[0750] Exceptions and POJOs - A Match Made in Heaven

Exceptions are perfect for POJOs and other encapsulation classes that protect data because they provide a way for you to send back errors if something tries to violate your encapsulation rules.

So far, I've sort of glossed over what you should do if someone tries to create an invalid object or set a field to an invalid value. That's because, in reality, you should throw an exception.

Specifically, POJOs need to validate ALL data values that are passed to them - especially the values passed into their constructors and into any setters that they provide. For example, while a constructor might take a parameter called "numCards" as an "int", the POJO needs to ensure that "numCards" is not negative and that it doesn't exceed some maximum upper bound. In that case, the constructor needs to check for those two possibilities and throw an exception - specifically **ArgumentOutOfRangeException** - if either of them occurs.

Here's the Dice example with some of those exceptions in the Die and Cup constructors:

```
using System;

namespace LWTech.DiceSimulation
{
   public class Die
   {
      private static Random rng = new Random();

      public int Sides { get; private set; }
      public int Value { get; private set; }

      public Die(int sides = 6, int value = 0)
      {
         if (sides < 2 || sides > 30)
            throw new ArgumentsOutOfRangeException("Incorrect number of sides for new die.");

         this.Sides = sides;
         this.Value = (value > 0 && value <= sides) ? value : RandomDieValue();
      }

      public void Roll()
      {
         Value = RandomDieValue();
      }

      private int RandomDieValue()
      {
         return rng.Next(Sides) + 1;
      }
```

```
    public override string ToString()
    {
      return "[" + Value + "]";
    }
  }

  public class Cup
  {
    private Die[] dice;

    public int NumDice { get; private set; }

    public Cup(int numDice = 5)
    {
      if (numDice < 1 || numDice > 20)
        throw new ArgumentOutOfBoundsException("Incorrect number of dice for new Cup.");

      this.NumDice = numDice;
      dice = new Die[numDice];
      for (int i = 0; i < numDice; i++)
        dice[i] = new Die();
    }

    public void Roll()
    {
      foreach (Die d in dice)
        d.Roll();
    }

    public override string ToString()
    {
      string s = "[";
      foreach (Die d in dice)
        s += d.ToString();
      s += "]";
      return s;
    }
  }


  public class Program
  {
    public static void Main()
    {
      Cup myYahtzeeCup = new Cup(5);
      Console.WriteLine(myYahtzeeCup);
      myYahtzeeCup.Roll();
      Console.WriteLine(myYahtzeeCup);
    }
  }
}
```