How does predictive text work?
Markov Model Assignment
Markov Model Assignment

It takes, as input, a sample of text.
Markov Model Assignment

It takes, as input, a sample of text.

It generates text in the same "style" as the input.
Markov Model Assignment

It takes, as input, a sample of text.

It generates text in the same "style" as the input.

There are two phases to this process:

1. Use the input to build a model.
2. Use the model to generate text.
Markov Model Assignment

It takes, as input, a sample of text.

It generates text in the same "style" as the input.

There are two phases to this process:

1. Use the input to build a model.
2. Use the model to generate text.

Actually, this is how a lot of artificial intelligence and machine learning works.
Markov Model Assignment

It takes, as input, a sample of text.

It generates text in the same "style" as the input.

There are two phases to this process:

1. Use the input to build a model.
2. Use the model to generate text.

The biggest difference about this assignment and previous assignments is the constructor does a lot of work here.
Constructor

As mentioned, the constructor for this program is where a lot of the magic happens.
As mentioned, the constructor for this program is where a lot of the magic happens.

The constructor uses a data structure you'll learn more about called a **symbol table** (ST.java).
Constructor

As mentioned, the constructor for this program is where a lot of the magic happens.

The constructor uses a data structure you'll learn more about called a symbol table (ST.java).

A symbol table is like a encyclopedia, you can look something up and find out more information about it.
Constructor

As mentioned, the constructor for this program is where a lot of the magic happens.

The constructor uses the `symbol table` data structure (ST.java).

A symbol table is like a encyclopedia, you can look something up and find out more information about it.

For this program, we're using our "encyclopedia" to look up how often a specific letter appears after a given word.
K-grams

We have some input text. We divide it into "k-grams".
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of $k$ adjacent characters.

How many k-grams are in this text if $k$ is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

4?

Nope.
We have some input text. We divide it into "k-grams".

A k-gram is a series of **k adjacent** *characters*.

How many k-grams are in this text if k is 3?

Sam saw her pal.
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of \textbf{k adjacent characters}.

How many k-grams are in this text if \( k \) is 3?

\textbf{Sam saw her pal.}

<table>
<thead>
<tr>
<th>Sam</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of **k adjacent characters**.

How many k-grams are in this text if k is 3?

**Sam saw her pal.**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>am_</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of **k adjacent characters**.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>am_</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th>saw</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th>saw</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th>saw</th>
<th>her</th>
<th>pal</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td>aw_</td>
<td>er_</td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td>w_h</td>
<td>r_p</td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td>_he</td>
<td>_pa</td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>saw</td>
<td>her</td>
<td>pal</td>
<td></td>
</tr>
<tr>
<td>am_</td>
<td>aw_</td>
<td>er_</td>
<td>al.</td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td>w_h</td>
<td>r_p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td>_he</td>
<td>_pa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th>saw</th>
<th>her</th>
<th>pal</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td>aw_</td>
<td>er_</td>
<td>al.</td>
</tr>
<tr>
<td>m_s</td>
<td>w_h</td>
<td>r_p</td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td>_he</td>
<td>_pa</td>
<td></td>
</tr>
</tbody>
</table>

14?
We have some input text. We divide it into "k-grams".

A k-gram is a series of k adjacent characters.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th></th>
<th>saw</th>
<th>her</th>
<th>pal</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_sa</td>
<td>_he</td>
<td>_pa</td>
<td></td>
</tr>
<tr>
<td>w_h</td>
<td></td>
<td>r_p</td>
<td></td>
</tr>
<tr>
<td>_he</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I.S
K-grams

We have some input text. We divide it into "k-grams".

A k-gram is a series of k **adjacent characters**.

How many k-grams are in this text if k is 3?

Sam saw her pal.

<table>
<thead>
<tr>
<th>Sam</th>
<th>saw</th>
<th>her</th>
<th>pal</th>
</tr>
</thead>
<tbody>
<tr>
<td>am_</td>
<td>aw_</td>
<td>er_</td>
<td>al.</td>
</tr>
<tr>
<td>m_s</td>
<td>w_h</td>
<td>r_p</td>
<td>l.S</td>
</tr>
<tr>
<td>_sa</td>
<td>_he</td>
<td>_pa</td>
<td>.Sa</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.
Character Frequencies

We read our input text, $k$ characters at a time.

For each $k$-gram, we record the character after it.
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is banana and our k is 2.
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is 2.
Character Frequencies

We read our input text, $k$ characters at a time.

For each $k$-gram, we record the character after it.

We're building a table of $k$-grams and char. frequencies.

Let's do an example. Our input is **banana** and our $k$ is 2.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is banana and our k is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
</table>
| ba| 0 | 0 | 0 | 0

You need one column for every possible character.
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is **2**.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, $k$ characters at a time.

For each $k$-gram, we record the character after it.

We're building a table of $k$-grams and char. frequencies.

Let's do an example. Our input is **banana** and our $k$ is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is **2**.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ba</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>an</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, $k$ characters at a time.

For each $k$-gram, we record the character after it.

We're building a table of $k$-grams and char. frequencies.

Let's do an example. Our input is **banana** and our $k$ is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ba</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>an</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>an</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, \( k \) characters at a time.

For each \( k \)-gram, we record the character after it.

We're building a table of \( k \)-grams and char. frequencies.

Let's do an example. Our input is **banana** and our \( k \) is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ba</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>an</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>na</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
We read our input text, \( k \) characters at a time.

