Lab 7 Example
For this example we will train a Markov model on the string “abbabaab” using Mardov constant K=2.

In our training phase (in class MarkovModel) we will maintain a string S, which is always the most recent K characters of the training input, and the next character, C.

At each step of our training loop we
1. Read the next value of C from the input.
2. Update our State info with the fact that state S was followed by C.
3. Add C to S and drop the first character of S, so S maintains length K
   
   ```java
   S += (char) C;
   S = S.substring(1);
   ```
We start with $S$ consisting of the first $K$ letters of the input: $S = \text{"ab"}$

The remaining input is “babaab”. There are no states.
Step 1  S = “ab”  Remaining input: “babaab”.

C is the next character of the input: ‘b’. The remaining input is “abaab”.

We have string “ab” with followup ‘b’; there is no state for “ab” so we make one and add the fact that it has ‘b’ as a followup:

State “ab” suffixes{ (‘b’ 1) }

Then we add C to S and drop the first letter of S. S is now “bb”.
Step 2. $S = \text{"bb"}$ Remaining input: \text{"abaab"}.

C is the next character of the input: \text{"a"}. The remaining input is \text{"baab"}.

We have string \text{"bb"} with followup \text{"a"}; there is no state for \text{"bb"} so we make one and add the fact that it has \text{"a"} as a followup:

State \text{"ab"} suffixes \{ (\text{"b"} \text{ 1} ) \}
State \text{"bb"} suffixes \{ (\text{"a"} \text{ 1}) \}

Then we add C to S and drop the first letter of S. S is now \text{"ba"}. 
Step 3. $S = \text{"ba"}$ Remaining input: \text{"baab"}.

C is the next character of the input: \text{‘b’}. The remaining input is \text{"aab"}.

We have string \text{“ba”} with followup \text{‘b’}; there is no state for \text{“ba”} so we make one and add the fact that it has \text{‘b’} as a followup:

State \text{“ab”} suffixes\{ (\text{‘b’} 1) \}
State \text{“bb”} suffixes \{(\text{‘a’} 1)\}
State \text{“ba”} suffixes \{(\text{‘b’} 1)\}

Then we add C to S and drop the first letter of S. S is now \text{“ab”}. 
Step 4. $S = \text{"ab"}$ Remaining input: \text{"aab"}.

$C$ is the next character of the input: ‘a’. The remaining input is “ab”.

We have string “ab” with followup ‘a’; there is a state for “ba” so we add to it the fact that we found ‘a’ as a followup:

- State “ab” suffixes{ (‘b’ 1) (‘a’ 1)}
- State “bb” suffixes { (‘a’ 1)}
- State “ba” suffixes { (‘b’ 1)}

Then we add $C$ to $S$ and drop the first letter of $S$. $S$ is now “ba”.
Step 5. $S = \text{"ba"}$ Remaining input: \text{"ab"}.

C is the next character of the input: ‘a’. The remaining input is \text{"b"}.

We have string \text{"ba"} with followup ‘a’; there is a state for \text{"ba"} so we add to it the fact that we found ‘a’ as a followup:

State \text{"ab"} suffixes\{ (‘b’ 1) (‘a’ 1)\}
State \text{"bb"} suffixes {\{‘a’ 1\}}
State \text{"ba"} suffixes {\{‘b’ 1) (‘a’ 1}\}

Then we add C to S and drop the first letter of S. S is now \text{"aa"}.

C is the next character of the input: ‘’. The remaining input is the empty string. We have string “aa” with followup ‘b’; there is no state for “aa” so we make one and add to it the fact that we found ‘b’ as a followup:

State “ab” suffixes {('b' 1) ('a' 1)}
State “bb” suffixes {('a' 1)}
State “ba” suffixes {('b' 1) ('a' 1)}
State “aa” suffixes {('b' 1)}

Then we add C to S and drop the first letter of S. S is now “ab”.
Step 7. \( S = \text{“ab”} \) Remaining input: “”.

Since there is no followup character we are done.

The training resulted in the following states:
State “ab” suffixes\{ (’b’ 1) (’a’ 1)\}
State “bb” suffixes \{(’a’ 1)\}
State “ba” suffixes \{(’b’ 1) (’a’ 1)\}
State “aa” suffixes \{(’b’ 1)\}
To use this to generate text we can start with any valid state. The easiest way might be to start with the first K characters of the training text, in this case “ab”.
We will maintain two variables:
   Text consists of all of the characters we have generated so far.
   S consists of the last K characters of Text.

We start with both of these variables equal to the first K characters of the training text.

At each step we use S to generate the next character C.
We add C onto Text.
We add C onto S and drop the first character of S, so S is always of length K.
Again, here are our states:
State “ab” suffixes {(‘b’ 1) (‘a’ 1)}
State “bb” suffixes {‘a’ 1)}
State “ba” suffixes {(‘b’ 1) (‘a’ 1)}
State “aa” suffixes {(‘b’ 1)}

Step 1: Text = “ab” S = “ab”.
We get the state for S and find it has suffixes { (‘b’ 1) (‘a’ 1)}
This means ‘a’ and ‘b’ are equally likely as followups.
Suppose we choose ‘b’
Now Text = “abb” and S = “bb”.

Our states:
State “ab” suffixes { (‘b’ 1) (‘a’ 1) }
State “bb” suffixes { (‘a’ 1) }
State “ba” suffixes { (‘b’ 1) (‘a’ 1) }
State “aa” suffixes { (‘b’ 1) }

Step 2: Text = “abb”  S = “bb”.
We get the state for S and find it has suffixes { (‘a’ 1) }
This means ‘a’ is the only followup
Now Text = “abba” and S = “ba”.
Our states:
State “ab” suffixes{ (‘b’ 1) (‘a’ 1)}
State “bb” suffixes { (‘a’ 1)}
State “ba” suffixes { (‘b’ 1) (‘a’ 1)}
State “aa” suffixes { (‘b’ 1)}

Step 3: Text = “abba”  S = “ba”. We get the state for S and find it has suffixes { (‘b’ 1) (‘a’ 1)}
In this situation, half of the time we’ll choose ‘b’ and half of the time ‘a’. Let’s choose ‘a’. Text = “abbaa”  S = “aa”
Our states:
State “ab” suffixes {('b' 1) ('a' 1)}
State “bb” suffixes {('a' 1)}
State “ba” suffixes {('b' 1) ('a' 1)}
State “aa” suffixes {('b' 1)}

Step 4: Text = “abbaa”  S = “aa”.
Here we might choose ‘b’
Text = “abbaab”  S = “ab”
Our states:
State “ab” suffixes {('b' 1) ('a' 1)}
State “bb” suffixes {('a' 1)}
State “ba” suffixes {('b' 1) ('a' 1)}
State “aa” suffixes {('b' 1)}

Step 5: Text = “abbaab”  S = “ab”.
Here we might choose ‘b’
Text = “abbaabb”  S = “bb”
We could keep this going for a long time, generating as many characters as we wish.
There is one problem that can occur. Consider the training text “abbabaabc”
This gives states
State “ab” suffixes{ ('b' 1) ('a' 1) ('c' 1)}
State “bb” suffixes {('a' 1)}
State “ba” suffixes {('b' 1) ('a' 1)}
State “aa” suffixes {('b' 1)}

At any point when we are in state “ab” we could choose to generate ‘c’, which puts us in state “bc”. But there is no state “bc”. This problem happens whenever the final K characters of the input don’t form a state (because they don’t have a followup character). When this happens just move into a different state (perhaps the first K characters of the training text, which also is the first K characters of the generated text).