

**Reza Shisheie**  
**Homework #3**  
**Due: March 30<sup>th</sup> 2017**

# Question 1:

```
package homework3_1;

import java.util.*;

public class homework3_1 {
    public static void main(String[] args) {

        List<Integer> arrayList = new ArrayList<>();
        int n = 100000;
        for (int i = 0; i<n; i++){
            arrayList.add(1); // 1 is autoboxed to new Integer(1)
        }

        // Linked list traverse using iterator
        long time_1 = System.currentTimeMillis();
        LinkedList<Object> linkedList = new LinkedList<>(arrayList);
        ListIterator<Object> listIterator = linkedList.listIterator();
        for (int i = 0; i<n; i++){
            listIterator.next();
        }
        long time_2 = System.currentTimeMillis();
        System.out.print("The time for traversing using iterator is: ");
        System.out.print(time_2 - time_1);
        System.out.println(" milli seconds");

        // Linked list traverse using get
        time_1 = System.currentTimeMillis();

        for (int i = 0; i<n; i++){
            linkedList.get(i);
        }
        time_2 = System.currentTimeMillis();
        System.out.print("The time for traversing using get is: ");
        System.out.print(time_2 - time_1);
        System.out.println(" milli seconds");
    }
}
```

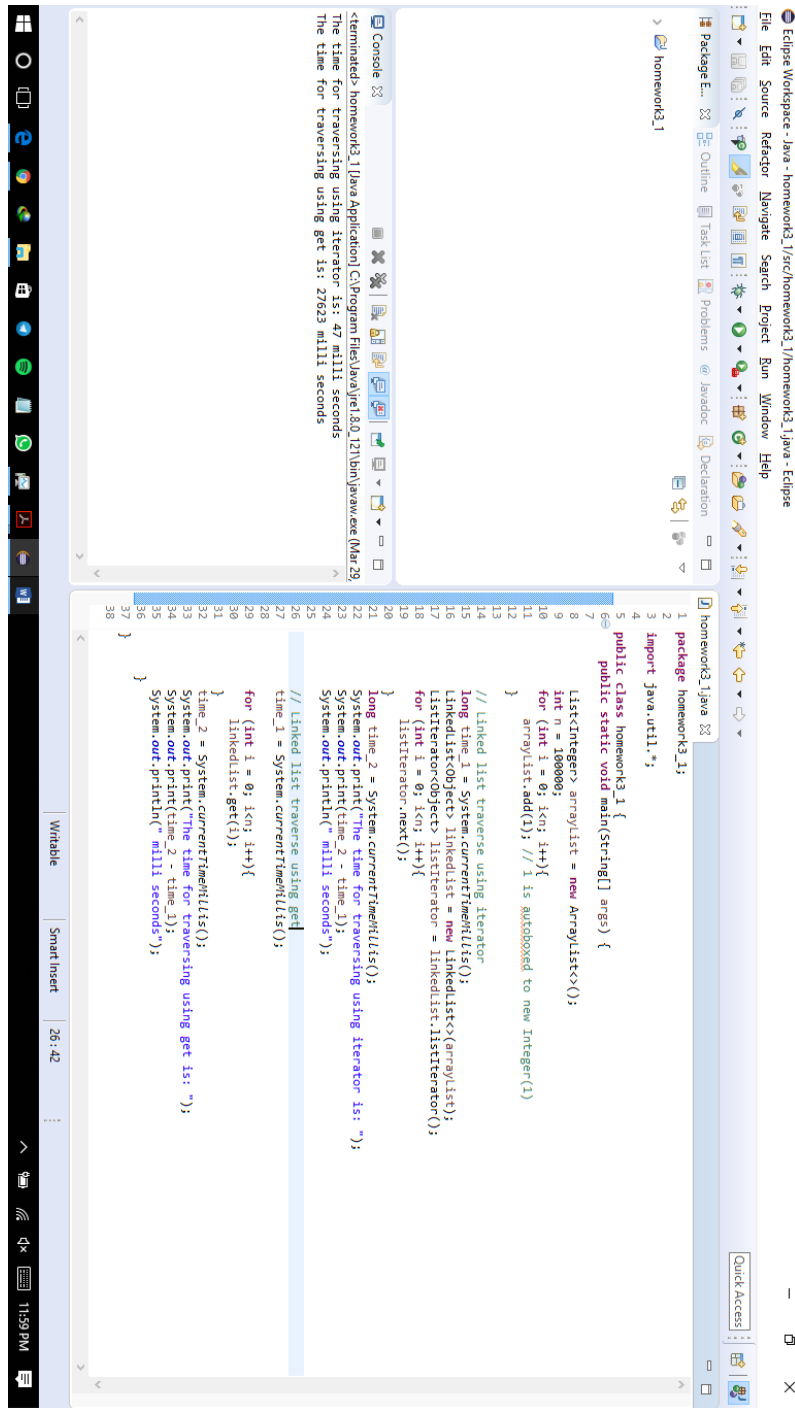
What this question is asking is about how long it takes to print all data of a linked list if we use iterator versus get. If I use iterator it starts from the first element and iterates to the end and prints all of that one by one. However, every single time I trigger get, it starts from the first element and iterates all the way to the target and only prints the target. In other words, as the input element of the get gets bigger, it takes more time to print all data.

