Customer *(Serializable)*

Attributes

private custNumber: String	A unique identifier for a customer; must be 7 digits in length; must be greater than 10000
private tin: String	Customer's tax reporting number - TIN, EIN, or SSN; must be 9 digits long
private last: String	Customer's last name (or surname) as per legal identification; If a value is offered, it must be neither null nor blank.
private first: String	Customer's first (or given) name as per legal indentification. If a value is offered, it must be neither null nor blank.
private margin: int	Margin limit for trading; expressed as whole dollars; if equal to zero - customer is not authorized for margin trading. Default value is zero. Negative values are not permitted.
private drip: boolean	Indicates customer's instructions on dividend reinvestment: true = automatic reinvestment; false = by customer instruction
private futures: boolean	Indicates whether customer is authorized for futures contracts trading

Operations

public Customer(nbr: String, id: String, lName: String, fName: String, lim: int, reUp: boolean, fAuth: boolean): Customer	Full constructor
public describeCustomer( ): String	Returns a formatted String object with a verbose description of the Customer object: Customer *custNumber*, *first last*. The account carries a margin limit of $*margin*. The account will reinvest dividends as recieved. The account is authorized to trade in futures contracts. **NOTE:** Alternates for the last two lines will read "Dividends will be deposited as cash funds." (when **drip**=false) and "The account is not authorized to trade futures contracts." (when *futures* = false).
public toString( ): String	Returns a formatted String object containing raw values of all instance variables: Customer:*custNumber*, *tin*, *last*, *first*, *margin*, *drip*, *futures*

NOTE: There can be no null constructor defined for this class.

Security *(Abstract)*

Interfaces: CostBasis, Serializable

Attributes

private custNumber: String	A unique identifier for a customer; must be 7 digits in length; must be greater than 10000
private purchDt: int	Purchase date of Security in Julian date format: *yyyyddd.* The value for year must be greater than 1900. The value for days must be between 1 and 365, inclusive.
private purchPrc: double	Purchase price of Security, per share, in dollars
private shares: double	Number of shares purchased; for display purposed, decimal positions should be limited to three (3)
private symbol: String	Market symbol of Security

Operations

public Security(nbr: String, date: int, price: double, qty: double, sym: String): Security	Full constructor
public toString( ): String	Returns a formatted String object describing the Security object: The Security belongs to Customer# *custNumber*; *shares* shares of *symbol* were purchased on *purchDt* for $*purchPrc* per share.

NOTES:

The interface method calcCost is not given a concrete definition here.

There can be no null constructor defined for this class.

MutualFund (subclass of Security)

Attributes

private type: String	Identifies general type of Mutual Fund: bonds, money market, exchange, etc.
private admin: double	Administrative fee cap, expressed as a percentage of value of funds held (e.g.: 0.0024 or 0.03)
private rptPeriod: char	Reporting period, expressed as character. The valid values are: A: annual Q: quarterly M: monthly
private mgmt: boolean	Describes whether or not the fund is "actively managed": true = active; false = passive management

Operations

public MutualFund(nbr: String, date: int, price: double, qty: double, sym: String, sort: String, cost: double, rpt: char	Full constructor
public toString( ):String	Returns a formatted String object describing the MutualFUnd object: The MutualFund belongs to Customer# *custNumber* ; *shares* shares of *symbol* were purchased on *purchDt* for $*purchPrc* per share. This is a *type* fund. Admin costs are capped at *admin* . Reporting cycle is Annual/Quarterly/Monthly(1). The fund is managed actively/passively(2). (1) Based on *rptPeriod* value. (2) Based on *mgmt* value.
public calcCost( ): double	Cost is calculated as: (*shares* * *purchPrc*) * (1 + *admin*)

NOTE: There can be no null constructor defined for this class.

Stock (sublcass of Security)

Attributes

private exchange: String	The exchange on which the stock trades (e.g.: NASDAQ, Nikkei, etc.)
private dividends: boolean	Indicates whether or not the stock pays dvidends: true = dividends paid; false = no dividends paid
private divDate: int	Indicates when dividends are paid by quarter; allowed values are 0 (if dividends = false) through 4
private divAmount: double	Dollar value of dividents/share; 0 if dividends = false and/or divDate is invalid

Operations

public MutualFund(nbr: String, date: int, prc: double, qty: double, sym: String, xchg: String, div: boolean, divDt: int, amt: double): Stock	Full constructor
public toString( ): String	Returns a formatted String object describing this Stock object: This Stock is traded on the exchange. This stock does not pay a dividend. [1] A dividend of $divAmount will be payed in the divDate quarter.[2] (1) If dividends = false (2) If dividends = true
public calcCost( ): double	Cost is calculated as: (shares * (purchPrc - divAmount)

NOTE: There can be no null constructor defined for this class.