CSc 227 — Program Design and Development
Spring 2014 (McCann)

http://www.cs.arizona.edu/classes/cs227/spring14/

Program #1: Your First Java Programs

Due Date: January 28th, 2014, at 9:00 p.m. MST

Overview: Often the hardest part of learning a new programming language is learning how to convince the computer to accept, run, and show you the result of your first program. That's why we're starting simple: In this assignment you'll just type in one simple program, and will write another simple program from scratch. At the same time, this assignment will give you some experience using the computers in our lab(s), and Piazza.

Assignment: This assignment has three parts: A Piazza check-in, the type-in exercise, and the write-a-program-from-scratch exercise.

(a) **Piazza:** As mentioned in class and on the syllabus, Piazza.com is hosting our discussion board for this class. It stands to reason that you are more likely to use it if you know how, so: Log into Piazza (after registering yourself if necessary) and post a not-anonymous answer to the question titled "The Program 1 Question." That's it! (But feel free to poke around Piazza while you're there.)

(b) **Falling Object:** The equation $t = \sqrt{\frac{h}{\frac{1}{2}a}}$ takes a height (h) above ground (in feet) and the acceleration due to gravity (a, about 32.174 feet per second squared) and determines the time (in seconds) required for a dropped object to hit the ground (assuming we're on Earth and air resistance is not a concern). Once we know the time, $s = at$ tells us the speed (s) at impact (in feet per second).

On the last page of this handout is a complete Java program that determines the time to impact and speed at impact of a dropped object. Here's what you are to do with it:

  (a) Visit the Gould-Simpson 228 lab, preferably during a time when a section leader is on duty. (That's so you can get help if you need it.) If the door isn't open, use your CatCard to unlock it.
  (b) Using what you learned in section the first week of classes, log in to a Linux machine.
  (c) Using an editor or IDE of your choice (e.g., vim, jGrasp, nano, Eclipse, emacs, . . .), and following the steps I demonstrated in class, create a file named `Prog1a.java` (the capitalization is important) and type in the Falling Object program you'll find on the last page of this handout, exactly as you see it.[1]
  (d) Compile (from the command line: `javac Prog1a.java`) and run (`java Prog1a`) the program to make sure it's working correctly.
  (e) If you aren't already, log onto `lectura.cs.arizona.edu`. (You've got to have your file in your CS account and be logged into lectura for this step to work.) From the command line, use `turnin` to submit your program. The command is: '`turnin cs227p01 Prog1a.java`' (w/o the quotes).
  (f) If `turnin` worked, you will see a confirmation message. To be sure you turned in the correct file, use the `-ls` (that's a lower-case 'L', not a one) flag with `turnin` (that is: `turnin -ls cs227p01`) to see a list of the files you have sumbitted for this assignment.

(c) **Compound Interest:** Having completed that first program, you'll be ready for this one: Write a complete Java program named `Prog1b.java` (with documentation similar to what you see in the Falling Object program) that computes the future value of a savings account the user opens on his or her next birthday, and also tells the user how much money they would cost themselves if they wait just five years to open the same account. We'll also assume that the user will need the money on his or her 70th birthday, and that the interest on the savings account is compounded daily (that is, there are 365 compounding periods per year).

[1]Of course, insert the names, as directed. Also, details such as the quantity of equal signs in the block comment need not match exactly.

To compute these future values, the user needs to tell the program three numbers: (1) Their age, in years, on their next birthday; (2) the amount to be invested (the *principal*); and (3) the savings account's *annual percentage yield*, or *APY*, as a percent.

There are two formulae your program needs to use:

(a) The value (V) of the invested principal (p) after y years in a savings account whose interest is compounded daily ($c = 365$) is computed using this formula:

$$V = p\left(1 + \frac{\text{APR}}{c}\right)^{cy}$$

(b) If you were reading carefully, you noticed that the first formula uses APR, not APY, to compute the value of the investment. The APY is what a bank will usually give you as the interest rate on the savings account. The APR is the *annualized percentage rate*, also known as the *nominal* or *per annum interest rate*. The APR is the actual interest rate of the account. The APY is the APR adjusted to reflect the effect of compound interest. The APY is a bit larger that the APR, which is why the banks advertise it.

To compute the APR from the APY, use this formula:

$$\text{APR} = c\left((1 + \text{APY})^{\frac{1}{c}} - 1\right)$$

where c is again 365 for this program, and APY is the given percentage in decimal (e.g., 5% in decimal is 0.05).

Both formulae raise values to powers. To compute $(b)^e$ in Java, use `Math.pow(b,e)`. `pow()` is a method of the `Math` class of Java's API, as is the `sqrt()` method you used in `Prog1a.java`.

Example: Let's say that the user will be 25 on her next birthday, and will have \$10,000 to invest in a savings account that has an APY of 5%.[2]

That savings account's APR is $365 * ((1 + 0.05)^{\frac{1}{365}} - 1) \approx 0.04879$, or 4.879%.

When the user turns 70 (in $70 - 25 = 45$ years), her investment will be worth $10000 * \left(1 + \frac{0.04879}{365}\right)^{365*45} \approx$ \$89,836.23.

If she waits five years to make that investment, the value will be worth just $10000 * \left(1 + \frac{0.04879}{365}\right)^{365*40} \approx$ \$70,390.24, close to \$20,000 less!