The Following simulation is for 100,000 data since it took very long to simulate for 5,000,000 data.

# Question 1 Output:

he time for traversing using iterator is: 47 milli seconds

The time for traversing using get is: 27623 milli seconds



The screenshot shows the Eclipse IDE with a Java project named 'homework3\_1'. The main editor displays the source code for 'homework3\_1.java'. The code defines a 'LinkedList' class with a 'main' method that creates a linked list of 100,000 integers. It then measures the time taken to traverse the list using two different methods: 'iterator' and 'get'. The console output shows that the 'iterator' method takes 47 milliseconds, while the 'get' method takes 27623 milliseconds.

```
1 package homework3_1;
2 import java.util.*;
3
4 public class homework3_1 {
5     public static void main(String[] args) {
6         // Create a linked list
7         List<Integer> arraylist = new ArrayList<>();
8         int n = 100000;
9         for (int i = 0; i < n; i++) {
10             arraylist.add(i); // i is autoboxed to new Integer(i)
11         }
12
13         // Linked list traverse using iterator
14         long time_1 = System.currentTimeMillis();
15         LinkedList<Integer> linkedlist = new LinkedList<>(arraylist);
16         ListIterator<Integer> listiterator = linkedlist.listIterator();
17         for (int i = 0; i < n; i++) {
18             listiterator.next();
19         }
20         long time_2 = System.currentTimeMillis();
21         System.out.println("The time for traversing using iterator is: ");
22         System.out.println(time_2 - time_1);
23         System.out.println(" milli seconds");
24
25         // Linked list traverse using get
26         time_1 = System.currentTimeMillis();
27         for (int i = 0; i < n; i++) {
28             linkedlist.get(i);
29         }
30         time_2 = System.currentTimeMillis();
31         System.out.println("The time for traversing using get is: ");
32         System.out.println(time_2 - time_1);
33         System.out.println(" milli seconds");
34     }
35 }
36
37
38
```

Console Output:

```
<terminated> homework3_1 [Java Application] G:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Mar/29/2016 11:59:42 AM)
The time for traversing using iterator is: 47 milli seconds
The time for traversing using get is: 27623 milli seconds
```

