CS206 Introduction to Data Structures
Homework 5
Stacks

There are three parts to this homework. In the first part you will complete a partial implementation of an array-based stack. In the second and third parts you will use that stack to: investigate a game I call "Infinite Hopscotch", and simulate the UNDO function of a text editor.

Part 0

Create a new folder, presumably named Assignment5, and copy the following code into it.

Histogrammer.java
StackIntf.java
ArrayStackHW5.java
IHDice.java

All of the code is available on powerpuff in the directory /home/gtowell/Public/206/a5/. Use cp or scp as needed to copy this code to the directory containing the work for this assignment.

Part 1

Two functions (push and pop) in ArrayStackHW5.java need to be implemented. Do that.

Part 2

Simulating Undo in an editor

Editors use a stack for Undo. In this part of the assignment you will use the stack from part 1 to simulate typing with undo. Here is what you need to do:

Use scanner to read from the keyboard one line at a time.

With each character in the line:

```
if the character is a 1, UNDO the previous character.
    (if there is nothing to undo, do nothing

else if the character is a 9
    print the contents of the stack by popping each item then exit

else if the character is an upper or lower case letter add it to the stack

else if the character is an underscore '_' (ASCII value 95), add it to the stack
    we use _ instead of space because I can see _ and I can't see ' '.

else ignore all other characters.

if you run out of characters in a line before getting a '9', read another line
```

So for instance, if I entered

```
asdf_ghij1111jj8765439
```

then the output of the program would be

```
jj_fdsa
```

Given the input

```
qwert_yuio_hhh
1111abc19wwwww
```

The output of the program would be

```
baoiuy_trewq
```

The charAt function of String and an ASCII table will both be useful for this part.

Part 3

Infinite Hopscotch

The game of infinite hopscotch is my invention; it is played much like hopscotch. The idea is you have an N sided die (the singular of dice) where N is an even number. Two of the sides are labelled “0”, (N-2)/2 are labelled “1” and (N-2)/2 are labelled “2”. The game starts by rolling the die. If you get a 1, jump forward onto one foot; a 2 jump forward onto 2 feet. After jumping, roll the die again and make the jump appropriate jump. Repeat. If you roll a 0, turn around and exactly reverse the jumps you made on the way out.

The more sides the die has, the lower the probability of turning around on any given jump. (If you roll a zero on the first roll, your turn is over.) If you jump incorrectly at any time, or fail to exactly reverse your jumps, you loose.

Clearly the biggest challenge in infinite hopscotch is to keep track of your outbound jumps so you can exactly reverse yourself coming back.

The solution, use a stack!

To aid you in playing this game, I provide the class IHDice.java (Infinite Hopscotch Dice). **You must use this class.** To use it:

```
IHDice ihDice = new IHDice(26);
```

will make a new 26 sided die

```
int roll = ihDice.nextHop();
```

will return 0, 1 or 2 according the probabilities appropriate to the number of sides.

There are two sub-tasks for part 3.

1. Print the jumps made during 20 plays of the game with 26 sided die. For instance
   ```
   2 2 2 1 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2
   1 2 1 1 2 1
   etc
   ```
2. With a 100 sided die, run the program a lot (at least 1000 times), recording the number of hops made before turning around. Store that information in a way such that you can use the Histogrammer class to create a lo-res histogram of the frequency of the each number of hops made before turning around. (See the Histogrammer class for more details about its use.)

What to turn In

1. All code your wrote or edited, well commented
2. A file of containing the histogram from 3.2. (See the Histogrammer class for how to output directly to a file.)
3. A separate file with the 20 plays of a 26-sided die (3.1) This could be in your README
4. README with the usual information. In particular, be sure it is clear to me how to use your UNDO simulator.

The following steps for submission assume you created a project named `AssignmentN` in the directory `/home/YOU/cs206/` on the CS department UNIX computers (e.g. powerpuff)

- For this assignment N=5
- Put the README file into the project directory
- Go to the directory /home/YOU/cs206
- Enter `submit –c 206 –p N –d AssignmentN`

For more on using the submit script http://systems.cs.brynmawr.edu/Submit_assignments