Assignment 1

Due date: October 9th, 2017
Total points: 30

You will write three Java classes (not counting the enum types): Transaction, Security, and InvalidTransactionException for investment management, as we described in the class. The classes must follow the APIs given below:

Transaction: implements Comparable<Transaction>
public Transaction(TransactionType type, String price, String quantity, LocalDate date) throws InvalidTransactionException, NumberFormatException // Throw InvalidTransactionException when price or quantity is less than 0, or when the quantity is not 0 and the type is INC_SHARE. The NumberFormatException is thown by the BigDecimal constructor.
public Transaction(int id, TransactionType type, String price, String quantity, LocalDate date) throws InvalidTransactionException, NumberFormatException // Same as above
public int getId()
public TransactionType getType()
public BigDecimal getPrice()
public BigDecimal getQuantity()
public BigDecimal getTotal() // price*quantity
public LocalDate getDate()
public String toString() // A string containing the date, type, price, quantity, and total of the transaction.
* note: For bonuses, the quantity is the total number of shares that you hold, and the price is the bonus per share. For an increase of shares, the price is 0.
* note: Throwing InvalidTransactionException and NumberFormatException here is for practice. In a real situation, I would let the application layer do the parsing and checking, and would directly use BigDecimal for the constructor with the assumption that price and quantity are nonnegative.

Security:
public Security (String name, int id, SecurityType type)
public String getName()
public int getId()
public SecurityType getType()
public BigDecimal getQuantity()
public boolean isActive() // True if quantity > 0; false if quantity = 0.
public double getCurrentPrice()
public void setCurrentPrice(BigDecimal price) // If price < 0, do not update; clarify this point in the Javadoc.
public List<Transaction> getTransactions()
public String toString() // A (multiline) string with all the properties of the security in 1 line and all its transactions line by line, in chronical order.

public BigDecimal getCurrentValue() // current price*quantity
public void addTransaction (Transaction t) throws InvalidTransactionException // Add a transaction to this security, make corresponding changes to the properties. Prevent adding the transaction and throw an exception if the quantity would become negative after selling.
public BigDecimal floatingProfit() // Calculate the profit of a security based on the current price.
public BigDecimal cumulativeReturn() // Calculate the simple profit rate for you investment in this security, defined as (current value + total gain) / total investment, then minus 1.
public BigDecimal annualizedReturn() // Calculate the annualized profit rate for you investment in this security, defined as the solution to the following equation: $\sum_{t.type=BUY} t.total * (1+x)^{t.yearsPassed} = CurrentValue + \sum_{t.type\in\{SELL,BONUS\}} t.total * (1+x)^{t.yearsPassed}$.

* note: When doing complex computations like in annualizedReturn(), it is better to convert the numbers to double, do the computation, and convert the result back to BigDemical for the output. Objects are very inefficient for computation.

InvalidTransactionException extends Exception
public InvalidTransactionException(TransactionType type, String price,
String quantity, LocalDate date, String errorMessage)
public InvalidTransactionException(String securityName, TransactionType type,
String price, String quantity, LocalDate date, String errorMessage)
public String getSecurityName()
public TransactionType getType()
public LocalDate getDate()
public String getPrice()
public String getQuantity()

public enum TransactionType {BUY, SELL, BONUS, INC_SHARE}
public enum SecurityType {STOCK, FUND}

The classes should be packed in the package 'cn.edu.tsinghua.stat.investment'. We will use another class (not part of your package) to call the APIs of your classes to test their functionality and correctness. Therefore, if you don't follow the APIs exactly, the program will not compile, and you will lose a significant portion of the score. We will also check if your objects are well encapsulated, have proper unit tests and Javadocs. Note that since we will call getType() from outside of the package, the enum definition must be public.

Submit all the source code (classes, exceptions, enum types) written in good coding style. We will put them in the subdirectory 'cn/edu/tsinghua/stat/investment/' and compile them locally. Do not submit the class files.