ECU178 Computer Science: 210CT - Programming, Algorithms and Data Structures Portfolio

Due on Monday, December 15th, 2014

Dr James Shuttleworth

Robert Rigler : 4939377

Contents

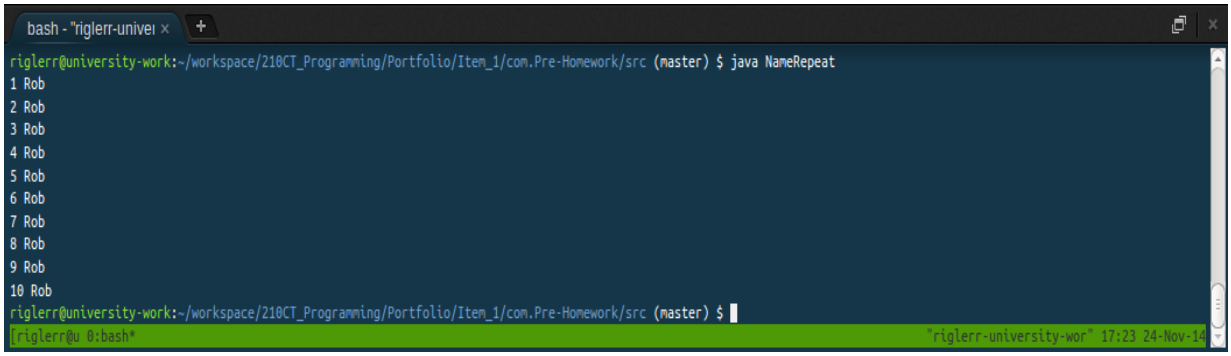
Item 1: Week 3 - Linear Search and Duplicate Finder

Pre-Homework 1: Write a Program that displays your name 10 times

Listing 1: NameReapeat class JAVA code

```
/**
 * Created by Rob on 23/10/2014.
 */
public class NameRepeat {

    public static void main(String[] args){

        NameRepeat myObject = new NameRepeat();  /*Create Object*/

     /*Use object to call PrintName() method*/
        myObject.PrintName("Rob");
    }

    public void PrintName(String _name){

        for (int i = 0; i<10;i++){  /* Loop 10 times*/

     /*, print the number && _name parameter each time.*/
            System.out.println((i+1) + " " + _name);


        }


    }
}
```

Evidence

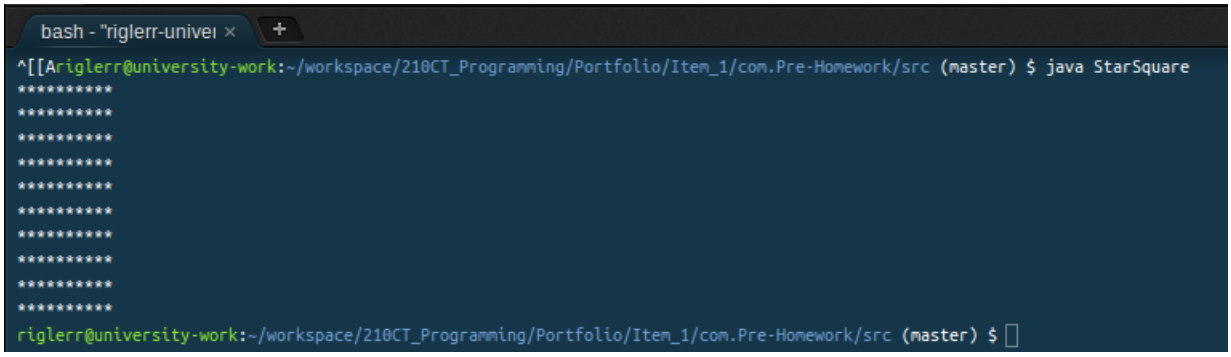

bash - "riglerr-univer
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $ java NameRepeat
1 Rob
2 Rob
3 Rob
4 Rob
5 Rob
6 Rob
7 Rob
8 Rob
9 Rob
10 Rob
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $
[riglerr@u 0:bash*
"riglerr-university-wor" 17:23 24-Nov-14

