**Aia Al-Qasab**

**T5616SN**

OBJECT-ORIENTED PROGRAMMING

VISUAL STUDIO CONSOLE APPLICATION

Report 1

Information Technology Degree

August 2017



VISUAL STUDIO CONSOLE APPLICATION

A console application, in the context of C#, is an application that takes input and displays output at a command line console with access to three basic data streams: standard input, standard output and standard error. A console application facilitates the reading and writing of characters from a console - either individually or as an entire line. The program structure of a console application facilitates a sequential execution flow between statements. Designed for the keyboard and display screen, a console application is drive by keyboard and system events generated by network connections and objects.

A console application designed for the following reasons:

* To provide a simple user interface for applications requiring little or no user interaction, such as samples for learning C# language features and command-line utility programs.
* Automated testing, which can reduce automation implementation resources.

Console applications developed in C# have one main entry point (static main method) of execution, which takes an optional array of parameters as its only argument for command-line parameter representation.  
The .NET Framework provides library classes to enable rapid console application development with output display capability in different formats. System. Console (a sealed class) is one of the main classes used in the development of console applications.

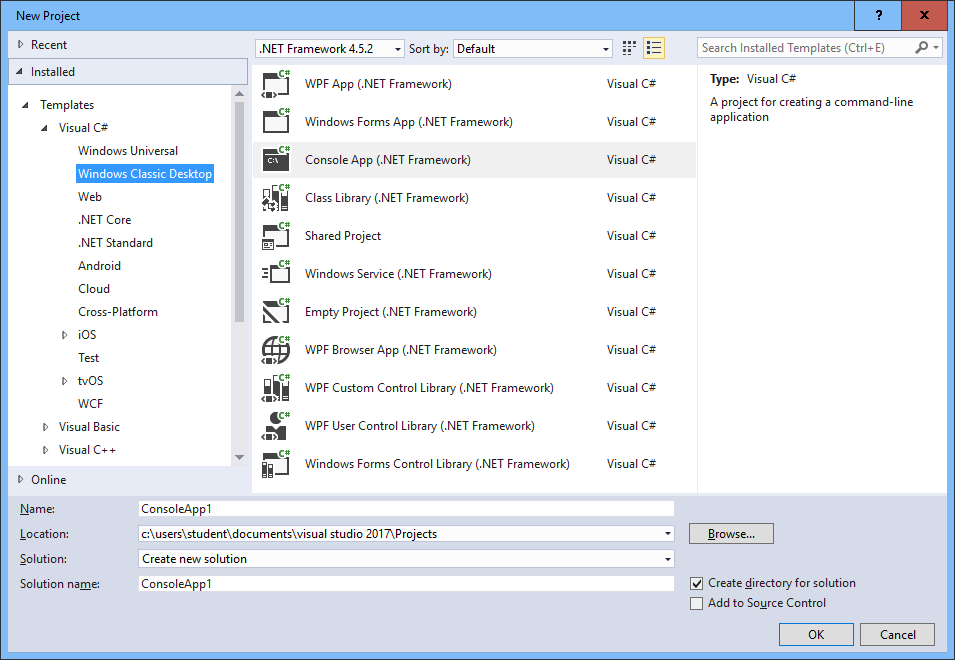
Console applications make you able to:

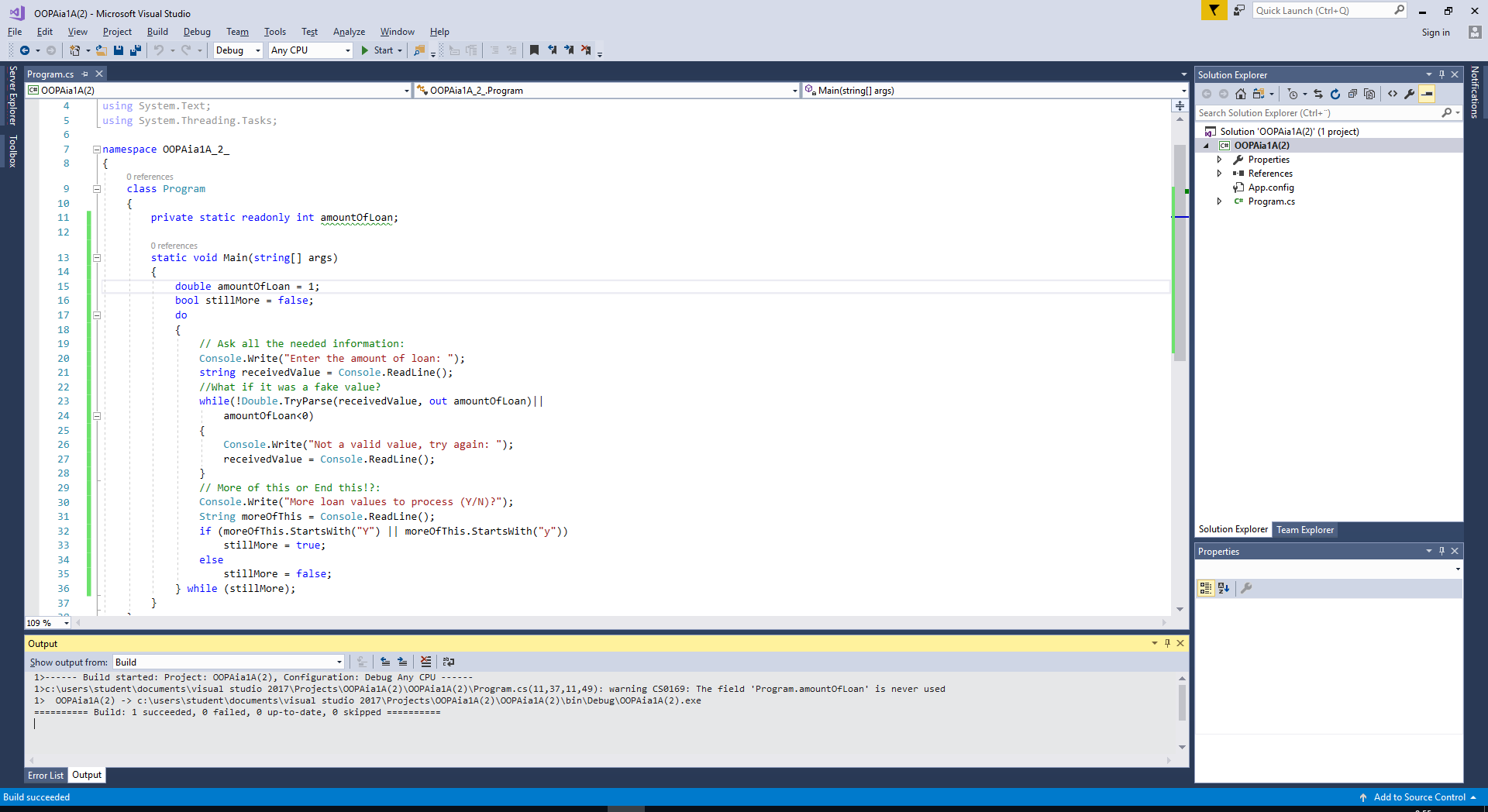
* Run the program, be greeted with a prompt, and then enter commands corresponding to various methods defined in a specific area of the application.
* Receive feedback, error messages, and such
* Easily add/remove commands
* Generally be able to put together a functioning console application, with predictable and familiar interactive behavior, without re-inventing the wheel every time.

