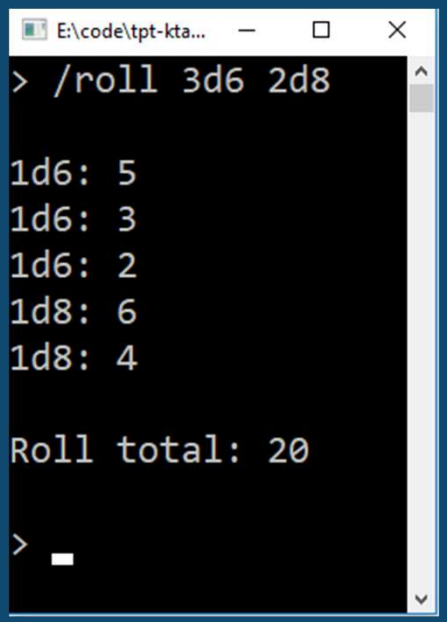
LET'S HARDCODE & PRINT OUT THE PROGRAM FLOW

What is the most simple program flow to illustrate the problem we are about to solve?

- Dungeons & Dragons rollimängus on suur osakaal täringutel. Täringuid on väga mitmete erinevate tahkude arvuga (4, 6, 8, 10, 12, 20). Tavaliselt lahingu käigus visatakse 20 tahulist täringut, et selgitada välja kas löök läheb pihta või mitte, ning seejärel üks või mitu erivat väiksematahulist täringut löögi tugevusena. Erinevate täringute kirjeldamiseks kasutatakse lühendeid d4, d6, d8, d10, d12 ja d20.

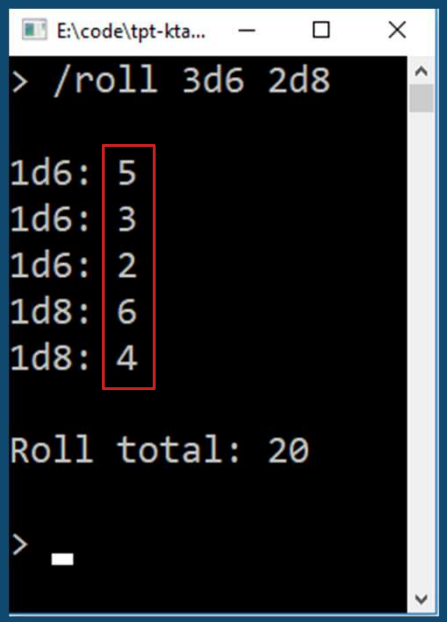
- Programmile peab olema võimalik anda käsklust stiilis /roll d20 mille korral kuvatakse arv vahemikus 1 - 20. Lisaks peab saama veeretada mitut täringut stiilis /roll d6 d8 d12, või siis stiilis /roll 2d6 1d8

PROBLEM STATEMENT



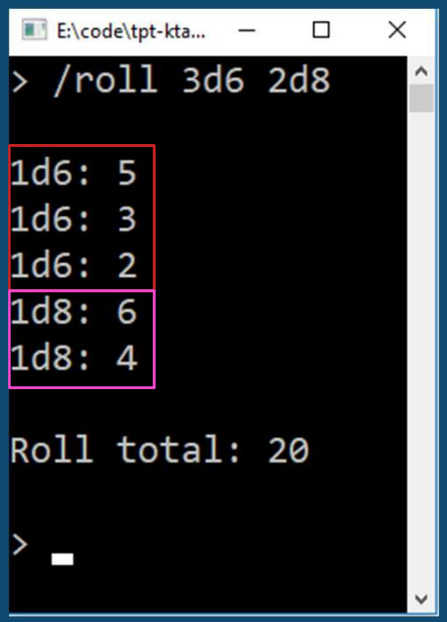
- Create new solution named „DiceRoller“
- Add new console project named „DiceRoller“ to the solution
- Print out text to match what is seen on the image
- **DO NOT ADD ANY BUSINESS LOGIC! JUST PRINT OUT THE TEXT!**

