1 & getCommissionAmount & getCashAccount

Initialize the instance, get the commission back and get the address of the cashAsset instance back.

Make sure it is the very Cash account we want by checking the savings rate, loan rate, loanLimit and overdraftPenalty.

> Customer customer1 = new Customer (new CashAsset(0.5, 0.6, 10, 100), 100.0)

> customer1.getCommissionAmount()

100.0

> customer1.getCashAccount()

CashAsset@1c4d0d9

> customer1.getCashAccount().getSavingsRate()

0.5

> customer1.getCashAccount().getLoanRate()

0.6

> customer1.getCashAccount().getLoanLimit()

10.0

> customer1.getCashAccount().getOverdraftPenalty()

100.0

Correct.

* 1. Constructor2 & getCashAccount & getBond & getMutualFund & getStock & getCommissionAmount

Initialize the four types of account for the customer and get their address as well as the amount of the commission back.

And check the specific parameter in each instance to make sure it is the very instance that you want.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCommissionAmount()

100.0

> customer2.getCashAccount()

CashAsset@415962

> customer2.getCashAccount().getOverdraftPenalty()

999.0

> customer2.getBond()

Bond@19f77fc

> customer2.getBond().getInterestRate()

1.0

> customer2.getMutualFund()

MutualFund@10285cc

> customer2.getMutualFund().getSymbol()

'@'

> customer2.getStock()

Stock@fa0fdf

> customer2.getStock().getSymbol()

'#'

Correct.

* 1. setBond & getBond

set a new bond instance which is different from the previous one, to make sure it is different:

change the input parameter of the new Bond using setter methods

Using getInterestRate methods to check it is exactly what you initialize.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getBond().getInterestRate()

1.0

> customer2.setBond(new Bond ("another", 150, 4.1))

> customer2.getBond().getInterestRate()

4.1

Correct.

* 1. setMutualFund & getMutualFund

set a new mutual fund instance which is different from the previous one, to make sure it is different:

change the input parameter using the new MutualFund setter methods

Using getSymbol methods to check the symbol is exactly what you initialize.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getMutualFund().getSymbol()

'@'

> customer2.setMutualFund(new MutualFund ("another", 'b', 200))

> customer2.getMutualFund().getSymbol()

'b'

Correct.

* 1. setStock & getStock

set a new stock instance which is different from the previous one, to make sure it is different:

change the input parameter using the new Stock setter methods

Using getSymbol methods to check the symbol is exactly what you initialize.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getStock().getSymbol()

'#'

> customer2.setStock(new Stock ("another", '?', 200))

> customer2.getStock().getSymbol()

'?'

Correct.

* 1. setCashAccount & getCashAccount

set a new cash account instance which is different from the previous one, to make sure it is different:

change the input parameter using the new CashAccount setter methods

Using getLoanLimit methods to check the loan limit is exactly what you initialize.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCashAccount().getLoanLimit()

100.0

> customer2.setCashAccount(new CashAsset (1.0, 8.0, 9999, 20))

> customer2.getCashAccount().getLoanLimit()

9999.0

Correct.

* 1. setComiwemissionAmount & getCommissionAmount

Set a new commission amount which is different from the previous one and get it back.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCommissionAmount()

100.0

> customer2.setCommissionAmount(9999)

> customer2.getCommissionAmount()

9999.0

Correct.

* 1. currentValue

get the value in the four components and make sure the sum is presented exactly the same in the return.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(1000)

> customer2.buyBond()

true

> customer2.buyBond()

true

> customer2.buyMutualFund(100)

true

> customer2.buyStockShares(5)

true

> customer2.getCashAccount().getBalance()

100.0

> customer2.getBond().getCurrentPrice()

100.0

> customer2.getStock().getCurrentPrice()

100.0

> customer2.getStock().getNumberShares()

5.0

> customer2.getMutualFund().getNumberShares()

1.0

> customer2.getMutualFund().getCurrentPrice()

100.0

> customer2.currentValue()

800.0

100 + 100 + 100 \* 5 + 100 \* 1 = 800

Correct.

* 1. getCapitalGains

First we give the customer some deposit. We buy some of the bond, mutual fund and stock.

And we raise the price and sell our bond, mutual fund and stock.

We get the three capital gains and check if the capital gain of the customer equal the sum.

>Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCapitalGains()

0.0

> customer2.deposit(10000)

> customer2.buyBond()

true

> customer2.buyBond()

true

> customer2.buyStockShares(10)

true

> customer2.buyMutualFund(220.3)

true

> customer2.getMutualFund().setCurrentPrice(300)

> customer2.getMutualFund().sell(200)

200.0

> customer2.getStock().setCurrentPrice(700)

> customer2.getStock().sell(8,10)

5590.0

> customer2.getBond().setCurrentPrice(300)

> customer2.getBond().sell()

300.0

> customer2.getBond().getCapitalGains()

200.0

> customer2.getMutualFund().getCapitalGains()

133.33333333333337

> customer2.getStock().getCapitalGains()

4710.0

> customer2.getCapitalGains()

5043.333333333333

200 + 133.33333333 + 4710 = 5043.333333333

Correct.

* 1. deposit & getBalance

First check balance (should be 0)

Then deposit 10.

And get the balance back. Should be 10.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCashAccount().getBalance()

0.0

> customer2.deposit(10)

> customer2.getCashAccount().getBalance()

10.0

Correct.

* 1. Withdraw & getBalance

If the balance is smaller than the amount, we should get false.

Since now the deposit is 10, we try to withdraw 11.

> customer2.withdraw(11)

False

And we try to withdraw 9 and the we should get the balance is 1

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(10)

> customer2.getCashAccount().getBalance()

10.0

> customer2.withdraw(9)

true

> customer2.getCashAccount().getBalance()

1.0

Correct.

* 1. sellBond & getNumberOwned & getBalance

if the number owned of the bond is 0, and we call sellBond, we should get nothing, and nothing should happen.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getBond().getCostBasis()

0.0

> customer2.getBond().getNumberOwned()

0

> customer2.sellBond()

> customer2.getBond().getNumberOwned()

0

> customer2.getBond().getCostBasis()

0.0

Correct.

if the number owned of the bond is not 0. When we call sellBond, we should get nothing.

After buy 3 bonds, we raise the price to 600.

And when we have 8500 in our cash account, when we sell a 600 bond, we get the account balance is 9100.

After sell 1 bond, the numberOwned of Bond decrease from 6 to 5

We have 2 bonds in hand which is bought by 500 each, we will have a costBasis of 1000.

And after buy the bond for 500 and sell it for 600, we get 100 profit, which is the CapitalGains.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(10000)

> customer2.buyBond()

true

> customer2.buyBond()

true

> customer2.buyBond()

true

> customer2.getCashAccount().getBalance()

8500.0

> customer2.getBond().getCostBasis()

1500.0

> customer2.getBond().getNumberOwned()

3

> customer2.getBond().setCurrentPrice(600)

> customer2.sellBond()

> customer2.getCashAccount().getBalance()

9100.0

> customer2.getBond().getCostBasis()

1000.0

> customer2.getBond().getNumberOwned()

2

> customer2.getBond().getCapitalGains()

100.0

Correct.

* 1. buyBond & getNumberOwned & getBalance

if current price of the bond Is bigger than the current price of the customer, we will get false.

So when the price of the bond is 500 and the current price of the customer is 0, we get false.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.currentValue()

500.0

> customer2.buyBond()

true

> customer2.currentValue()

0.0

> customer2.buyBond()

False

Correct.

If the current price of the customer is big enough.

Make the deposit 10000, when we buy a 50 bond, we get the balance = 10000 – 50 = 9500

And the cost basis = 0 + 500 = 500

The number owned of the bond is 1

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(10000)

> customer2.buyBond()

true

> customer2.getCashAccount().getBalance()

9500.0

> customer2.getBond().getCostBasis()

500.0

> customer2.getBond().getNumberOwned()

1

Correct.

* 1. payBondInterest & getBalance

The constructor set the principal 100 and the interest rate 1, so the balance will increase 100.

First check the balance should be 0, then make sure the balance will be 100.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 100, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.getCashAccount().getBalance()

0.0

> customer2.payBondInterest()

> customer2.getCashAccount().getBalance()

100.0

Correct.

* 1. withdrawMutualFund & getBalance

use 10000 buy 5000 usd mutual fund.

Raise the price to 200, withdraw 3000 usd from the mutual fund.

