Using and Improving GladLibs

Introduction

Duke
UNIVERSITY

GladLib: Stories from Templates

- Ready to tackle modifying the GladLibs story-telling program: learning while telling stories!

GladLib: Stories from Templates

- Ready to tackle modifying the GladLibs story-telling program: learning while telling stories!
  - All the pieces of the program are ones you could write yourself!
  - You'll start with a working program, and understand and extend the program
- Class design, method design, understanding program limitations but reveling in telling stories!
  - Creating re-usable story-telling components

Structure of GladLib.java

- Constructor initializes object
  - ArrayList for nouns, verbs, colors, …
  - Random object for generating random choices
- Read story template, process each word
  - If <country> or <timeframes>s found: replace
- Print!

```
In <country> a long time ago, nearly <number>
<timeframe>s ago, there lived a <color>,
<adjective> <animal>. It so loved to sing and
dance, but there was a <adjective>,
<adjective> <animal> named <name> that
scared it so much!
```

Stories from Template

- Running program generates stories
  - One public method: **`makeStory`**

In Kenya a long time ago, nearly 245 decades ago, there lived a pink, funny tiger. It so loved to sing and dance, but there was a angry, gigantic lion named Lance that scared it so much!

Stories from Template

- Running program generates stories
  - One public method: **`makeStory`**

In Ecuador a long time ago, nearly 105 months ago, there lived a jovial, yellow polar bear. It so loved to sing and dance, but there was a furious, angry rabbit named Albert that scared it so much!

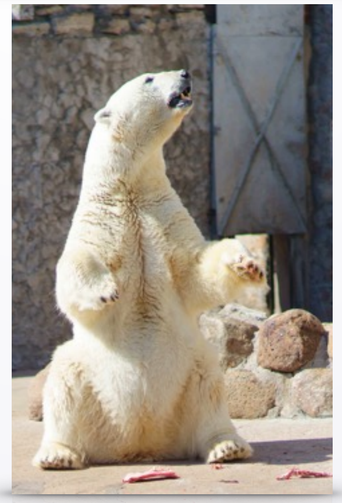
Stories from Template

- Running program generates stories
  - One public method: **`makeStory`**

Stories from Template

- Running program generates stories
  - One public method: **`makeStory`**

In Ecuador a long time ago, nearly 105 months ago, there lived a yellow, jovial polar bear. It so loved to sing and dance, but there was a furious, angry rabbit named Albert that scared it so much!



Reading Words and Printing

- Template read from file or URL, simply loop over each word, if <label>, then replace
  - Finding labels straightforward use of .indexOf and .substring, for <noun>! or "<color>"

```
private String processWord(String w){
    int first = w.indexOf("<");
    int last = w.indexOf(">",first);
    if (first == -1 || last == -1){
      return w;
    }
    …
}
```

Reading Words and Printing

- Template read from file or URL, simply loop over each word, if <label>, then replace
  - Finding labels straightforward use of .indexOf and .substring, for <noun>! or "<color>"
- Printing story displays words in console

```
private void printOut(String s, int lineWidth){
    int charsWritten = 0;
    …
```

Reading Words and Printing

- Template read from file or URL, simply loop over each word, if <label>, then replace
  - Finding labels straightforward use of .indexOf and .substring, for <noun>! or "<color>"
- Printing story displays words in console
  - Method uses parameterized line-width

```
private void printOut(String s, int lineWidth){
    int charsWritten = 0;
    …
```

Reading Words and Printing

- Template read from file or URL, simply loop over each word, if <label>, then replace
  - Finding labels straightforward use of .indexOf and .substring, for <noun>! or "<color>"
- Printing story displays words in console
  - Method uses parameterized line-width
  - Could modify to print to file

```
private void printOut(String s, int lineWidth){
    int charsWritten = 0;
    …
```

Instance Variables

- ArrayLists for each <noun>, <color>, …
  - Name variables appropriately,
  - Create and initialize in constructor
  - Need all of them to replace a word.

```
private ArrayList<String> adjectiveList;
private ArrayList<String> nounList;
private ArrayList<String> colorList;
private ArrayList<String> countryList;
private ArrayList<String> nameList;
private ArrayList<String> animalList;
private ArrayList<String> timeList;
private Random myRandom;
```

Instance Variables

- ArrayLists for each <noun>, <color>, …
  - Name variables appropriately,
  - Create and initialize in constructor
  - Need all of them to replace a word.

```
private ArrayList<String> adjectiveList;
private ArrayList<String> nounList;
private ArrayList<String> colorList;
private ArrayList<String> countryList;
private ArrayList<String> nameList;
private ArrayList<String> animalList;
private ArrayList<String> timeList;
private Random myRandom;
```