ASSIGNMENT #1



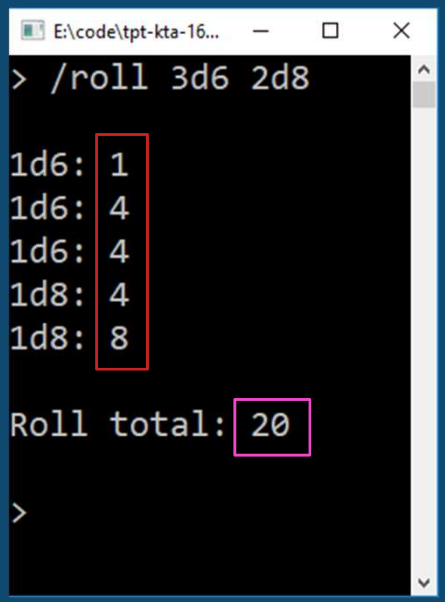
- Change values within **red box** to be generated randomly
- Use C# Random class to do this
- Don't worry about the „total: 20" part, we'll get to this soon :)

ASSIGNMENT #2



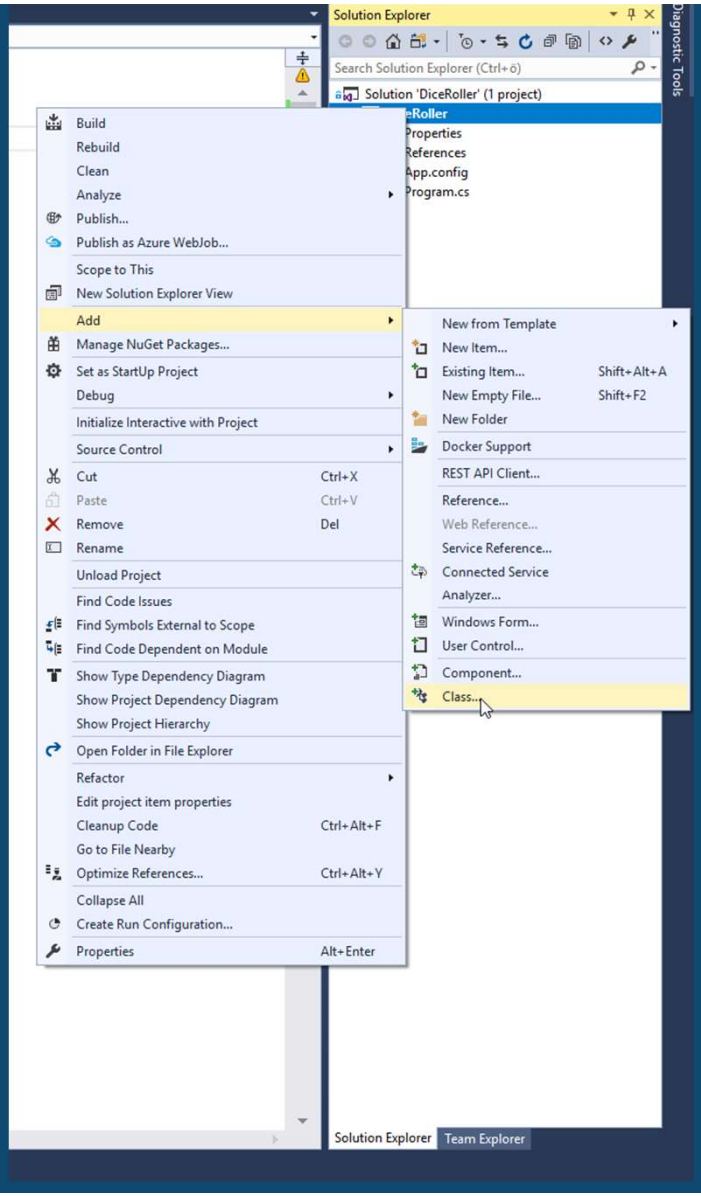
- Use two 'for' loops to remove duplicated code (Console.WriteLine...)
- One loop will print out 1d6 portion
- Another loop will print out 1d8 portion
- You should have only one 'Console.WriteLine...' in each 'for' block

ASSIGNMENT #3



- As you can see, sum of values do not equal the total
- Introduce ‘variable’ to keep track of individual dice rolls
- Print out the correct total

ASSIGNMENT #4



- Create new class named „Dice“

ASSIGNMENT #5

- Create ‘public’ method, named ‘Roll’, which returns a value of type ‘int’ and accepts number of sides as a parameter

- Return value must be random, between 1 and ‘number of sides’, specified by the parameter

ASSIGNMENT #6

- What we just created is a Dice **class**, but we need a Dice **object**
- To create a object, we need to 'new up' the corresponding **class**
- In C#, there is special keyword 'new' for this purpose

- Instantiate ('*new up*') the Dice class, to create a Dice object in 'Program.cs' file.
- Refactor (*change*) 'Program.cs' so, that there is **no logic for generating random numbers**