Balance = 5000 + 3000 = 8000;

NumberShares = 50 – 3000 / 200 = 35;

CostBasis = 35 / 50 \* 5000 = 3500;

capitalGains = 0 + 3000 – ( 5000 – 3500 ) = 1500

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(10000)

> customer2.getCashAccount().getBalance()

10000.0

> customer2.getMutualFund().getNumberShares()

0.0

> customer2.getMutualFund().getCostBasis()

0.0

> customer2.buyMutualFund(5000)

true

> customer2.getCashAccount().getBalance()

5000.0

> customer2.getMutualFund().getNumberShares()

50.0

> customer2.getMutualFund().getCostBasis()

5000.0

> customer2.getMutualFund().getCapitalGains()

0.0

> customer2.getMutualFund().setCurrentPrice(200)

> customer2.withdrawMutualFund(3000)

> customer2.getCashAccount().getBalance()

8000.0

> customer2.getMutualFund().getNumberShares()

35.0

> customer2.getMutualFund().getCostBasis()

3500.0

> customer2.getMutualFund().getCapitalGains()

1500.0

Correct.

* 1. buyMutualFund & getBalance

if current price of mutualFund < customer current value

Deposit 10000. Balance is 10000. NumberShares and CostBasis is 0.

Buy 5000 usd MUtualFund.

Balance = 10000- 5000 = 5000,

NumberShares = 5000 / 100 = 50,

CostBasis = money spent = 5000.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 100.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(10000)

> customer2.getCashAccount().getBalance()

10000.0

> customer2.getMutualFund().getNumberShares()

0.0

> customer2.getMutualFund().getCostBasis()

0.0

> customer2.buyMutualFund(5000)

true

> customer2.getCashAccount().getBalance()

5000.0

> customer2.getMutualFund().getNumberShares()

50.0

> customer2.getMutualFund().getCostBasis()

5000.0

Correct.

If current price of mutualFund > customer current value

We should get false.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 1000.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.currentValue()

500.0

> customer2.buyMutualFund(200)

false

* 1. sellStockShares & getNumberShares & getBalance

we have 5 stocks and per stock is 100.

Raise the price to 200 and sell 4 stocks.

numberShares = 5 - 4 = 1

CostBasis = ( 1 / 5 ) \* 600 = 120

Capital gains = 0 + (4 \* 200 – 100 ) – (600 - 120) = 220

Balance = 400 + (4 \* 200) – 100 = 1100

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 1000.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(1000)

> customer2.getStock().getNumberShares()

0.0

> customer2.getStock().getCostBasis()

0.0

> customer2.buyStockShares(5)

true

> customer2.getCashAccount().getBalance()

400.0

> customer2.getStock().getNumberShares()

5.0

> customer2.getStock().getCostBasis()

600.0

> customer2.getStock().getCapitalGains()

0.0

> customer2.getStock().setCurrentPrice(200)

> customer2.sellStockShares(4)

> customer2.getCashAccount().getBalance()

1100.0

> customer2.getStock().getNumberShares()

1.0

> customer2.getStock().getCostBasis()

120.0

> customer2.getStock().getCapitalGains()

220.0

Correct.

* 1. buyStockShares & getNumberShares & getBalance

if numberShares \* stock.getCurrentPrice + commissionAmount > this.currentValue

we should get false.

At this time numberShares \* stock.getCurrentPrice + commissionAmount = 10 \* 100 + 100 = 1100. currentValue = 500.

So we get false.

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 1000.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.currentValue()

500.0

> customer2.buyStockShares(10)

False

Correct.

if numberShares \* stock.getCurrentPrice + commissionAmount < this.currentValue

deposit 1000. Every stock is 100. Buy 5 stocks.

numberShares = 500 / 100 = 5

costBasis = ( 5 \* 100 + 100) = 600

Account balance = 1000 – ( 5 \* 100 + 100) = 400

> Customer customer2 = new Customer (new CashAsset(0.8, 1.6, 100, 999), new Bond ("myBond", 500, 1.0), new MutualFund("myMutualFund", '@', 1000.0), new Stock("myStock", '#', 100.0), 100.0)

> customer2.deposit(1000)

> customer2.getCashAccount().getBalance()

1000.0

> customer2.getStock().getNumberShares()

0.0

> customer2.getStock().getCostBasis()

0.0

> customer2.buyStockShares(5)

true

> customer2.getStock().getNumberShares()

5.0

> customer2.getStock().getCostBasis()

600.0

> customer2.getCashAccount().getBalance()

400.0

Correct.