SHOWTIME

Java Developer Evaluation
Take-Home Development & Analysis Test

Job Title:	Java Developer
Interviewer:	Jason Pelzer (jason.pelzer@showtime.net)

If you feel there are any errors or omissions in this document, or have suggestions for improvements, please bring the issue to the attention of your interviewer.

Please attach the files individually to your response email and name the files:

- **ReplaceLastnameFirstname.java** (question 1)
- **solution.txt** (question 2)

Pre-Interview 'Homework':
I'm sorry, but I've got some Java homework for you. These questions are meant to be straightforward, but don't hesitate to ask if you have any questions. Reference books and Internet research are allowed unless otherwise specified, but please write ALL code yourself.

1. Please write a reversed string replacement method (**public String ecalpeResrever(…)**) that takes three String arguments, 'haystack,' 'needle,' and 'replacement'.

   The method will perform a left-to-right string replacement, followed by reversing the resultant strings *words*. ie. ("I like cats", "cat", "dog") → "dogs like I"

   The method must compile and run under JDK 1.5. The method will be tested for speed and correctness on haystacks up to one megabyte. There is a fast linear solution to this problem. Be aware of edge cases. Please leave all testing code intact so that we may evaluate your testing procedure. If you wish to use JUnit, you can create a separate class with 'Test' appended to the end of your primary class' name, and extending TestCase.

   Please create the containing class in the following package: **com.sho.hire.hw**, and please name the containing class: **ReplaceLastnameFirstname** and give it a default no-argument constructor.

   For example, my name "Jason Pelzer" would be **com.sho.hire.hw.ReplacePelzerJason**

   Example Test Cases (you should come up with more edge cases):

   ```
   ecalpeResrever("ABC", "A", "a").equals("aBC");
   ecalpeResrever("AAA AAB BAA", "AA", "a").equals("Ba aB aA");
   ecalpeResrever("I Work.", "Work", "Play").equals("Play. I");
   ecalpeResrever("Tests are the best!","the best!","just ok.").equals("ok. just are
   Tests");
   ```

2. The following code compiles, but produces unexpected results (sub-optimal) even under single-threaded conditions (hint: this is not a threading question). Please debug using any tools or techniques you have at your disposal to isolate the problem.

   The solution should simply be named 'solution.txt' and contain your analysis of the problem. You do not have to write code to fix the problem.

```
import java.util.*;

public class Stack {
    private Object[] elements;
    private int size = 0;

    public Stack(int initialCapacity) {
        this.elements = new Object[initialCapacity];
     }

     public void push(Object e) {
         ensureCapacity();
         elements[size++] = e;
     }

     public Object pop() {
         if (size==0)
           throw new EmptyStackException();
         Object pop = elements[--size];
         return pop;
     }

     /**
      * Ensure space for at least one more element, roughly
      * doubling the capacity each time the array needs to grow.
      */
     private void ensureCapacity() {
         if (elements.length == size) {
             Object[] oldElements = elements;
             elements = new Object[2 * elements.length + 1];
             System.arraycopy(oldElements, 0, elements, 0, size);
         }
     }
}
```