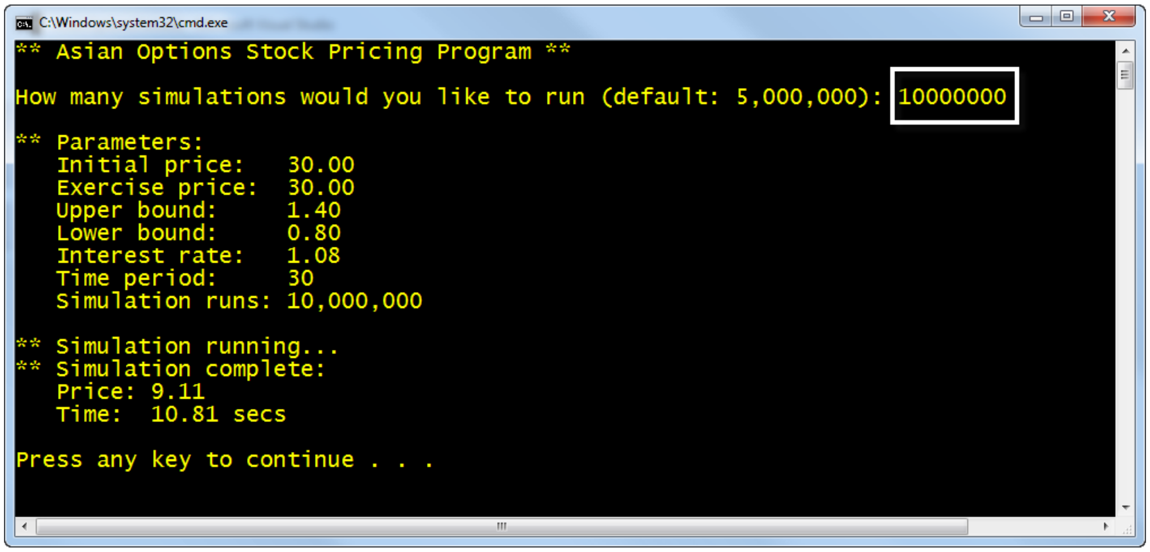
CS 341
Programming Language Design and Implementation
Fall 2015

Homework #7

Complete By: Friday, October 23rd @ 11:59pm
Assignment: completion of following exercise
Policy: Individual work only, late work *is* accepted
Submission: electronic submission via Blackboard

Background

The assignment is to take an existing, console-based C# program and create an equivalent but more flexible GUI-based C# program. The program as given performs ***Asian Options Stock Pricing***[1], using Monte Carlo simulations (aka coin-flipping) to predict the market. This type of approach is surprisingly accurate if the number of simulation runs is large. Here's an example run:



```
** Asian Options Stock Pricing Program **

How many simulations would you like to run (default: 5,000,000): 10000000

** Parameters:
   Initial price:   30.00
   Exercise price:  30.00
   Upper bound:     1.40
   Lower bound:     0.80
   Interest rate:   1.08
   Time period:     30
   Simulation runs: 10,000,000

** Simulation running...
** Simulation complete:
   Price: 9.11
   Time:  10.81 secs

Press any key to continue . . .
```

We are providing the source code that performs the simulation; your job is to build a graphical user interface to run simulations. The console-based program consists of 2 source code files: *Program.cs* and *AsianOptionsPricing.cs*. The latter performs a simulation run; the former shows you how to call the Simulation function. These files can be viewed from the course web page: Lectures\hw7-files. [*The files are provided as .txt files so you can simply open, copy, and paste*.]

[1] See http://en.wikipedia.org/wiki/Asian_option for more info.

Getting Started

Build a console-based application using the provided code so you see how the application is supposed to behave. To construct a console-based app, do the following:

1. Create a new project in Visual Studio: C#, Windows, console application
2. A program template is generated — delete the Main() function
3. Open the provided "Program.cs.txt" and copy-paste the Main() function
4. Add a new class to the C# program: Project menu, Add class…
5. Delete the pre-generated class code
6. Open the provided "*AsianOptionsPricing*.cs.txt" and copy-paste into the new class file

At this point you should be able to build and run the console app. Each simulation run may yield different results because the program is based on a random number generator that behaves randomly by default. This makes it difficult to compare different implementations for correctness. When you want predictable behavior (e.g. when comparing the output from the console app to that of the GUI app), seed the random number generator at the top of the **Simulation( )** function with a known value, like this:

```
Random rand = new Random( 123 );  // seed with value 123:
```

Now each time you run, you'll get the same result.

Assignment Details

The assignment is to build a GUI front-end to the provided simulation code. Note that you cannot convert the console-based app to a Windows Form app, you have to create a new WinForms-based project from scratch. However, reuse the provided *AsianOptionsPricing.cs* file exactly as is, and build a GUI-based front-end that allows the user to input the same values and perform the same simulations as the console-based app. In other words, the user should be able to run multiple simulations, providing inputs such as # of simulation runs, initial prices, time period, etc.

The program must be written in C#, and you must build a GUI-based application that yields the same results as the console-based program. The GUI can be simple or not, but the user interface must meet these requirements:

- allow the user to change the number of simulations to run
- allow the user to modify any of the simulation parameters (initial price, exercise price, etc.)
- provide reasonable defaults for each of the above values (use defaults from console app) — if you are using textboxes for the user to enter the values, this means the default values must be displayed in the textbox initially
- display simulation result — price and elapsed time
- allow the user to run as many simulations as he/she wants

You may assume the user will provide valid input; input validation is not required. You will need to convert the user input into numeric values; assuming s is a string, use System.Convert.ToDouble(s) to convert to double and System.Convert.ToInt64( ) to convert to long.

If you want to display a "wait" cursor while the simulation is running, this can be done as follows. To change the cursor, do

```
this.Cursor = Cursors.WaitCursor;
```

To reset the cursor back to its default, do:

```
this.Cursor = Cursors.Default;
```

List boxes are a handy UI element for the display of string-based data.. To display a string s at the end of the list, do

```
this.listBox1.Items.Add(s);
```

To insert at the beginning of the list, call the **Insert** method and specify position 0.

Electronic Submission

The first step is to create a .zip / compressed folder of your *entire* Visual Studio project folder. Then, using Blackboard, submit this .zip / compressed file under the assignment "HW7". One of your C# source code files should include a header comment at the top along the lines of

```
//
// GUI-based Asian Options Stock Pricing program.
//
// <<YOUR NAME HERE>>
// U. of Illinois, Chicago
// CS341, Fall 2015
// Homework 7
//
```

Your program should be readable with proper indentation and naming conventions; commenting is expected where appropriate. You may submit as many times as you want before the due date, but we grade the last version submitted. This implies that if you submit a version before the due date and then another after the due date, we will grade the version submitted after the due date — we will *not* grade both and then give you the better grade. We grade the last one submitted. In general, do not submit after the due date unless you had a non-working program before the due date.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates" or Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at http://www.uic.edu/depts/dos/studentconductprocess.shtml .