- **Hint:** dice.Roll(6) or dice.Roll(8)

ASSIGNMENT #7

```
1   namespace DiceRoller
2   {
3       using System;
4
        2 references | Kristjan Toots, 18 minutes ago | 1 author, 2 changes | 2 work items
5       public class Dice
6       {
            1 reference | 0 changes | 0 authors, 0 changes
7           private Random Random { get; } = new Random();
8
            2 references | Kristjan Toots, 18 minutes ago | 1 author, 1 change | 1 work item
9           public int Roll(int sides)
10          {
11              return this.Random.Next(1, sides + 1); // max value is exclusive, so we have to add +1
12          }
13      }
14  }
```

- We have to take random initialization out from the ‘Roll’ method beacuse we should initialize it only once; when the class is constructed

SOLUTION #8

```
0 references | Kristjan Toots, 13 minutes ago | 1 author, 5 changes | 5 work items
public static void Main(string[] args)
{
    Console.WriteLine("> /roll 3d6 2d8");
    Console.WriteLine();

    Dice d6 = new Dice(6);

    var total = 0;

    for (var i = 0; i < 3; i++)
    {
        var roll = d6.Roll();
        total += roll;

        Console.WriteLine($"1d6: { roll }");
    }

    Dice d8 = new Dice(8);
    for (var i = 0; i < 2; i++)
    {
        var roll = d8.Roll();
        total += roll;

        Console.WriteLine($"1d8: { roll }");
    }
```

- Change your code in 'Program.cs' to match the image
- Make **absolute minimum** changes in 'Dice.cs' to make your **code compile**
- It is okey if existing functionality breaks for a moment

ASSIGNMENT #9

- You need to store ‘int sides’ parameter in a class level variable, so that you can access it from the ‘*Roll()*’ method
- **Hint:** Google „C# auto-implemented properties“

- Add auto-implemented property of type ‘int’ called ‘Sides’ with ‘get’ accessor
- Assign constructor parameter ‘int sides’ value to ‘Sides’ property
- Use ‘Sides’ property in ‘*Roll()*’ method to restore previous functionality

ASSIGNMENT #10

```
11
12          Dice d6 = new Dice(6, "d6");
13
14          var total = 0;
15
16          for (var i = 0; i < 3; i++)
17          {
18              var roll = d6.Roll();
19              total += roll;
20
21              Console.WriteLine($"1{ d6.Description }: { roll }");
22          }
23
24          Dice d8 = new Dice(8, "d8");
25          for (var i = 0; i < 2; i++)
26          {
27              var roll = d8.Roll();
28              total += roll;
29
30              Console.WriteLine($"1{ d8.Description }: { roll }");
31          }
32
33
```

- Change your code in ‘Program.cs’ to math the image
- Add description parameter to the ‘Dice’ constructor
- Implement ‘get auto-implemented property’ for Description and assign its value

ASSIGNMENT #11

```
0 references | Kristjan Toots, 8 minutes ago | 1 author, 7 changes | 7 work items
public class Program
{
    0 references | Kristjan Toots, 8 minutes ago | 1 author, 7 changes | 7 work items
    public static void Main(string[] args)
    {
        Console.WriteLine("> /roll 3d6 2d8");
        Console.WriteLine();

        Dice d6 = Dice.D6;

        var total = 0;

        for (var i = 0; i < 3; i++)
        {
            var roll = d6.Roll();
            total += roll;

            Console.WriteLine($"1{ d6.Description }: { roll }");
        }

        Dice d8 = Dice.D8;
        for (var i = 0; i < 2; i++)
        {
            var roll = d8.Roll();
            total += roll;

            Console.WriteLine($"1{ d8.Description }: { roll }");
        }
```

```
{
    using System;

    9 references | Kristjan Toots, 9 minutes ago | 1 author, 6 changes | 6 work items
    public class Dice
    {
        2 references | Kristjan Toots, 22 minutes ago | 1 author, 1 change | 1 work item
        public int Sides { get; }

        3 references | Kristjan Toots, 9 minutes ago | 1 author, 1 change | 1 work item
        public string Description { get; }

        1 reference | Kristjan Toots, 1 hour ago | 1 author, 1 change | 1 work item
        private Random Random { get; } = new Random();

        2 references | Kristjan Toots, 9 minutes ago | 1 author, 1 change | 1 work item
        private Dice(int sides, string description)
        {
            this.Sides = sides;
            this.Description = description;
        }

        1 reference | 0 changes | 0 authors, 0 changes
        public static Dice D6 => new Dice(6, "d6");

        1 reference | 0 changes | 0 authors, 0 changes
        public static Dice D8 => new Dice(6, "d8");

        2 references | Kristjan Toots, 22 minutes ago | 1 author, 2 changes | 2 work items
        public int Roll()
        {
            return this.Random.Next(1, this.Sides + 1); //
```

