Use C# devenv to Program the Following :

1. Extend the code in lecture notes slide 7 lecture 2 to read and write any number of chars, make the GUI functional
2. Extend the ***filecopy*** program (slide 12 lecture 2), Add attractive GUI and Make it work.
3. The purpose of this exercise is to use the Read method in class Stream, and subclasses of class Stream. Write a variant of the file copy program. Your program should copy the entire file into a byte array. Instead of the method ***ReadByte*** you should use the ***Read*** method, which reads several bytes into a byte array. (Please take a careful look at the documentation of ***Read*** in class ***FileStream*** before you proceed). After this, write out the byte array to standard output such that you can assure yourself that the file is read correctly. Are you able to read the entire file with a single call to Read? Or do you prefer to read chunks of a certain (maximum) size? Which is better?
4. This example illustrates character encoding, and writing values of simple types

|  |  |
| --- | --- |
| **Writing a text string using three different encodings with StreamWriters.** | **Program: Writing values of simple types and objects of our own classes.** |
| 1. **using System;** 2. **using System.IO;** 3. **using System.Text;** 4. **public class TextWriterProg{** 5. **public static void Main(){** 6. **string str = "A æ u å æ ø i æ å",** 7. **strEquiv = "A \u00E6 u \u00E5 \u00E6 \u00F8 i \u00E6 \u00E5";** 9. **TextWriter** 10. **tw1 = new StreamWriter( // Iso-Latin-1** 11. **new FileStream("f-iso.txt", FileMode.Create),** 12. **Encoding.GetEncoding("iso-8859-1")),** 13. **tw2 = new StreamWriter( // UTF-8** 14. **new FileStream("f-utf8.txt", FileMode.Create),** 15. **new UTF8Encoding()),** 16. **tw3 = new StreamWriter( // UTF-16** 17. **new FileStream("f-utf16.txt", FileMode.Create),** 18. **new UnicodeEncoding());** 20. **tw1.WriteLine(str); tw1.WriteLine(strEquiv);** 21. **tw2.WriteLine(str); tw2.WriteLine(strEquiv);** 22. **tw3.WriteLine(str); tw3.WriteLine(strEquiv);** 23. **tw1.Close();** 24. **tw2.Close();** 25. **tw3.Close();** 26. **}** 27. **}** | **using System;**  **using System.IO;**  **public class TextSimpleTypes{**    **public static void Main(){**    **using(TextWriter tw = new StreamWriter("simple-types.txt")){**  **tw.Write(5); tw.WriteLine();**  **tw.Write(5.5); tw.WriteLine();**  **tw.Write(5555M); tw.WriteLine();**  **tw.Write(5==6); tw.WriteLine();**  **}**  **using(TextWriter twnst = new StreamWriter("non-simple-types.txt")){**  **twnst.Write(new Point(1,2)); twnst.WriteLine();**  **twnst.Write(new Die(6)); twnst.WriteLine();**  **}**  **}**  **}** |
| **Identify the output in both files** | |

1. Write a program that tosses a [Die](http://people.cs.aau.dk/~normark/oop-csharp/html/notes/source-programs/extra/user/normark/oop-csharp-1/sources/c-sharp/Die-variants/die-stuff-for-io-exercise/die.txt) 1000 times, and writes the outcome of the tosses to a text file. Use a TextWriter to accomplish the task.
   1. Use the following code to generate tosses

Random random = new Random(1);

for (int i = 0; i < 1000; i++)

Console.WriteLine("Toss outcome {0}", random.Next(1, 7));

1. Write another program that reads the text file. Report the number of ones, twos, threes, fours, fives, and sixes.
2. Use ***FileInfo*** and ***DirectoryInfo*** to make simple explorer (use ListView control in your GUI)
3. Any application have a set of configuration parameters that needs to be preserved even after the program is closed. There is no need to let user every time he opens the application to reassign those parameters. In this case the developer may create a class describing all the parameters and store the ***Conf\_Params*** in its attributes. Imagin that our configuration parameters are as follows:

ConnectionString = Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=control\_ITTU;Data Source=.\sql2008

TF\_EXCLUDE\_ID\_CAL\_GPA=True

TF\_EXCLUDE\_CAL\_GPA\_IDS\_VALs=10-19-20-12

FAC\_LAW\_SETTING=False

FAC\_EDU\_SETTING=False

TF\_SHOW\_GPA\_COL=True

TF\_SHOW\_CGPA\_COL=True

TF\_RATING\_ACCORDING\_CGPA=True

TF\_SHOW\_PERCENTAGE\_COL=False

TF\_PASS\_CONDITION=False

TF\_SHOW\_TOTAL\_COL=False

TF\_PASS\_CONDITIONTOT=False

TF\_RATING\_ACCORDING\_RATIO=False

ShowDays=True

FAC\_LAW\_SETTING\_AR=False

RLValue=30

FAC\_Tamreed\_SETTING\_SHAPES=True

* Create a class called ***Conf\_Params*** and add every configuration parameter as an attribute with suitable datatype.
* Create a clean GUI that enables user to change/display the value of every attribute and either to write/read those values to a file which correspond to the serialize/deserialize operations consequently.