Here's what your program's output should look like, for that same data:

```
Imagine that you make an investment on your next birthday.
This program will compute the value of that investment
when you turn 70.0, and will tell you the amount
you will lose by waiting just 5.0 years to make the investment.

How many years old will you be on your next birthday?: 25
Enter the amount, in dollars, to be invested: 10000
Enter the investment's APY as a percent (ex: 3.5): 5


Be aware:  Your investment's APR is just 4.879342524642616%.

At age 25.0, an investment of $10000.0,
compounded 365.0 times per year using an APY of 5.0,
will be worth $89850.07793501065 when you turn 70.0 years old.

If you wait just 5.0 years before investing the same amount,
```

[2]Good luck finding that APY today!

```
    it will be worth only $70399.88712130398 at age 70.0,
    a difference of $19450.190813706664.
```

Yes, the numbers are sometimes really long. That's because all of the values in this program are of type `double` (as they should be in your program). It is possible to round off the values for output purposes, but we don't yet know how. Also, the results from the program are a few dollars more than those of the example. That is because the example uses a truncated APR. Your program should display very nearly, if not exactly, the same result as given in this sample output.

When you have `Prog1b.java` completed and you believe it to be working as it should, use `turnin` on `lectura` to submit it to the `cs227p01` folder.

Turn In: As mentioned with each part, above, you need to have your files on `lectura` **and** you have to use `turnin` to submit them to us for grading. You can do this whenever you're ready. (Just be sure to do so before the due date and time so that you don't lose a late day!)

A note about `turnin`: You can submit a file as many times as you'd care to submit it; only the newest submission is stored. This means that you can turn in a program early, and later discover a bug, fix it, and resubmit without causing any trouble whatsoever.

Grading Criteria: We won't usually have grading criteria ready in advance for each assignment, but here's some for this one, just to give you a rough idea of what we're looking for:

Points	
5	Piazza question answered
15	Falling Object program entered
5	Falling Object program submitted using `turnin`
20	Interest program documentation and coding style
15	Interest program input prompting is clear and correct
15	Interest program calculations performed correctly
15	Interest program output formatted as is the sample output
10	Interest program submitted using `turnin`
100	Total Possible

For an assignment this straight-forward, we expect everyone to get a perfect score!

Other Requirements and Hints:

- Be sure to take the time to verify the output of your programs using muliple sets of sample data. Your SL will be executing your programs, looking for common errors that would cause your program to, for example, compute the savings account's final value incorrectly. You can check your program's output using a hand calculator ... and you should!
- Remember that late programs will be accepted, but at a cost of some/all of your late days, and/or big portions of your score. The policies are detailed in the class syllabus. Please plan ahead so that you don't have to use any late days on this first assignment!
- Also remember that this is an *individual* assignment; write your own compund interest program. But, do feel free to help each other figure out how to use an editor, how to use the `turnin` program, how to interpret Java error messages, etc.
- Last, but certainly not least: `Start early!` There are lots of little things that need to be done correctly to complete this assignment, and any one of them could trip you up. You are not likely to be able to complete all of them in time if you wait until an hour before the due date to get started.

(Continued...)

Here is the Falling Object program that we want you to type in as the second part of this assignment. **Other than replacing the names, type this in just as you see it here – blank lines and all!**

```
/*=============================================================================
 |   Assignment:  Program #1(a): Falling Object
 |       Author:  REPLACE THIS WITH YOUR NAME
 | Sect. Leader:  REPLACE THIS WITH YOUR SL'S NAME
 |
 |       Course:  CSc 227
 |   Instructor:  L. McCann
 |     Due Date:  January 28, 2014, by 9:00 p.m. MST
 |
 |  Description:  Given a height (in feet) entered by the user, this
 |                program determines how long (in seconds) it will take an
 |                object dropped from that height to hit the ground, using
 |                the kinematic equation t = sqrt(height / (0.5 * 32.174)),
 |                where 32.174 is the average acceleration due to gravity
 |                on Earth.  In addition, it will report the object's speed
 |                (s = 32.174 * t) at the time of impact.
 |
 | Deficiencies:  None known; this program meets specifications.
 *===========================================================================*/

import java.util.*;   // Java API's "util" package has the Scanner class

public class Prog1a
{

    public static void main (String [] args)
    {

        final double ACCEL_GRAVITY = 32.174;  // average acceleration due to
                                              // gravity (in ft/sec^2)

        double  height,    // height (in feet) from which object is dropped
                time,      // duration of the object's fall (in seconds)
                speed;     // speed (in feet per second) at time of impact
        Scanner keyboard;  // to reference Scanner object for keyboard input

        keyboard = new Scanner (System.in);

        System.out.println("\nThis program will determine how long it will "
                         + "take for an object dropped\nfrom a given height "
                         + "(in feet) to hit the ground here on Earth, and "
                         + "how fast\nit will be traveling when it hits.\n");

        System.out.print("How high (in feet) is the object?: ");
        height = keyboard.nextDouble();

        time = Math.sqrt(height / (0.5 * ACCEL_GRAVITY));
        speed = ACCEL_GRAVITY * time;

        System.out.println("\nIf you drop an object from "
                         + height + " feet, it will hit the ground\n"
                         + "in approximately " + time + " seconds "
                         + "at a speed\nof approximately "
                         + speed + " feet per second.\n");

    } // main

} // class FallTime
```