

## Re: Selecting a "random" quotation

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2008-06/msg01191.html>

---

- *From:* Tim Smith <[reply\\_in\\_group@xxxxxxxxxxxxxxxxxxxx](mailto:reply_in_group@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sat, 14 Jun 2008 00:26:19 -0700
- 

In article <[kbpv445nqelb655r16jn6qf8fiuddfop59@xxxxxxxx](mailto:kbpv445nqelb655r16jn6qf8fiuddfop59@xxxxxxxx)>, Roedy Green <[see\\_website@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:see_website@xxxxxxxxxxxxxxxxxxxxxxxx)> wrote:

Here is some code I use to select a "random" quotation to put on a web page. It changes every 4 hours.

```
/**
 * 86,400,000 the number of milliseconds in 24 hour day. Easily
 * fits into an int.
 * We change the quotations every 4 hours, UTC.
 */
private static final int CHANGE_INTERVAL_IN_MILLIS = 4 * 60 * 60 *
1000;

final Adler32 digester = new Adler32();
int seed = ( int ) ( System.currentTimeMillis() /
CHANGE_INTERVAL_IN_MILLIS );
```

Result: seed is an integer somewhere around 85000 (ballpark), that goes up by one every four hours.

```
/// digester.update(int) not clearly documented, probably
wants ubyte.
digester.update( ( byte ) ( seed & 0xff ) );
```

This byte changes every 4 hours.

```
digester.update( ( byte ) ( ( seed >>> 8 ) & 0xff ) );
```

This byte changes every 42 days.

## Re: Selecting a "random" quotation

```
digester.update( ( byte ) ( ( seed >>> 16 ) & 0xff ) );
```

This byte changes every 30 years. It is 1 right now. It will become 2 around 2030.

```
digester.update( ( byte ) ( ( seed >>> 24 ) & 0xff ) );
```

This byte changes every 7700 years.

```
digester.update( fileBeingProcessed.getPath().getBytes(
"UTF-8" ) );
```

What is fileBeingProcessed? Is that the quote file, or the file containing the web page you are generating a quote for?

If the former, then it would not be changing, and so the net result is that your input to the digest function (which is not really a digest function—see below) is only changing very slightly every four hours.

If the later, then that's a bit better. But you still have the problem that you are using Adler32.

Adler32 is not really meant to be a message digest function. It is meant to be used in places where you'd use something like CRC, for error checking. Adler32 is even less good as a message digest than CRC would be, because it was purposefully designed for use where speed was more important than reliability.

Furthermore, Adler32 has a serious problem for messages under 128 bytes. This is described in detail in RFC 3309:

[<http://tools.ietf.org/html/rfc3309>](http://tools.ietf.org/html/rfc3309)

The obvious question is this: why are you using Adler32? This doesn't sound like an application where speed would be critical. If it is for some reason, you can fix that with caching. Since you want to keep the quote for a given page for four hours, just keep a table of the current quote for each page, and regenerate the table every four hours.

Anyway, if you replace Adler32 with a real message digest, such as MD5 or something in the SHA family. With these functions, even a small change in input changes about half the output bits, in a way that is for almost all practical purpose, random.

## Re: Selecting a "random" quotation

```
// get rid of high sign bit, and low bit, which seem to be
1 disproportionately.
final int digest = (( ( int ) digester.getValue() ) << 1)

                2;
```

You won't need this with a good digest.

```
final String[] candidates = category.getQuotations();

final String chosenQuote = candidates[ digest %
candidates.length ];
```

Given a digest that is effectively a uniformly distributed random int, picking a candidate using mod that way introduces bias. Let me illustrate with bytes, not ints, to keep the numbers smaller.

So, imagine a randomly distributed unsigned byte, 0–255, R. Suppose you have 100 candidates. Consider candidate 70.  $R \% 100 == 70$  when R is 70 or 170.

Now consider candidate 40.  $R \% 100 == 40$  when R is 40, 140, or 240. Note that there are THREE values of R that give quote 40, but only TWO values of R that give quote 70. You'll end up seeing quote 40 three times for every two times you see quote 70.

However, if the number of candidates is small compared to the maximum value of your random number, you can probably ignore this problem. For example, if there were only 10 candidates instead of 100 in the example above, then candidate 4, for example, would be picked when R is one of:

```
4 14 24 34 44 54 64 74 84 94 104 114 124 134 144 154 164 174
184 194 204 214 224 234 244 254
```

and candidate 7 would be picked for these R values:

```
7 17 27 37 47 57 67 77 87 97 107 117 127 137 147 157 167 177
187 197 207 217 227 237 247
```

Quote 4 will be selected 26 times for every 25 times quote 7 is selected. You probably aren't going to notice that. If R is not a byte, but an integer that can go up to a full 32 bits, then even if you thousands of quotes, the bias would be very small.

However it often SEEMS to sometimes pick the same quote for two different pages, more often than I would intuitively think it would.

Re: Selecting a "random" quotation

You intuition is probably wrong here (that's not a slam at you—pretty much everyone is wrong here, which is why the phenomenon you are probably seeing is called the "Birthday Paradox" rather than the "Roedy Intuition Failure" :-)).

The usual way the birthday paradox is stated is as this problem:

How many people do you have to have in a room to have a 50/50 chance that two of them have the same birthday?

Most people figure it is around 180. The actual answer is around 23, a number most people are surprised by.

If the question had been:

How many people do you have to have in a room to have a 50/50 chance that someone else share's your birthday?

then it is around 180, but what was asked for was that there exist two people with the same birthday. Number the people, and consider person 1. Not much to consider, so go on to person 2. There is a  $1/365$  chance person 2 has the same birthday as 1, or a  $364/365$  person 2 does not match 1. In the later case, go on to 3. For 3 to not match 1 or 2, there is a  $363/365$  chance. For 4 to not match any of the previous 3, there is a  $362/365$  chance, and so on.

Thus, the chance that N people will not have any matches is:

$$P_{no} = 364/365 * 363/365 * \dots * (365-N+1)/365$$

and then chance N people WILL have a match is  $1-P_{no}$ . Those terms being multiplied together start close to 1, but there's an effect here a lot like compound interest—that product gets away from you a lot faster than you might expect at first.

If you generalize this, and do some approximations, it turns out that if you have a year of N days, you get a 50/50 chance of a match when you have about  $\sqrt{N}$  people.

Think of your pages as your people, and your quotes as their birthdays. So, if you have N quotes available, then you'd get about even chances of two pages having the same quote if you have  $\sqrt{N}$  pages.

—  
—Tim Smith