For each \( k \)-gram, we record the character after it.

We're building a table of \( k \)-grams and char. frequencies.

Let's do an example. Our input is **banana** and our \( k \) is 2.

\[
\begin{array}{ccc}
\text{a} & \text{b} & \text{n} \\
\hline
\text{ba} & 0 & 0 & 1 \\
\text{an} & 1 & 0 & 0 \\
\text{na} & 0 & 0 & 1 \\
\end{array}
\]
Character Frequencies

We read our input text, $k$ characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our $k$ is **2**.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>an</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>na</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is *banana* and our k is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>an</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>na</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is 2.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ba</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>an</strong></td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>na</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is **2**.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>an</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>na</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Character Frequencies

We read our input text, k characters at a time.

For each k-gram, we record the character after it.

We're building a table of k-grams and char. frequencies.

Let's do an example. Our input is **banana** and our k is 2.

<table>
<thead>
<tr>
<th>k-gram</th>
<th>a</th>
<th>b</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>an</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>na</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Character Frequencies

Your turn! The input is alfalfa and k is 3.
Your turn! The input is `alfalfa` and k is 3.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>l</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>alf</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lfa</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>fal</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>faa</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>aal</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Testing the Constructor

All of the work we did up until now was in the constructor.
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:

```plaintext
int order()                  // Returns k. Store k as an IV.
```
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int order()</td>
<td>Returns k. Store k as an IV.</td>
</tr>
<tr>
<td>String toString()</td>
<td>&quot;aa: a 1 g 1\nag: a 3 g 2\nncg: ...&quot;</td>
</tr>
</tbody>
</table>
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int order()</td>
<td>Returns k. Store k as an IV.</td>
</tr>
<tr>
<td>String toString()</td>
<td>&quot;aa: a 1 g 1\nag: a 3 g 2\ncg: ...&quot;, &quot;aa: a 1 g 1\nag: a 3 g 2\ncg: a 1\nga: a 1 g 4\ngc: g 1\ngg: a 1 c 1 g 1&quot;</td>
</tr>
</tbody>
</table>
# Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int order()</td>
<td>Returns k. Store k as an IV.</td>
</tr>
</tbody>
</table>
| String toString() | "aa: a 1 g 1
nag: a 3 g 2
ncg: ..."                  |
| int freq(String kgram) | How many times does kgram appear in the text? |
Testing the Constructor

All of the work we did up until now was in the constructor.

Once you have it working, you're ~50% done.

Here are some methods to help you test the constructor:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int order()</td>
<td>Returns k. Store k as an IV.</td>
</tr>
<tr>
<td>String toString()</td>
<td>&quot;aa: a 1 g 1\nag: a 3 g 2\ncg: ...&quot;</td>
</tr>
<tr>
<td>int freq(String kgram)</td>
<td>How many times does kgram appear in the text?</td>
</tr>
<tr>
<td>int freq(String kgram, char c)</td>
<td>How many times does c appear after kgram in the text?</td>
</tr>
</tbody>
</table>
Okay, now that we've built our model, let's use it!
Generating Text

Okay, now that we've built our model, let's use it!

There's just one method left: char random(String kgram).
Generating Text

Okay, now that we've built our model, let's use it!

There's just one method left: char random(String kgram).

```java
int[] array = {0, 0, 1, 1};
int index = StdRandom.discrete(array);
// there's a 50% chance index is 2, 50% it's 3
```
Okay, now that we've built our model, let's use it!

There's just one method left: char random(String kgram).

```java
int[] array = {0, 0, 1, 1};
int index = StdRandom.discrete(array);
// there's a 50% chance index is 2, 50% it's 3

int[] array2 = {1, 2, 3, 4};
int index2 = StdRandom.discrete(array2);
```
Generating Text

Okay, now that we've built our model, let's use it!

There's just one method left: `char random(String kgram)`.

```java
int[] array = {0, 0, 1, 1};
int index = StdRandom.discrete(array);
// there's a 50% chance index is 2, 50% it's 3

int[] array2 = {1, 2, 3, 4};
int index2 = StdRandom.discrete(array2);
```

How likely is it for index2 to be 3?
Generating Text

Okay, now that we've built our model, let's use it!

There's just one method left: char **random**(String kgram).

```java
int[] array = {0, 0, 1, 1};
int index = StdRandom.discrete(array);
// there's a 50% chance index is 2, 50% it's 3

int[] array2 = {1, 2, 3, 4};
int index2 = StdRandom.discrete(array2);
```

How likely is it for index2 to be 3?

\[
\frac{4}{1+2+3+4}, \text{ so } 40\%
\]
Lastly, the Client

TextGenerator.java is a short program.
Lastly, the Client

TextGenerator.java is a short program.

It uses StdIn.readAll().
Lastly, the Client

TextGenerator.java is a short program.

It uses StdIn.readAll().

Prints output one character at a time (to avoid having to build the entire output in a String).
Lastly, the Client

TextGenerator.java is a short program.

It uses StdIn.readLine().

Prints output one character at a time (to avoid having to build the entire output in a String).
Lastly, the Client

TextGenerator.java is a short program.

It uses StdIn.readLine().

Prints output one character at a time (to avoid having to build the entire output in a String).

Try generating papers, lyrics, code, music, etc. with your MarkovModel!
Thanks!

Thanks for coming.

Good luck on the assignment!

I'll stay after for questions.