CMPSCI 187 Discussion #2: Throwing a Million Pairs of Dice
Individual Handout

David Mix Barrington
19 September 2012

A **pseudorandom number generator** takes an initial number called a **seed** and repeatedly performs some operation on it to get a sequence of new numbers. The Java `Random` class defines objects that are such generators.

Looking at the API for `Random`, you should see that you can construct a `Random` object in two ways: with the constructor "`Random( )`", which starts the generator from a seed that it gets itself, unlikely to equal any other seed, or with the constructor "`Random (long seed)`", which starts it with the number `seed` as the seed.

In this discussion you're going to write a simple program using a `Random` object. In gaming, "throwing 2D6" means throwing two six-sided dice and adding the results, so that you get a result in the range from 2 through 12, inclusive. As you may know and as you may see later in CMPSCI 240, a score of 7 is most likely and scores of 2 and 12 are least likely.

Your job is to "throw 2D6" a million times, recording how many times each total comes up. You'll do this once with a generator that gets its own seed, and once with a generator whose seed is the number 1776. (You can plug this `int` into the `long` parameter of the constructor.) Once the `Random` object is created, you get die values from it by calling the method `nextInt(k)`, which gives you back an `int` in the range from 0 (inclusive) through k (exclusive) that is equally likely to be any of those k numbers. You'll need some arithmetic to get a number in the range from 1 through 6. Generate two such numbers, add them together, and you have the first of your million 2D6 values.

We hope you can figure out how to use an array to count the occurrences of each value, and how to print out the result. (There is usually a rather similar assignment in CMPSCI 121.) What you will hand in on the response sheet is the two sequences of eleven numbers, the number of times each value from 2 through 12 came up for each of your two generators. The hope is that all the 1776 generators will give the same counts, but there are ways that this might not happen.

Don't worry about optimizing your code – a million is a fairly small number in computer terms, and as long as you do something sensible your running time should be small. Your output format can be anything that tells you the counts, so that you can write them on the response sheet.

Remember that you should work in pairs (or just possibly groups of three) and submit one response per group. Your partner should not be your partner again during the term (though you may reuse your partner from last week, since that was an individual exercise).