Your PasswordCard - Algorithm

4-5 minutes

<< back to FAQ

What is this?

This page describes the algorithm used to create the PasswordCards, and provides a downloadable implementation in Java. It is intended to ensure that if this website were ever to disappear, it would still be possible to regenerate existing PasswordCards or generate new ones by reimplementing the algorithm or using the provided code.

The algorithm

This site is implemented in Java, and the algorithm makes use of some Java utility classes. The exact algorithms used by these classes are documented in their Javadocs, and are not repeated here. If you implement this algorithm in Java obviously you can just use the utility classes. For other languages you would have to create private implementations of the algorithms.

The Javadoc for the `java.util.Random` class can be found here, and the Javadoc for the `java.util.Collections.shuffle()` method here.

Beware that not all implementations of the Java API implement `Collections.shuffle()` the same! For instance, Android uses a different algorithm. In those cases you would have to create a private implementation of the `Collections.shuffle()` method.

This is the algorithm in pseudo code:

1. the card number is an unsigned 64-bit integer, but the pseudo random number generator expects a signed 64-bit number as seed, so convert the unsigned card number to a signed seed by interpreting it as a two's complement 64-bit integer
2. create an instance of `java.util.Random` or equivalent pseudo random number generator and seed it with the seed
3. create a two dimensional array of characters, 29 columns wide and 9 rows high
4. initialise the first, symbol row to "■□▲△○●★☂☀☁☹☺♠♣♥♦♫€¥£$!?¡¿⊙◐◩&#xfffd;"
5. shuffle it with the `java.util.Collections.shuffle()` or equivalent method, passing in the `java.util.Random` instance
6. iterate over the rest of the positions on the card left to right, top to bottom (breadth first) and for each position select a character from one of the three alphabets using the `Random.nextInt(`*n*`)` method, where *n* is the length of the alphabet
    1. if the card is to include symbols then use the alphanumeric plus symbols alphabet (see below) for every *even* column in this step (assuming zero-based column numbering); use the alphanumeric alphabet for the odd columns
    2. if the card is to have a digits-only area then use the numeric alphabet for the lower half of the rows

These are the three alphabets:

1. alphanumeric: "23456789abcdefghjkmnpqrstuvwxyzABCDEFGHJKLMNPQRSTUVWXYZ"
2. alphanumeric plus symbols: "23456789abcdefghjkmnpqrstuvwxyzABCDEFGHJKLMNPQRSTUVWXYZ@#$%&*<>?€+{}[]()/\"
3. numeric: "0123456789"

The most efficient way of implementing this algorithm depends on your chosen language and its available APIs. Giving programming advice is beyond the scope of this website so we can't help you there. For the exact implementation in Java you can download the source code below:

The code

This ZIP file contains the source code of the Java implementation of this algorithm, as well as a compiled jar file which you can run from the command line. See the included README file for details. You will need to have Java installed in order to run this, which you can get here.

Please note that this code is released as open source under the GPL version 3 license. This means that if you use it in your own project and you distribute that project, your project must be open source with the GPL license as well!

PasswordCard.zip

<< back to FAQ