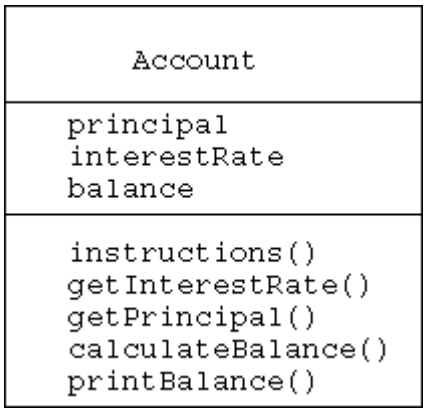
CS 150: Intro to OOP, Spring 2012
Assignment 5
Due via email by 11:59pm on March 7, 2012

For this assignment you must work individually on this program.

You will be producing a single program as specified in this assignment. The program will consists of a class definition and a main part which will use the class.

The application is a bank account which will draw interest. You will accept principal and interest rate from the user and produce output for yearly balance.

The class is to have three attributes and five methods as shown:



The instructions() method is to display this:

```
Welcome to the Balance Calculator
----------------------------------------
You will be asked to enter an interest rate, followed by a current principal.
The end of year balance will be computed and displayed.
You will be invited to execute this program until you specify that you would like
to terminate.
```

The getInterestRate() method is to prompt the user for an interest rate number. The interestRate attribute is to store this value as a raw interest rate, that is, if the interest rate is 5% then interestRate = 0.05. How you ask for this from the user is your choice, but it must be clear to the user what is to be entered. This number is to be validated. You must accept fractional interest rates (but no more then 2 digits), for example, 0.11% (interestRate = 0.0011) and disallow rates larger than 15%.

The getPrincipal() method is to prompt the user for a principal. The principal can be accepted as a number in dollars and cents, for example, 11.34 (eleven dollars and thirty-four cents), but do not accept fractional cents and do not accept a negative principal.

The calculateBalance() method is to calculate the balance using the formula:

balance = principal + (principal * interestRate)

The printBalance() method is to print the details of the account. For example, if an account has a principle of $1000 and an interest rate of 2%, then the output could show:

```
A principal of $1000.0 at an interest rate of 2% yields a yearly balance of $1020.0
```

Do not print more than 2 digits to the right of the decimal point.

The main program shall instantiate an Account object, then call each of the methods in the order presented. It should then prompt the user if another account is desired, if so then repeat, until the user declines. An example session might be:

```
Enter interest rate in %: 3.1
Enter principal: 100
A principal of $100.0 at an interest rate of 3.1% yields a yearly balance of $103.1

Another account? y
Enter interest rate in %: 0.12
Enter principal: 10.01
A principal of $10.01 at an interest rate of 0.12% yields a yearly balance of
$10.02

Another account? n
```

User input is in bold.

Extra credit: Frequently monetary data and computation is done in what is called fixed-point arithmetic. This is done because the inexact values of binary floating-point numbers are a problem in producing exact balances across a large number of computations. Python does not offer this data type, but it can be simulated using integer/long arithmetic. In this case, we are interested in dollar values with a maximum of 4 places to the right of the decimal. This is driven by our allowing interest rate to be as small as 0.0001. Thus, we would store dollar values in units of 0.0001 dollars or hundredths of cents. For example, if the users enters a principal of 100 dollars, that would be stored as 1000000 (100*10000) and an interest rate of 5% that would be stored as 500 (.05*10000), yielding a balance of 1050000 (1000000 + 1000000 * 500 / 10000, the multiplication adds 4 more digits of precision which need to be divided off). For printing, two digits are thrown away, or better rounded away, yielding a number in cents (10500), then a decimal point inserted to print dollars and cents (105.00).