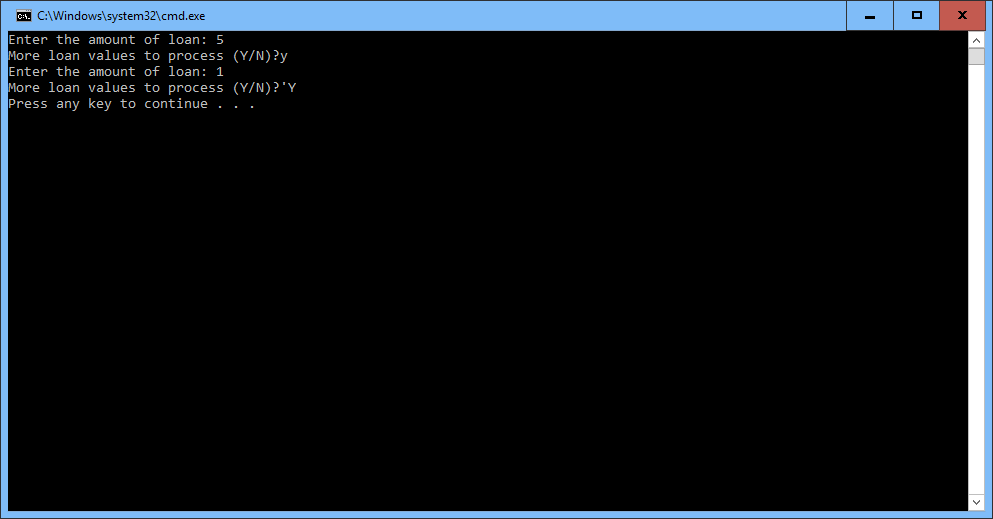
Before the console can do anything else, it needs to be able to take input from the user, and provide feedback where needed. We are all familiar with Console.Write() or Console.WriteLine(), and also Console.Read() and Console.ReadLine(). However, these methods in and of themselves are utilitarian at best, and don't quite get us to the interactive Input/output loop we are looking for.Also, the .NET Console does not directly afford us the ability to add a custom prompt.

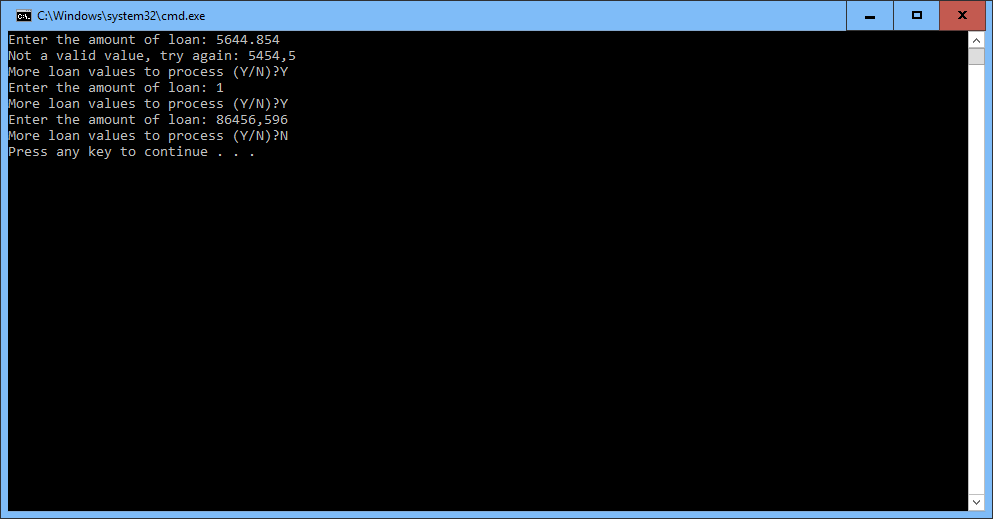
Once execution is in our Run() method, we begin to see how our basic interactive IO loop works. Before we look at Run() , however, let's look at the other two methods we see, at the bottom of the class. WriteToConsole() and ReadFromConsole() look suspiciously like Console.WriteLine() and Console.ReadLine() and in fact, we wrap those two methods within each of our own.

The primary purpose of the ReadFromConsole() method is to show a prompt, and collect user input. Note the use of Console.Write() as opposed to Console.WriteLine() here. This way, our custom prompt is displayed on the same line that input will occur.









Using Threading:  
  