SOLUTION #12

```
0 references | Kristjan Toots, 30 minutes ago | 1 author, 8 changes | 8 work items
6   public class Program
7   {
        0 references | Kristjan Toots, 30 minutes ago | 1 author, 8 changes | 8 work items
8       public static void Main(string[] args)
9       {
10          Console.WriteLine("> /roll 3d6 2d8");
11          Console.WriteLine();
12
13          DiceRoller diceRoller = new DiceRoller();
14          List<DiceRoll> diceRolls = diceRoller.Roll(
15              new List<Dice> { Dice.D6, Dice.D6, Dice.D6, Dice.D8, Dice.D8 });
16
17          |
18          /*
19          Dice d6 = Dice.D6;
20
```

- Change your code in ‘Program.cs’ to match the image
- Make this code to compile
  - Create ‘DiceRoller’ and ‘DiceRoll’ classes. Implement ‘*Roll()*’ method just **enough to compile**

ASSIGNMENT #13

```
0 references | Kristjan Toots, Less than 5 minutes ago | 1 author, 9 changes | 9 work items
public class Program
{
    0 references | Kristjan Toots, Less than 5 minutes ago | 1 author, 9 changes | 9 work items
    public static void Main(string[] args)
    {
        Console.WriteLine("> /roll 3d6 2d8");
        Console.WriteLine();

        DiceRoller diceRoller = new DiceRoller();
        List<DiceRoll> diceRolls = diceRoller.Roll(
            new List<Dice> { Dice.D6, Dice.D6, Dice.D6, Dice.D8, Dice.D8 });

        foreach (var diceRoll in diceRolls)
        {
            Console.WriteLine($"1{diceRoll.Dice}: {diceRoll.Value}");
        }
```

- Change your code in ‘Program.cs’ to match the image
- Make this code to compile
  - Create ‘Dice’ and ‘Value’ public auto-implemented properties. Do just **enough** to make this **to compile**!

ASSIGNMENT #14

```
DiceRoller diceRoller = new DiceRoller();
List<DiceRoll> diceRolls = diceRoller.Roll(
    new List<Dice> { Dice.D6, Dice.D6, Dice.D6, Dice.D8, Dice.D8 });

foreach (var diceRoll in diceRolls)
{
    Console.WriteLine($"1{diceRoll.Dice}: {diceRoll.Value}");
}

Console.WriteLine();
Console.WriteLine($"Roll total: { diceRolls.Sum(x => x.Value) }");
Console.WriteLine();
Console.Write("> ");
```

THIS IS LINQ – LANGUAGE INTEGRATED QUERY (SOLUTION #15)

```
2 references | Kristjan Toots, 54 minutes ago | 1 author, 2 changes
public class DiceRoller
{
    1 reference | Kristjan Toots, 54 minutes ago | 1 author, 2 changes
    public List<DiceRoll> Roll(List<Dice> dices)
    {
        var result = new List<DiceRoll>();

        foreach (var dice in dices)
        {
            result.Add(new DiceRoll(dice.Description, dice.Roll()));
        }

        return result;
    }
}
```

```
3 references | Kristjan Toots, 53 minutes ago | 1 author, 3 changes
public class DiceRoll
{
    2 references | Kristjan Toots, 1 hour ago | 1 author, 1 change
    public string Dice { get; }

    3 references | Kristjan Toots, 1 hour ago | 1 author, 1 change
    public int Value { get; }

    0 references | Kristjan Toots, 53 minutes ago | 1 author, 1 change
    public DiceRoll(string dice, int value)
    {
        this.Dice = dice;
        this.Value = value;
    }
}
```

WHAT-EVER IS NEEDED TO MAKE OBJECT WORK – WE SHOULD GIVE IT WHEN WE ARE CREATING IT (SOLUTION #16)

```
13 references | Kristjan Toots, 1 hour ago | 1 author, 4 changes
public class Dice
{
    2 references | Kristjan Toots, 1 hour ago | 1 author, 1 change
    public string Description { get; }

    private static readonly Random Random = new Random();
```

Make the Random field static, so the same instance is shared across all instances of Dices

THE FIX (SOLUTION #17)