CSCI 2320: Principles of Programming Language
Programming Assignment 4: OOP in Ruby

Points: 15

Due: Tuesday, November 21 (11:59 PM)

Collaboration Level: 1 (Individual Assignment)

Part I (10 pt)
Submit .rb source file

Write a Ruby program to crawl the WWW beginning at http://www.bowdoin.edu with the goal of detecting broken links *within the Bowdoin domain*. You must print the number of broken links as well as all the broken links. Your implementation must be *object-oriented*. Define the necessary class(es) (and modules if needed).

Recursion

Doing a recursion intelligently is the key here. For example: Suppose that http://www.bowdoin.edu has a link to http://www.bowdoin.edu/computer-science, and that the computer science web page has a broken link http://www.bowdoin.edu/~invalid in it. In order to detect this broken link, you will have to recursively fetch web pages as follows.

1. First read the content of http://www.bowdoin.edu and gather all the hyperlinks there.
2. Then visit each hyperlink and do the same.
    - E.g., read the content of http://www.bowdoin.edu/computer-science and gather all the hyperlinks that are there, including the broken hyperlink http://www.bowdoin.edu/~invalid.
3. Never visit any link twice; otherwise, you will fall into an infinite loop. Also, never go outside of the Bowdoin domain.

Caution: it will take a really long time, even if you visit each link just once.

One possible design

(This is not a recommendation, just one possibility among numerous others)

- Define a WebCrawler class

- Use array instance variables @visitedLinks and @brokenLinks (and initialize them to empty array denoted by []). You can also create an instance variable @startURL, which should be set to "http://www.bowdoin.edu" and an array instance variable @allLinks, which should be initialized to ["http://www.bowdoin.edu"].

- Define a getLinks method within the WebCrawler class. This method will be similar to the one we did in class (change the parameters if needed).

- Define an explore method that will do the following:
    - While the @allLinks array is non-empty
        - x = the first element of @allLinks
        - If x hasn't yet been visited, check if x is broken. If it is broken, add x to @brokenLinks. Otherwise, get all the links that are in the web page x, and add each link to @allLinks *if it's not already there in @allLinks*. Also, make sure that you don't add any non-Bowdoin link to @allLinks.
        - Remove x from @allLinks and put it in @visitedLinks.

- Create an object of the WebCrawler class outside of that class and get things rolling by calling the explore method. Print the number of broken links and the @brokenLinks array.

Caution: The above design is very high-level with many details missing. As a result, do not take it as a guideline for a solution.

Implementation Tips

- Don't attempt to crawl the whole http://www.bowdoin.edu domain. Instead, start at a smaller scale. For example, crawl the http://www.bowdoin.edu/computer-science and limit yourself to http://www.bowdoin.edu/computer-science (that is, don't read any web page outside of the computer-science domain even if it's within the bowdoin.edu domain).
- Don't worry about an exact, accurate count of broken links, since there are many special cases (e.g., an "anchor" within a page is denoted by # in HTML). Part of the learning objective here is to recognize such special cases in WWW. Having said that, a rough count is sufficient for getting the full credit in this assignment.

Testbed

I have set up a testbed for testing your web crawler. Here it is:
http://www.bowdoin.edu/~mirfan/TestCrawler/
There are in total 5 unique links, 4 of them are valid and 1 broken (cows.html). It has most of our test cases, including circular links.

Relative vs. absolute links

For Bowdoin.edu (also generalizable to other sites), here are the two rules for converting relative links to absolute links:

1. If a relative link starts with a / (e.g., /zebra.html), it must be prepended by http://www.bowdoin.edu (e.g., http://www.bowdoin.edu/zebra.html).

2. If a relative link doesn't start with a / (e.g., zebra.html), it must be prepended by the url where it appears. Let's say you find the link "zebra.html" in the "http://www.bowdoin.edu/animals" website. Then its absolute link is: http://www.bowdoin.edu/animals/zebra.html.

So, how should we test if a link is relative or not? Here's an algorithm:

```
if the link does not start with "http" #a relative link
   if the link starts with "/"
      link.insert(0, "http://www.bowdoin.edu")
   else
      link.insert(0, base_url + "/") #Assuming base_url doesn't end with /
      #Here, base_url is the url (not necessarily bowdoin.edu) where you found the link.
```

Part II (5 pt)
Submit as a pdf file

1. Describe the difference between polymorphism and inheritance in OOP.
2. What is the difference between polymorphism in imperative languages like C and polymorphism in OOP like Java?
3. What is the difference between encapsulation and abstraction?