Instance Variables

- ArrayLists for each <noun>, <color>, …
  - Name variables appropriately,
  - Create and initialize in constructor
  - Need all of them to replace a word.

```
private ArrayList<String> adjectiveList;
private ArrayList<String> nounList;
private ArrayList<String> colorList;
private ArrayList<String> countryList;
private ArrayList<String> nameList;
private ArrayList<String> animalList;
private ArrayList<String> timeList;
private Random myRandom;
```

Instance Variables

- ArrayLists for each <noun>, <color>, …
  - Name variables appropriately,
  - Create and initialize in constructor
  - Need all of them to replace a word.

```
private ArrayList<String> adjectiveList;
private ArrayList<String> nounList;
private ArrayList<String> colorList;
private ArrayList<String> countryList;
private ArrayList<String> nameList;
private ArrayList<String> animalList;
private ArrayList<String> timeList;
private Random myRandom;
```

Instance Variables

- ArrayLists for each <noun>, <color>, …
  - Name variables appropriately,
  - Create and initialize in constructor
  - Need all of them to replace a word.

```
private ArrayList<String> adjectiveList;
private ArrayList<String> nounList;
private ArrayList<String> colorList;
private ArrayList<String> countryList;
private ArrayList<String> nameList;
private ArrayList<String> animalList;
private ArrayList<String> timeList;
private Random myRandom;
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements

```
private String getSubstitute(String label) {
    if (label.equals("country")) {
      return randomFrom(countryList);
    }
    if (label.equals("color")){
      return randomFrom(colorList);
    }
    if (label.equals("noun")){
      return randomFrom(nounList);
    }
    if (label.equals("name")){
      return randomFrom(nameList);
    }
// more code here
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements

```
private String getSubstitute(String label) {
    if (label.equals("country")) {
      return randomFrom(countryList);
    }
    if (label.equals("color")){
      return randomFrom(colorList);
    }
    if (label.equals("noun")){
      return randomFrom(nounList);
    }
    if (label.equals("name")){
      return randomFrom(nameList);
    }
// more code here
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements

```
private String getSubstitute(String label) {
    if (label.equals("country")) {
      return randomFrom(countryList);
    }
    if (label.equals("color")){
      return randomFrom(colorList);
    }
    if (label.equals("noun")){
      return randomFrom(nounList);
    }
    if (label.equals("name")){
      return randomFrom(nameList);
    }
// more code here
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements

```
private String getSubstitute(String label) {
    if (label.equals("country")) {
      return randomFrom(countryList);
    }
    if (label.equals("color")){
      return randomFrom(colorList);
    }
    if (label.equals("noun")){
      return randomFrom(nounList);
    }
    if (label.equals("name")){
      return randomFrom(nameList);
    }
// more code here
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements
  - Choose value from list at random in method **`randomFrom`**, return for story-telling

```
private String randomFrom(ArrayList<String> source){
    int index = myRandom.nextInt(source.size());
    return source.get(index);
}
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements
  - Choose value from list at random in method **`randomFrom`**, return for story-telling
  - Both methods are private, used by sequence of calls from public method **`makeStory`**

```
private String randomFrom(ArrayList<String> source){
    int index = myRandom.nextInt(source.size());
    return source.get(index);
}
```

Find Substitute for <color>

- Based on <color> or <noun> or … use the appropriate list of replacements
  - Choose value from list at random in method **`randomFrom`**, return for story-telling
  - Both methods are private, used by sequence of calls from public method **`makeStory`**
  - Argument passed to source is instance variable

```
private String randomFrom(ArrayList<String> source){
    int index = myRandom.nextInt(source.size());
    return source.get(index);
}
```

Initializing ArrayLists

- We must construct and initialize many instance variables
  - Source could be URL or local directory

`initializeFrom("data")` or **`initializeFrom("http:..")`**

```
private void initializeFromSource(String source) {
   adjectiveList= readIt(source+"/adjective.txt");
   nounList     = readIt(source+"/noun.txt");
   colorList    = readIt(source+"/color.txt");
   countryList  = readIt(source+"/country.txt");
   nameList     = readIt(source+"/name.txt");
   animalList   = readIt(source+"/animal.txt");
   timeList     = readIt(source+"/timeframe.txt");
}
```

ArrayList Instance Variables

- To create a new label, <verb>, must create new instance variable, e.g., **`verbList`**
  - Modify code in two methods with **`verbList`**
- **`initializeFromSource`**
  - called: GladLib constructors
- **`getSubstitute`**
  - called: makeStory, fromTemplate, processWord
- Program documentation should include information on how to modify code