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

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

    a. You are describing an *RetirementAccount*.

    b. The account has a *balance*, which can be positive or negative (ouch!) and measures the amount of money in dollars, including cents as a decimal fraction thereof.

    c. The account may be *taxDeferred*, or not.

    d. The account also has a *holder*, who owns the asset

    e. All of the above attributes must be set upon initialization
        i. IMPORTANT EXCEPTION: If no balance is specified, it should be initialized to $0.00

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

    g. It should be possible to *toggleDeferredStatus* such that a tax-deferred account becomes taxable and vice-versa.

    h. It should be possible to *contribute* additional money to the account – but not to withdraw money.
        i. If a negative contribution is specified, this method should return *false* to indicate the problem. Otherwise, it should return *true* to indicate the successful change.

    i. It should be possible to cause the account to *appreciate* (gain value), where the amount to increase in value is specified as a whole-number *percentage*.

    Negative appreciation indicates that the account has lost money. For example,
        i. $100 appreciating by 1% yields a balance of $101.
        ii. $100 appreciating by -2$ yields an account balance of $98.

    An account should lose its tax-deferred status if it appreciates by more than (positive) 10%.

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

    k. The *RetirementAccount* should be able to *compareTo* another retirement account to indicate if it is greater than, less than, or equal to that account. This should be done in the usual way.

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

3. The .equals() method can be invoked on any object – even if not defined within the object's class specification. Why? [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: *balance, taxDeferred*, and *holder*, deductions per each
    a. access specifier is *private* [1 point]
    b. type is correct [1 point]
    c. "extras" [1 point]
4. Constructors
    a. No "pass in everything" version [5 points]
    b. No "pass in everything but balance" version [5 points]
    c. No constructor at all, additional penalty [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. *toggleDeferredStatus*
    a. Not public [1 point]
    b. Parameters [1 point]
    c. Return type is not void [1 point]
    d. Incorrect function [2 points]

6. *contribute*
    a. Signature incorrect [3 points]
    b. if predicate incorrect [3 points]
    c. if structure not correct [2 points]
    d. Problem with return(s) [2 points]

7. *appreciate*
    a. Signature incorrect [3 points]
    b. if predicate incorrect [3 points]
    c. if structure not correct [2 points]
    d. Problem with computation [1 points]
    e. Problem with boolean operation [1 points]

8. *toString*
    a. Incorrect signature [3 points]
    b. Minor problem [2 points]
    c. Additional for major problem[5 points]

9. *compareTo*
    a. Incorrect signature [3 points]
    b. Minor problem [2 points]
    c. Additional for major problem[5 points]

    d. The *RetirementAccount* should be able to *compareTo* another retirement account to indicate if it is greater than, less than, or equal to that account. This should be done in the usual way.

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. No understanding of it being pre-defined [-5 points]
2. No specific understanding of inheritance [-3 points]
3. No specific understanding of overriding [-1 point]