Introduction to Computing II (ITI 1121) — Exercises

Marcel Turcotte

March 22, 2014

Students often come to my office with personal projects for which they need help. The exercises herein are largely based on these coding sessions with students. I hope that you will find these examples useful. Suggestions for exercises are always welcomed.

1 Stack

1.1 CountFilesWithStack

Write a method that takes as input a **File** object and returns as output 1 if the **File** object is an ordinary file, and the total number of files found in the sub-tree if the **File** object is a directory. Your method must use a stack to keep track of sub-directories that are waiting to be processed.

1.2 CountFiles

Re-write the method for counting the number of files in a sub-tree, but this time you must use recursion to keep track of the sub-directories that are waiting to be processed.

2 Queue

2.1 CircularLog

The context for this question is the development of a messaging application for a smart phone. Writing efficient code is paramount when developing for smart phones since excessive computations will drain the battery.

Here the problem was to write log system that would keep track of a fixed number entries and be efficient. For this, we propose to use fixed-size array and the circular array technique seen in class. Once the array is full, the method **log** writes over the oldest log entry. With this technique, **CircularLog** keeps track of the latest n log entries. Adding never requires moving any element. Finally, random access to entries is fast, thanks to the underlying array.

A **CircularLog** object has a method **size** that returns the number of entries currently stored (logical size). It also has a method **get**, where **get(0)** is the oldest entry still present in the **CircularLog**, and **get(size()-1)** is the newest.

```
CircularLog l;
l = new CircularLog(4);

l.log("For a variable of a primitive type, the value is found at the address designated by the identifier.");
l.log("True.");
l.log("A constructor has a return value and the same name as its class.");
l.log("False. A constructor has no return value.");
l.log("An abstract class contains only abstract methods.");
l.get(0); // returns "True."
```

Obviously, for a “real-world” application, the physical size of the **CircularLog** would be in the 100s or 1000s.

3 Iterator

3.1 CircularLogIterator

Implement an iterator for the class **CircularLog** (Question 2.1).

3.2 Iterable CircularLog

Make the class **CircularLog** from Question 2.1 **Iterable**, and write a test program illustrating the adapted **for** loop for **Iterable** data structures.

A Solutions

Question 1.1 CountFilesWithStack on page 1

```java
import java.util.Stack;
import java.io.File;

public class CountFilesWithStack {

    private static long countFiles(File in) {

        Stack<File> work = new Stack<File>();
        work.push(in);

        long count = 0;

        while (! work.isEmpty()) {

            File f = work.pop();

            if (f.isFile()) {
                count += 1;
            } else if (f.isDirectory()) {

                File[] files = f.listFiles();

                if (files != null) {
                    for (File g : files) {
                        work.push(g);
                    }
                }

            }
        }

        return count;
    }

    public static void main(String[] args) {

        if (args.length == 1) {
            System.out.println(countFiles(new File(args[0])));
        } else {
            System.out.println(countFiles(new File("/Users/turcotte/Sites")));
        }

    }

}
```

Question 1.2 CountFiles on page 1

```java
import java.io.File;

public class CountFiles {

    private static long countFiles(File in) {

        long count = 0;

        if (in.isFile()) {

            count = 1;

        } else if (in.isDirectory()) {

            File[] files = in.listFiles();

            if (files != null) {
                for (File f : files) {
                    count += countFiles(f);
                }
            }
        }

        return count;
    }

    public static void main(String[] args) {

        if (args.length == 1) {
            System.out.println(countFiles(new File(args[0])));
        } else {
            System.out.println(countFiles(new File("/Users/turcotte/Sites")));
        }

    }

}
```

Note: Re-implement using NIO.2 and compare speed.

Question 2.1 CircularLog on page 1

```
import java.util.Iterator;

public class CircularLog implements Iterable<String> {

    private static final int DEFAULT_CAPACITY = 100;

    private String[] lines;
    private int last;
    private int size;

    public CircularLog(int capacity) {
        lines = new String[capacity];
        last = 0;
        size = 0;
    }

    public CircularLog() {
        this(DEFAULT_CAPACITY);
    }

    public void log(String line) {
        lines[last] = line;
        last = (last + 1) % lines.length;
        if (size < lines.length) {
            size++;
        }
    }

    public int size() {
        return size;
    }

    public String get(int pos) {
        int index = (last + lines.length - size + pos) % lines.length;
        return lines[index];
    }
}
```

A.1 Question 3.1 CircularLogIterator on page 1

```java
import java.util.Iterator;

public class CircularLog {

    private static final int DEFAULT_CAPACITY = 100;

    private String[] lines;
    private int last;
    private int size;

    public CircularLog(int capacity) {
        lines = new String[capacity];
        last = 0;
        size = 0;
    }

    public CircularLog() {
        this(DEFAULT_CAPACITY);
    }

    public void log(String line) {
        lines[last] = line;
        last = (last + 1) % lines.length;
        if (size < lines.length) {
            size++;
        }
    }

    public int size() {
        return size;
    }

    public String get(int pos) {
        int index = (last + lines.length - size + pos) % lines.length;
        return lines[index];
    }

    private class LogIterator implements Iterator<String> {

        private int index;

        public LogIterator() {
            index = 0;
        }

        public boolean hasNext() {
            return index < size;
        }

        public String next() {
            int start = (last + lines.length - size) % lines.length;
            int current = (start + index) % lines.length;
            index++;
            return lines[current];
        }

        public void remove() {
            throw new UnsupportedOperationException();
        }

    }

    public Iterator<String> iterator() {
        return new LogIterator();
    }

}
```

A.2 Question 3.2 Iterable CircularLog on page 2

```java
import java.util.Iterator;

public class CircularLog implements Iterable<String> {

    private static final int DEFAULT_CAPACITY = 100;

    private String[] lines;
    private int last;
    private int size;

    public CircularLog(int capacity) {
        lines = new String[capacity];
        last = 0;
        size = 0;
    }

    public CircularLog() {
        this(DEFAULT_CAPACITY);
    }

    public void log(String line) {
        lines[last] = line;
        last = (last + 1) % lines.length;
        if (size < lines.length) {
            size++;
        }
    }

    public int size() {
        return size;
    }

    public String get(int pos) {
        int index = (last + lines.length - size + pos) % lines.length;
        return lines[index];
    }

    private class LogIterator implements Iterator<String> {

        private int index;

        public LogIterator() {
            index = 0;
        }

        public boolean hasNext() {
            return index < size;
        }

        public String next() {
            int start = (last + lines.length - size) % lines.length;
            int current = (start + index) % lines.length;
            index++;
            return lines[current];
        }

        public void remove() {
            throw new UnsupportedOperationException();
        }

    }

    public Iterator<String> iterator() {
        return new LogIterator();
    }

}
```

```
public class TestCircularLog {

    public static void main(String[] args) {

        CircularLog l;
        l = new CircularLog(8);

        for (int i = 0; i < 24; i++) {
            l.log("Line-" + i);
            System.out.println("Iterator");
            for (String s : l) { // for over Iterable object
                System.out.println(s);
            }
            System.out.println("get");
            for (int j=0; j<l.size(); j++) {
                System.out.println(l.get(j));
            }
            System.out.println();
        }

    }
}
```