Pre-Homework 2: Write a function that draws a square of stars given as a parameter

Listing 2: StarSquare class JAVA code

```
/**
 * Created by Rob on 23/10/2014.
 */
public class StarSquare {

     //Create variable to hold asterisk character.
    public char ast = '*';
    public static void main(String[] args){

        StarSquare sSquare = new StarSquare(); /*Create Object*/

     /*Use object to call writeSquare() method*/
        sSquare.writeSquare(10);

    }


    public void writeSquare(int size){

        for(int i =0; i<size;i++){ /*OuterLoop 'size' times*/
            for(int j = 0;j<size;j++){ /*InnerLoop 'size' times*/

                System.out.print(ast); /*Print line of asterisks*/

            }

          /* Start new line when inner loop finishes*/
            System.out.println();
        }
    }
}
```

Evidence

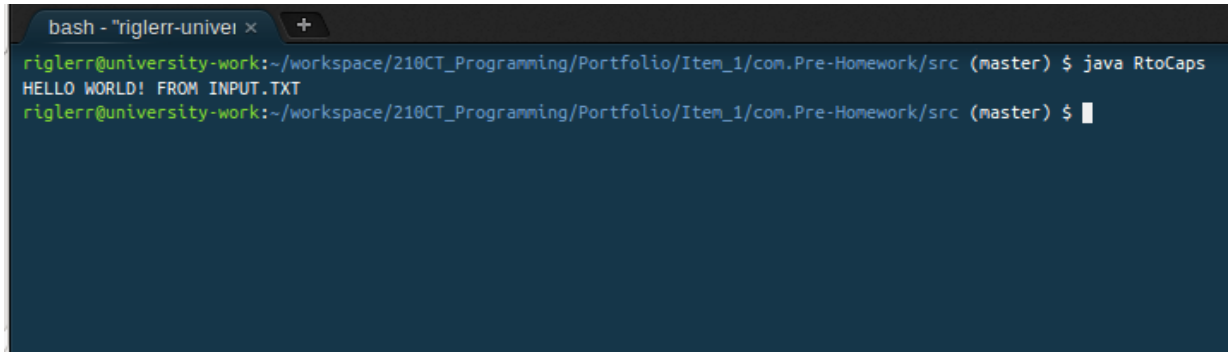

bash - "riglerr-univer
^[[Ariglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $ java StarSquare
****
****
****
****
****
****
****
****
****
****
riglerr@university-work:~/workspace/210CT_Programming/Portfolio/Item_1/com.Pre-Homework/src (master) $

Pre-Homework 3: Write a program to open a file and display it's contents in capitals

Listing 3: RtoCaps class JAVA code

```
/**
 * Created by Rob on 23/10/2014.
 */
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class RtoCaps {

 public static void main(String[] args)throws FileNotFoundException
{
        File inFile = new File("input.txt");
                /*Create a file object */
        RtoCaps obj = new RtoCaps(); /*Create class object*/

     /*Use Class object to call rInput() method*/
        obj.rInput(inFile);
    }

    public void rInput(File inFile) throws FileNotFoundException{

     /*Create a new scanner to read from the file*/
        Scanner in = new Scanner(inFile);

     /*Loop WHile there is still lines left in the document*/
        while(in.hasNextLine())
        {
          /* Place the next line in a strin varibale*/
            String line = in.nextLine();

           /* Print the line in uppercase*/
            System.out.println(line.toUpperCase());
        }

    }


}
```

Evidence



1. Pseudocode for linear search

Algorithm 1 LinearSearch

```
1: procedure BOOL LINEARSEARCH(item, list[ ])
2:     for each element i in list do
3:         if list[i] = item then
4:             return true
5:         end if
6:     end for
7: return false
8: end procedure
```

2. Pseudocode for finding duplicates in a list

Algorithm 2 Examining for duplicates

```
1: procedure BOOL EXFORDUPES(list[ ])
2:     for each element i in list[ ] do
3:         for each element j in list[ ] do
4:             if list[i] = list[j] then
5:                 return true
6:             end if
7:         end for
8:     end for
9: end procedure
```

continued on next page...

