Metropolitan State University, St. Paul, MN
ICS 372 Object-Oriented Design and Implementation
Individual Assignment 3

1 Due Date and Goals

1.1 Due:

February 21, 2019

1.2 Goals:

1. To learn the adapter pattern.
2. To learn the Java coding standards.
3. To become more familiar with the Java API.

2 The Problem

Create, test, and document a Java class using the adapter pattern to implement the following interface.

```
/**
 * A PushbackableTokenizer allows the user to read
 * a token and push it back to the stream
 * from which the token was read. Tokens are
 * assumed to be separated by white space.
 * Any number of tokens may be pushed back.
 * @author Brahma Dathan
 *
 */
public interface PushbackableTokenizer {
/**
 * Returns the next token
 * @return the next token
 */
  public String nextToken();
/**
 * Returns true if and only if there are more tokens
 * @return true if there is at least one more token; else false
 */
  public boolean hasMoreTokens();
/**
 * The last token read and is not pushed back
 * is pushed back, so it can be read again using nextToken.
 */
```

```
  public void pushback();
}
```

Your task is to implement the above interface using the Adapter pattern. For this, use the `StringTokenizer` and `Stack` classes. You must also submit a driver program that creates an instance of the class and exercises all of its methods.

You can think of the tokens as of two types:

- the ones that have been supplied to the client reading the `PushbackableTokenizer` and were either not pushed back or were pushed back but then reread as many times as the number of pushbacks.
- the ones that have been supplied to the client reading the `PushbackableTokenizer` and have been pushed back (perhaps multiple times), but have not been read back after the last pushback.

For example, suppose the tokens `1, 2, 3, 4, 5, 6` were read and then the client pushes back `6`, `5` and then `4` and then rereads `4`. Then `1, 2, 3,` and `4` belong to the first category and `5` and `6` belong to the second category.

You must use a `Stack` object with the actual parameter `String` to store both types of tokens.

The implementation must be called `PushbackTokenizer`. It must have the signature given in the following skeleton of its only constructor.

```
public PushbackTokenizer(String data) {
}
```

Test your implementation using `assert` statements. Every method should be thoroughly tested. Document and lay out your code properly as specified under the coding standards document.

To elaborate on the functionality, assume that we create a `PushbackTokenizer` as below.

```
PushbackableTokenizer pushbackTokenizer =
      new PushbackTokenizer("Hello this is a test");
```

Consider some operations on the above object.

```
System.out.println(pushbackTokenizer.nextToken());
```

will print `"Hello"`.

```
System.out.println(pushbackTokenizer.nextToken());
```

will print `"this"`.

```
System.out.println(pushbackTokenizer.nextToken());
```

will print `"is"`.

```
pushbackTokenizer.pushback(); // pushes back "is"
pushbackTokenizer.pushback(); // pushes back "this"
```

Now

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "this".
The next

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "is".

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "a".

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "test".

```
pushbackTokenizer.pushback(); // pushes back test
pushbackTokenizer.pushback(); // pushes back a
```

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "a".

```
System.out.println(pushbackTokenizer.nextToken());
```

will print "test".

2.1 Important

There can be just two stacks in the `PushbackTokenizer` class and they must store only the tokens that have been supplied at least once to the client.

3 Documentation and Coding Conventions

Follow the requirements described in the coding standards document.

4 Submission

Submit the program as an Eclipse project. For this, do the following. (There is a demonstration video on this in D2L.)

1. In the Eclipse Project Explorer window, right click on the project.
2. Click on Export...
3. In the window for "Export," expand "General" and click on "Archive File." Click Next.
4. In the next window, browse to any folder of your choice and any file name you wish. Save the file as a zipped file.
5. Upload the zipped file to the dropbox for this assignment.

5 Grading

Your assignment will be graded as per the following distribution.

5.1 Correctness (20 points)

If you have a bad constructor, you will not receive any credit for this part. So be sure that the following code works.

```
PushbackableTokenizer pushbackTokenizer =
    new PushbackTokenizer("Hello this is a test");
```

I will check for the ability to push back multiple tokens.

Criterion	Points
Ability to create an object	2
Retrieve tokens	4
Push back a token and retrieve it	2
Push back multiple tokens retrieve them in the proper order	8
Check for end of tokens	4

5.2 Program Structure (10 points)

Including proper implementation as an object adapter, efficient use of the `Stack` and `StringTokenizer` classes, and proper use of generics.

5.3 Your testing using the driver program (5 points)

How well you have exercised the functionality.

5.4 Coding Standards (15 points)

Criterion	Points
Your name properly placed in all classes and interfaces	2
Meaningful documentation using /** and */ before all class and interfaces	3
Meaningful documentation using /** and */ before all methods	5
Coding conventions	5

Take a look at the rubrics for the assignment.

If you don't submit an Eclipse project as a zip file that opens correctly or is not executable directly, you will lose 10 percent of the credit.