```
// Lab 8: RunnyStack
// This lab made me crave cheese.
// Andrea Smith
// CSCI 1913

class RunnyStack<Base>
{

  // To make it run
  class Run
  {
    private Base base;
    private int length;
    private Run next;

    private Run(Base base, int length, Run next)
    {
      this.base = base;
      this.length = length;
      this.next = next;
    }
  }

  private Run top;
  private int depth = 0; // number of objects
  private int runs; // number of runs

  public RunnyStack() { }

// The value of depth
  public int depth()
  {
    return depth;
  }

// Checks if stack is empty
  public boolean isEmpty()
  {
    return top == null;
  }

// Returns the base at the top of the stack
  public Base peek()
  {
    if (isEmpty())
    {
      throw new IllegalStateException("The stack is empty.");
```

```
48      }
49      else
50      {
51        return top.base;
52      }
53    }
54
55  // Yeets a base off of the stack
56    public void pop()
57    {
58      if (isEmpty())
59      {
60        throw new IllegalStateException("The stack is empty.");
61      }
62
63      if (top.length > 1)
64      {
65        top.length--;
66      }
67      else
68      {
69        top = top.next; // If it hits zero, remove it
70        runs--;
71      }
72      depth--; // keep count of the depth, this must be outside of the
        if else statement or it screws everything, apparently
73    }
74
75  // Pushes a new base to the stack
76    public void push(Base base)
77    {
78      if (isEmpty())
79      {
80        top = new Run(base, 1, null); // Add a new run of one Base at
          the top of the stack. Null bc reasons
81        runs++;
82      }
83      else
84      {
85        if (top.base == base)
86        {
87          top.length++;
88        }
89        else
90        {
91          top = new Run(base, 1, top); // Add a new run of one Base at
```

```
          top of the stack, top because we don't want to reset, want
          to add on top of the previous one
          runs++;
        }
      }
      depth++; // I don't know if anyone will read this but THIS LINE
      RIGHT HERE GAVE ME SUCH A HEADACHE!! IT WAS JUST NESTED ONE TOO
      DEEP AND I SPENT SO LONG DEBUGGING AAA

    }

// Keeps track of the values of the runs
    public int runs()
    {
      if (top == null)
      {
        return 0;
      }
      else
      {
        return runs;
      }
    }

}

//
//  Tests for CSci 1913 Lab 8
//  James Moen
//  20 Mar 17
//
//  The TRY-CATCH statements catch exceptions thrown by RUNNY
STACK's methods,
//  so that the program can continue to run even if a method fails.
We still
//  haven't talked about TRY-CATCH'es in the lectures yet.
//
//  Most tests have comments that show what they should print, and
how many
//  points they are worth, for a total of 40 points.
//
//  Camembert is a soft French cheese. It may be runny. It can be
stacked.
//

class Camembert
{
```

```
131   public static void main(String [] args)
132   {
133     RunnyStack<String> s = new RunnyStack<String>();
134
135     System.out.println(s.isEmpty());         //  true       1 point
136     System.out.println(s.depth());           //  0          1 point
137     System.out.println(s.runs());            //  0          1 point
138
139     try
140     {
141       s.pop();
142     }
143     catch (IllegalStateException ignore)
144     {
145       System.out.println("No pop");          //  No pop     1 point
146     }
147
148     try
149     {
150       System.out.println(s.peek());
151     }
152     catch (IllegalStateException ignore)
153     {
154       System.out.println("No peek");         //  No peek    1 point
155     }
156
157     s.push("A");
158     System.out.println(s.peek());            //  A          1 point
159     System.out.println(s.depth());           //  1          1 point
160     System.out.println(s.runs());            //  1          1 point
161
162     System.out.println(s.isEmpty());         //  false      1 point
163
164     s.push("B");
165     System.out.println(s.peek());            //  B          1 point
166     System.out.println(s.depth());           //  2          1 point
167     System.out.println(s.runs());            //  2          1 point
168
169     s.push("B");
170     System.out.println(s.peek());            //  B          1 point
171     System.out.println(s.depth());           //  3          1 point
172     System.out.println(s.runs());            //  2          1 point
173
174     s.push("B");
175     System.out.println(s.peek());            //  B          1 point
176     System.out.println(s.depth());           //  4          1 point
177     System.out.println(s.runs());            //  2          1 point
```

```
178
179     s.push("C");
180     System.out.println(s.peek());           //  C          1 point
181     System.out.println(s.depth());          //  5          1 point
182     System.out.println(s.runs());           //  3          1 point
183
184     s.push("C");
185     System.out.println(s.peek());           //  C          1 point
186     System.out.println(s.depth());          //  6          1 point
187     System.out.println(s.runs());           //  3          1 point
188     s.pop();
189     System.out.println(s.peek());           //  C          1 point
190     System.out.println(s.depth());          //  5          1 point
191     System.out.println(s.runs());           //  3          1 point
192
193     s.pop();
194     System.out.println(s.peek());           //  B          1 point
195     System.out.println(s.depth());          //  4          1 point
196     System.out.println(s.runs());           //  2          1 point
197
198     s.pop();
199     System.out.println(s.peek());           //  B          1 point
200     System.out.println(s.depth());          //  3          1 point
201     System.out.println(s.runs());           //  2          1 point
202
203     s.pop();
204     s.pop();
205     System.out.println(s.peek());           //  A          1 point
206     System.out.println(s.depth());          //  1          1 point
207     System.out.println(s.runs());           //  1          1 point
208
209     s.pop();
210     System.out.println(s.isEmpty());        //  true       1 point
211     System.out.println(s.depth());          //  0          1 point
212     System.out.println(s.runs());           //  0          1 point
213
214     try
215     {
216       System.out.println(s.peek());
217     }
218     catch (IllegalStateException ignore)
219     {
220       System.out.println("No peek");        //  No peek    1 point
221     }
222   }
223 }
224
```