Item 2: Week 4 - Time complexities and Big-O notation

1. Describe the runtime bounds of the linear search algorithm

Algorithm 3 LinearSearch

```
1: procedure BOOL LINEARSEARCH(item, list[ ])
2:
3:     for each element i in list do          (n)
4:         if  list[i] = list then t          (n)
5:            return true                     (n)
6:         end if
7:     end for
8: return false                               (1)
9: end procedure
```

The time complexity of the algorithm is O(n)

2. Describe the runtime bounds of the duplicate finder algorithm

Algorithm 4 Examining for duplicates

```
1: procedure BOOL EXFORDUPES(list[ ])
2:     for  each element i in list[ ] do          (n)
3:         for  each element j in list[ ] do      (n*n)
4:             if  list[i] = list[j] then         (n*n)
5:            return true                         (n*n)
6:             end if
7:         end for
8:     end for
9: return false                                   (1)
10: end procedure
```

The time complexity of the algorithm is $O(n^2)$

Additional work: Critical values of relative runtimes

Write a function that determines the critical value at which the relative runtime of two linear algorithms swap.

Algorithm 5 Relative runtime comparison algorithm

```
1: procedure CRITVAL(m1, k1, m2, k2)
2:     switch ← false
3:     n ← 0
4:
5:     // Which Expression has a greater value for n=0
6:     if (((m1 * n) + k1) > ((m2 * n) + k2)) then
7:
8:         //While Expression 1 (m1, k1) is greater than Expression 2(m2, k2), do:
9:         while !switch do
10:
11:            // If Expression 1 become less than Expression 2 for that value of n
12:            if (((m1 * n) + k1) < ((m2 * n) + k2)) then
13:
14:                //switch becomes true, which exits the loop and both if statements
15:                switch ← true
16:            else
17:                n + +
18:            end if
19:        end while
20:
21:    else
22:
23:        //While Expression 2 (m1, k1) is greater than Expression 1(m2, k2), do:
24:        while !switch do
25:
26:            // If Expression 2 become less than Expression 1 for that value of n
27:            if (((m1 * n) + k1) > ((m2 * n) + k2)) then
28:
29:                //switch becomes true, which exits the loop and both if statements
30:                switch ← true
31:            else
32:                n + +
33:            end if
34:        end while
35:
36:    end if
37:    // Return the value of n at which either while loop was fulfilled.
38: return n
39:
40: end procedure
```

Item 3: Week 6 - Harmonic Series or Pivot Selection

Pre-Homework 1.: 3-Bit binary number

Write a function that takes 3 boolean parameters, a, b and c and returns an integer value they represent if they are the three bits of a three-bit number, with a being the most significant and c being the least.

Algorithm 6 3-Bit Binary Number

```
1: procedure PRE1(bool a, bool b, bool c)
2:     total ← 0
3:     if a then
4:         total ← total + 1
5:     end if
6:     if b then
7:         total ← total + 2
8:     end if
9:     if c then
10:        total ← total + 4
11:    end if
12: return total
13: end procedure
```

Pre-Homework 2.: Boolean sequence binary number

Write a function that takes a sequence of values of any given length and returns the integer value they represent.

Algorithm 7 Return integer value from list a of boolean value

```
1: procedure PRE2(bool list[ ])
2:     total ← 0
3:     len ← LengthOf.list[ ]
4:     for  i ← len to 0 do
5:         if list[i] = true then
6:             total ← total + 2^i
7:         end if
8:     end for
9: end procedure
```

1. Harmonic Series (Pseudocode)

Use pseudocode to specify a recursive algorithm to compute the nth value of the harmonic series, for some integer n.

2. Harmonic Series (JAVA Implementation)

The Harmonic Series computation algorithm implemented in Java

Item 4: Week 7 - Heapworksheet or RPN Calculator

Item 5: Week 8 - Linked List Delete function or Linked List Sortings