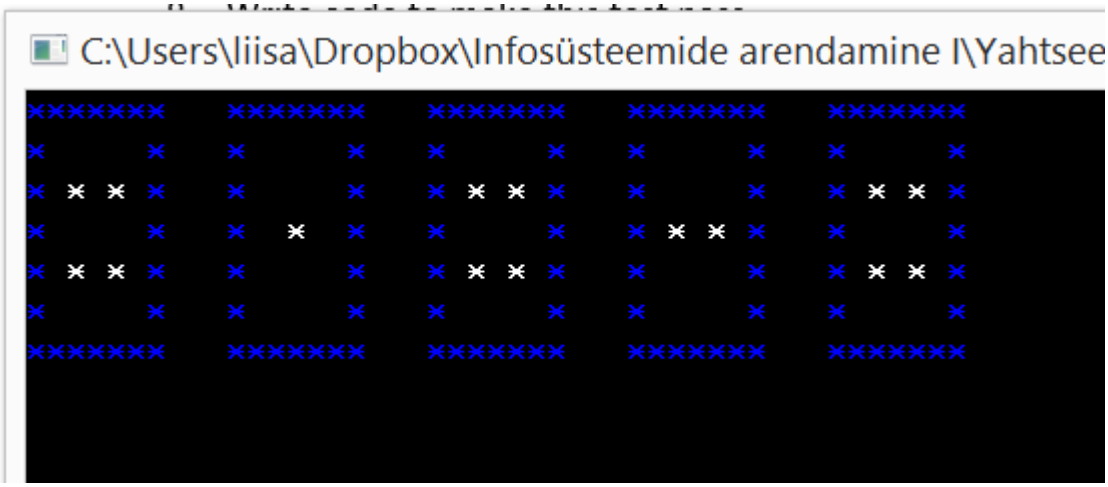
Exercise 1

Create a program for rolling a dice. The dice is displayed with a random number.

The dice is drawled in the console. When 'enter' key is pressed the dice should be rolled again and a new value displayed.

Example of how the dice should look like: (you only need one).



How to solve:

1. Draw an example of the dice on a squared paper and see how the indexes are.
2. Create a method for drawing the outer (blue) shape for the dice. The size should be 7x7 squares.
3. Then create a method for generating a random number and displaying it inside the dice.
4. Then create the shapes representing the numbers from 1-6 and replace the number with the visual.

It is possible to change the location of the cursor with:

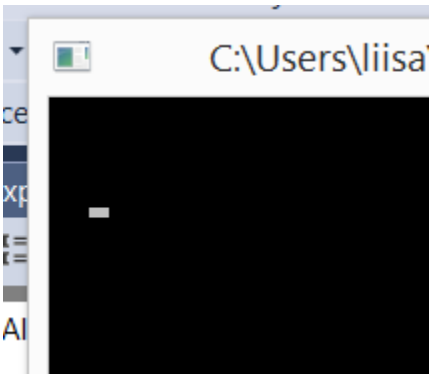
```
Console.CursorLeft = value;

Console.CursorTop = value;
```

Example:

Console.CursorLeft = 0; Console.CursorTop = 0; places the cursor in the top left of the screen.

Console.CursorLeft = 2; Console.CursorTop = 2; -> places the cursor 2 columns from left and 2 rows from top



Advanced: create 5 dices.

Exercise 2

New topics:

1. Enums : https://www.dotnetperls.com/enum NB! Enums are defined outside a class.
2. Get/set syntax: https://www.dotnetperls.com/property

We need to keep library for books.

Book has: title, genre (*žanr*) and status (borrowed out or in the library) and methods for getting and changing the values.

Available genres are: Horror, Science Fiction, Romance and Drama. Keep them in an enum.

Library can store one or more books. Library has:

- Method for adding book to the library (takes the book as a parameter)
- Method for finding book from the library by name (takes the name as a parameter)
- Method for finding all books of a certain genre. (takes the genre as a parameter).
- Method for borrowing out or returning a book to the library (takes the book as a parameter).

Create tests!

Example code:

```
public enum Difficulty { Easy, Normal, Hard }

class Task
{
    private Difficulty _difficulty; //private field

    private string _title; //private field

    public Difficulty CurrentDifficulty //public property
    {
```

```
            get { return _difficulty; } //returns the value
            set { _difficulty = value; } //sets the value
        }

        public string Title //public property
        {
            get { return _title; }
            set {
                    if (value != "") //we only set the value for title if its not an
empty string
                    {
                        _title = value;
                    }
                    else
                    {
                        Console.WriteLine("Invalid value!");
                    }
                }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Task task1 = new Task();
            task1.CurrentDifficulty = Difficulty.Easy;
            task1.Title = "Wash dishes";
        }
    }
```

Exercise 3.

