Making class Stack “iterable”

Here is class Stack, with its method bodies elided because we do not have to look at them. The constructor and methods push(), pop(), and size() are there. Method iterator() is there. To the right is inner class StackIterator, with its constructor and methods hasNext() and next().

```
/** An instance is a stack */
public class Stack<E> {
    private E[] b;  // stack values are b[0..h-1],
    private int h;  // with b[h-1] at the top

    /** Constructor: stack of ≤ m values */
    public Stack(int m) {…}

    /** Push e onto the stack. if no room,
      * throw RuntimeException("no space") */
    public void push(E e) {…}

    /** Pop and return top stack value. Throw
      * EmptyStackException if stack empty. */
   public E pop() {…}

    /** = size of the stack */
    public int size() {…}

    /** = an Iterator over stack, top to bottom. */
    public Iterator<E> iterator() {…}
```

```
/** An instance: iterator over stack, top to bottom. */
private class StackIterator<E>
                              implements Iterator<E> {
   int n; // b[n] is next element to enumerate
   // (n == -1 means no more to enumerate)

/** Constructor: iterator over stack, top to bottom. */
public StackIterator() {…}

/** = there is another element to enumerate */
public @Override boolean hasNext() {…}

/** Return next element to enumerate. Throw
  * NoSuchElementException if no next element. */
public @Override E next() {…}
}
}
```

Let’s look at the API documentation for interface java.lang.Iterable. To find it, google “java 11 iterable” or use this URL: docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Iterable.html.

The description of this interface says:

> public interface **Iterable<T>**
> Implementing this interface allows an object to be the
> target of the "for-each loop" statement. See **For-each Loop**

So, we change the header of class Stack to implement Iterable:

```
public class Stack<E>  implements Iterable<E> {
```

Looking further down in the API documentation, we see that implementing this interface requires declaring method iterator():

```
/** Return an iterator over elements of type T */
public abstract Iterator<T> iterator()
```

But we have already written this method! Therefore, simply by saying that this class implements Iterable<E>, we have made it possible to use a *foreach* loop to enumerate the stack elements.

For example, we can write code to create and save a new Stack object, push three strings onto it, and use a foreach loop to print the stack elements from top to bottom.

```
Stack<String> st= new Stack<String>(50);
st.push("10"); st.push("6"); st.push("8");
for (String s: st) {
   System.out.println(s);
}
```

Executing this code will produce three lines containing the integers 8, 6, and 10.

Java 7 version of interface Iterable has only method iterator(). Java 8 and later versions has two other default methods, which we do not discuss and do not use.