15-100 Exam #1
Spring 2005
Kesden/Afternoon Edition

1. Please write a Java class specification that specifies a type as described below [90 points]:

    a. You are describing an *TransactionSummary*.

    b. The *totalAmountInDollars* involved, which is normally positive, but can be negative in the event of a refund. It measures the amount of money in dollars, including cents as decimal fractions thereof.

    c. A flag to indicate that the transaction *isARefund*. Otherwise, it is presumed not to be a refund.

    d. A *changeCounter,* initially 0, that indicates the total number of times the value changes (increases or decreases) after initialization.

    e. All of the above attributes must be set upon initialization
        i. IMPORTANT NOTE: The transaction is a refund if, and only if, the initial *amountOfMoney* is negative. It is a sale otherwise.

    f. Each of the above attributes should be individually accessible.

    g. It should be possible to *increaseTheValue* of a transaction. The specified *changeInValue* should always be expressed as a non-negative dollar amount. It should increment the *changeCounter* and return *true* if successful. It should return *false*, without affecting the *changeCounter,* if the *changeInValue* is negative.

    h. It should be possible to *decreaseTheValue* of a transaction. The specified *changeInValue* should always be expressed as a non-negative dollar amount. It should take no action and return *false* if the dollar *changeInValue* exceeds the *totalAmountInDollars* or if the *changeInValue* is negative. It should increment the *changeCounter* and return *true*, otherwise. .

    i. The *TransactionSummary* should be able to reduce itself to a *String* representation via the usual technique.

    j. The *TransactionSummary* should be able to *compareTo* another transaction summary to indicate if it is greater than, less than, or equal to the other transaction. This should be done in the usual way. The primary basis for the comparison is the *totalAmountInDollars* (Note: This is always positive, regardless of transaction type). In the event these are equal, the *changeCounter* should be used (higher is greater).

2. Please write a main() method, with the usual form, that creates a new *TransactionSummary*, with the attributes of your choice, and then prints it out. [5 points]

3. Please distinguish between *overloading* and *overriding* [5 points]

Rubric:

Important note to grader: Grade the thought process, structure, form, logic, understanding &c -- not the details. Focus on the *natural complexity* not the *arbitrary complexity*.

Question #1:

1. Unreasonable class name [1 point]
2. Unclear class structure [5 points]
3. Instance variables: *totalAmountInDollars*, *isARefund*, and *changeCounter*, deductions per each
    a. access specifier is *private* [1 point]
    b. type is correct [1 point]
    c. "extras" [1 point]
4. Constructors
    a. No "initialize everything" constructor [5 points]
    b. No "initialize without flag" constructor [5 points]
    c. Incorrect logic in second form of constructor [5 points]

4. Accessors, per each
    a. Not public [1 points]
    b. Parameters [1 points]
    c. Incorrect return type [1 points]
    d. Incorrect function [2 points]

5. *increaseTheValue*
    a. Not public [1 point]
    b. Wrong argument list[1 point]
    c. Return type is not boolean [1 point]
    d. Does not change count [1 points]
    e. No/bad "if logic" [3 points]

6. *decreaseTheValue*
    a. Not public [1 point]
    b. Wrong argument list[1 point]
    c. Return type is not boolean [1 point]
    d. Does not change count [1 points]
    e. No/bad "if logic" for negative value [3/2 points]
    f. No/bad "if logic" for "too high" amount [3/2 points]

7. *toString*
    a. Incorrect signature [2 points]
    b. Minor problem [2 points]
    c. Additional for major problem[2 points]

8. *compareTo*
    a. Incorrect signature [2 points]
    b. Problem with primary comparison [2 points]
    c. Problem with secondary comparison[2 points]

Question #2:

1. Incorrect signature [2 points]
2. Incorrect instance variable declaration [1 point]
3. Incorrect object instantiation [1 point]
4. Incorrect output [1 point]

Question #3:

1. Fails to demonstrate understanding of *overriding* [2 points]
2. Fails to demonstrate understanding of *overloading* [2 points]
3. Fails to clarify *distinction* between two terms [1 point]