Test Driven Development: first write an unit test, make sure the test fails and then write as little code as possible to make it work. Then write next test and repeat.

Fibonacci sequence: (we start from 1, not 0)

1, 1, 2, 3, 5, 8, 13, 21, ...

For example, calling Fib*(6)* would return 8.

Create a class public static method called Fib in the main class which takes number as a parameter. Create also test class.

1. Write a test for getting the first number(index 0), expected result is 1
2. Write code to make this test pass
3. Write a test for getting the second number (index 1), expected result is 1
4. Write code to make this test pass
5. Write test for getting the third number(index 2), expected result is 2
6. Write code to make this test pass
7. Write test for getting the fourth number(index 3), expected result is 3
8. Write code to make this test pass

9. Write a test for getting the fourth number (index 4), expected result is 4
10. Write code to make this test pass

Fib(0) = 1 (seems to be a defined output to begin the series)

Fib(1) = 1 (also seems to be defined)

Fib(2) = 1 + 1 = 2

Fib(3) = 2 + 1 = 3

Fib(4) = 3 + 2 = 5

Fib(n) = Fib(n-1) + Fib(n-2) ???

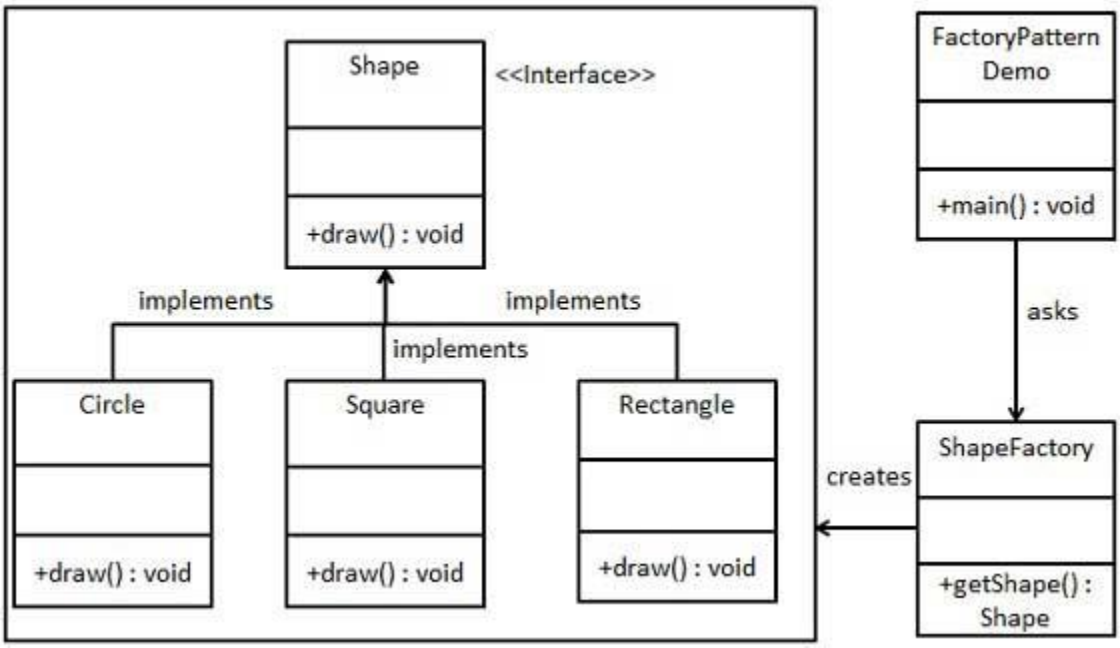
Exercise 4.

Design patterns. Read:

https://csharpdesignpatterns.codeplex.com/wikipage?title=Factory%20Method%20Pattern&referringTitle=Home

"In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface."

Task:



1. Create an interface for shape with one method "Draw"

2. Create concrete classes (Circle, Square, Retangle) and implement draw methods
3. Create a Factory to generate object of concrete class based on given information.
   Add method “getShape” which takes the shape to create as a parameter and creates the object then.
4. Use the Factory to get object of concrete class by passing an information such as type.