Markov model data type. Create an immutable data type `MarkovModel` to represent a Markov model of order $k$ from a given text string. The data type must implement the following API:

```java
public class MarkovModel
{
    // Note: all of the below constructors/methods should be public.
    public MarkovModel(String text, int k) // create a Markov model of order k from given text
    { // Assume that text has length at least k.
        // order k of Markov model
        int order()
        int freq(String kgram) // number of occurrences of kgram in text
        { // (throw an exception if kgram is not of length k)
            int freq(String kgram, char c) // number of times that character c follows kgram
            { // (throw an exception if kgram is not of length k)
                char rand(String kgram) // random character following given kgram
                { // (Throw an exception if kgram is not of length k.
                    // Throw an exception if no such kgram.)
                    String gen(String kgram, int T) // generate a String of length T characters
                    { // by simulating a trajectory through the corresponding
                        // Markov chain. The first k characters of the newly
                        // generated String should be the argument kgram.
                        // Throw an exception if kgram is not of length k.
                        // Assume that T is at least k.
                    }
                }
            }
        }
    }
}
```

- **Constructor.** To implement the data type, create a symbol table, whose keys will be `String k-grams`. You may assume that the input text is a sequence of characters over the ASCII alphabet so that all `char` values are between 0 and 127. The value type of your symbol table needs to be a data structure that can represent the frequency of each possible next character. The frequencies
should be tallied as if the text were **circular** (i.e., as if it repeated the first \( k \) characters at the end).

- **Order.** Return the order \( k \) of the Markov Model.

- **Frequency.** There are two frequency methods.
  - \( \text{freq}(k\text{gram}) \) returns the number of times the \( k \)-gram was found in the original text.
  - \( \text{freq}(k\text{gram}, c) \) returns the number of times the \( k \)-gram was followed by the character \( c \) in the original text.

- **Randomly generate a character.** Return a character. It must be a character that followed the \( k \)-gram in the original text. The character should be chosen randomly, but the results of calling \( \text{rand}(k\text{gram}) \) several times should mirror the frequencies of characters that followed the \( k \)-gram in the original text.

- **Generate pseudo-random text.** Return a \texttt{String} of length \( T \) that is a randomly generated stream of characters whose first \( k \) characters are the argument \( k\text{gram} \). Starting with the argument \( k\text{gram} \), repeatedly call \( \text{rand()} \) to generate the next character. Successive \( k \)-grams should be formed by using the most recent \( k \) characters in the newly generated text. Use a \texttt{StringBuilder} object to build the stream of characters (otherwise, as we saw when discussing performance, your code will take order of \( N^2 \) time to generate \( N \) characters, which is too slow).

Implement throwing the \texttt{RuntimeExceptions} indicated by the API above.