String Processing

- Often need to go thru string and change it.
  Typically
    - Look for some **thing** in *inputString*
    - Modify that **thing**
    - Add part of input searched so far and **modified** **thing** to *outputString*

- **Strings are immutable**
  - Can’t really change them
  - No matter how it looks, changes mean new string is created
String Processing

- If looking for characters in *inputString*
  - Typically use `charAt()` to locate
  - If modification results in a `char`, can appended `char` to `outputString` with `+`
  - Else append modified `String`
  - (Can do all of the above using `substring()`)

- More complicated changes invariably involve use of `substring()`
String Processing

❖ General Recipe
  ❑ Locate position of next textToChange using \texttt{indexOf()}.
  ❑ Append all of \texttt{inputString} up to position to \texttt{outputString}.
  ❑ Strip the corresponding text from \texttt{inputString} using \texttt{substring()}

  ❑ Modify located textToChange
  ❑ Append modifiedText to \texttt{outputString}
  ❑ Strip corresponding text from \texttt{inputString}

  ❑ Repeat until at end of \texttt{inputString}
String Processing Example

- Replace all occurrences of key with sub

```java
String replace(String in, String key, String sub) {
    String out = "";
    int keyLen = key.length();
    int pos = in.indexOf(key);
    while (pos >= 0) {
        out += in.substring(0, pos); // copy prefix to out
        String mod = sub; // "modify" key
        in = in.substring(pos + keyLen); // Strip prefix + key
        out += mod; // add "modified key" to out
        pos = in.indexOf(key); // set up for repeat (if any)
    }
    out += in;
    return out;
}
```
Lab: Random Text Generation: n-gram

- Given a text source (book)
  
  this history of mish mash
  
  is a bit tristy.

- Pick an n-gram (random string length n)

  - This will start generation of random text that is similar in feel to the original text

- Example: 2-gram from above: is
Predictors

- What are predictors for “is”?
  
  this_history of mish mash
  is a bit tristy.

- Predictors: “ “, “t”, “h”, “ “, “t”

- Pick one of them and add on to predictor

- Say t is picked

- “ist”

- Use last n-1 char plus chosen letter as new predictor.
Generating random Text

- **Ngram “is”** – predictors(“ “, “t”, “h”, “ “, “t”)
  - Pick “t” - result is “ist”
- **Ngram “st”** – predictors(“o”, “y”)
  - Pick “o” – result is “isto”
- **Ngram “to”** – predictors (“r”)
  - Pick “r” result is “istor”