## Question 2:

```
package homework3_2;
import java.util.*;

public class homework3_2 {
    public static void main(String[] args) {
        PriorityQueue<String> queue1 = new PriorityQueue<>();
        queue1.offer("George");
        queue1.offer("Jim");
        queue1.offer("John");
        queue1.offer("Blake");
        queue1.offer("Kevin");
        queue1.offer("Michael");

        PriorityQueue<String> queue2 = new PriorityQueue<>();
        queue2.offer("George");
        queue2.offer("Katie");
        queue2.offer("Kevin");
        queue2.offer("Michelle");
        queue2.offer("Ryan");

        PriorityQueue<String> Union = new PriorityQueue<>();
        PriorityQueue<String> Difference = new PriorityQueue<>();
        PriorityQueue<String> Intersection = new PriorityQueue<>();

        String A = "";
        String B = "";

        // While loop until one queue is ended
        while (queue1.size() > 0 && queue2.size() > 0 ) {
            A = queue1.peek();
            B = queue2.peek();
            //System.out.print(A + " ");
            //System.out.println(B);
            if( A.compareTo(B) == 0){
                Intersection.offer(A);
                Union.offer(A);
                queue1.remove();
                queue2.remove();
            } else if (A.compareTo(B) < 0){
                Union.offer(A);
                Difference.offer(A);
                queue1.remove();
            } else {
                Union.offer(B);
                Difference.offer(B);
                queue2.remove();
            }
        }

        // While for the loop with remaining data
```

```

while (queue1.size() > 0){
    A = queue1.remove();
    Union.offer(A);
    Difference.offer(A);
}

// While for the loop with remaining data
while (queue2.size() > 0){
    B = queue2.remove();
    Union.offer(B);
    Difference.offer(B);
}

System.out.print("List of Unions: ");
while (Union.size() > 0) {
    System.out.print(Union.remove() + " ");
}
System.out.println("");

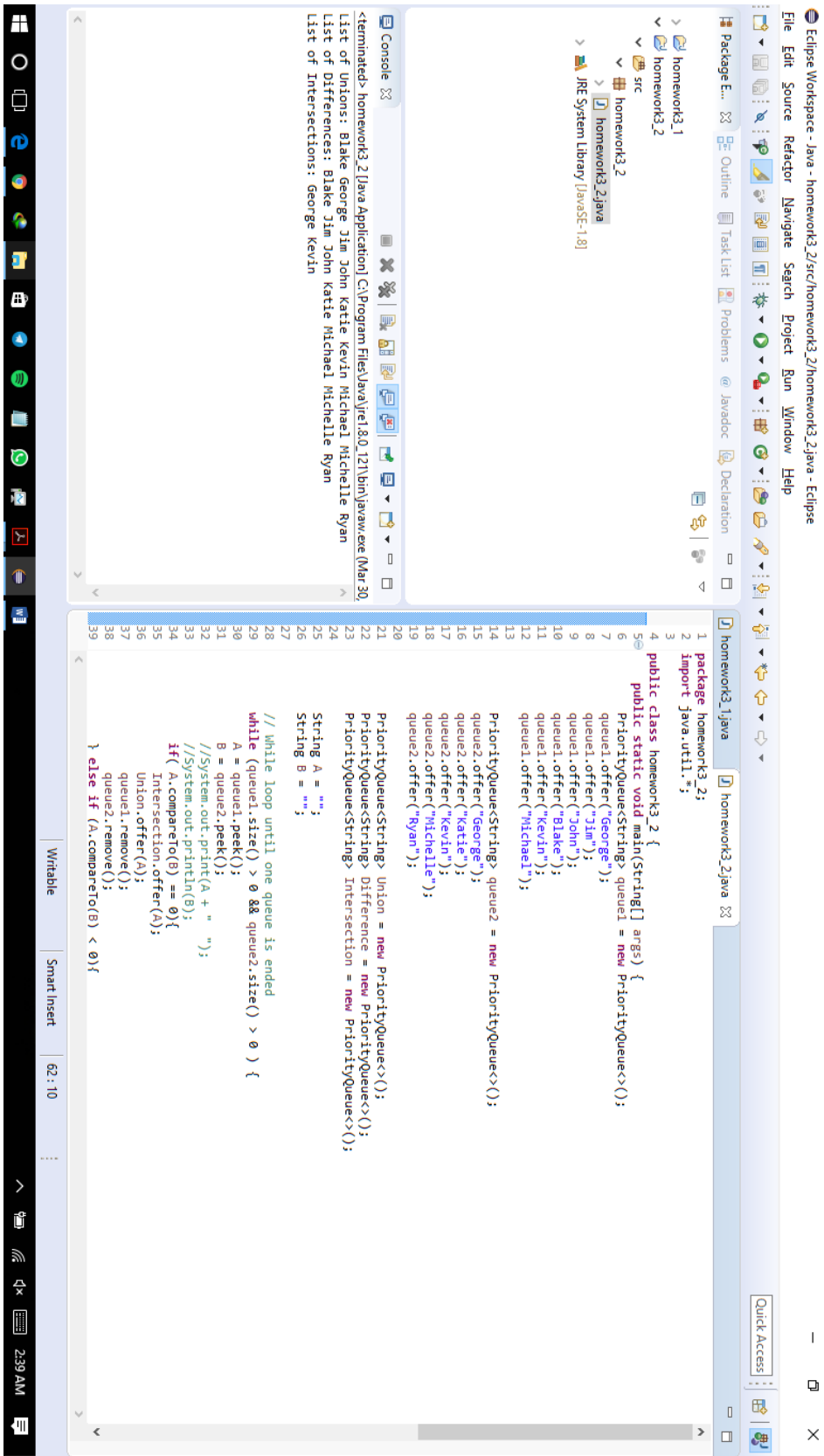
System.out.print("List of Differences: ");
while (Difference.size() > 0) {
    System.out.print(Difference.remove() + " ");
}
System.out.println("");

System.out.print("List of Intersections: ");
while (Intersection.size() > 0) {
    System.out.print(Intersection.remove() + " ");
}
System.out.println("");
}
}

```

## Question 2 Output:

List of Unions: Blake George Jim John Katie Kevin Michael Michelle Ryan  
 List of Differences: Blake Jim John Katie Michael Michelle Ryan  
 List of Intersections: George Kevin



## Question 3:

```
package homework3_3;
import java.util.*;
import java.io.*;

public class homework3_3 {
    public static void main(String[] args) throws Exception {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a Java source file: ");
        String filename = input.nextLine();
        // C:\Users\ARSH\Box Sync\Personal\Eclipse
        Workspace\homework3_3\bin\homework3_3\text.txt

        File file = new File(filename);
        if (file.exists()) {
            //System.out.println("The number of keywords in " + filename
            + " is " + countKeywords(file));
            int [] result = countKeywords(file);
            System.out.println("The number of vowels and consonants in the
            file are " + result[0] + " and " + result[1] );
        }
        else {
            System.out.println("File " + filename + " does not exist");
        }
    }

    public static int[] countKeywords(File file) throws Exception {
        String[] keywordString = {"A", "E", "I", "O", "U", "a", "e", "i", "o",
        "u" };

        Set<String> keywordSet = new HashSet<>(Arrays.asList(keywordString));
        int count_vowel = 0;
        int count_consonant = 0;
        Scanner input = new Scanner(file);
        System.out.println("The string in the file are:");

        while (input.hasNext()) {
            String word = input.next();
            System.out.println(word + " ");
            String [] new_word = word.split("");

            for (int i = 0; i<word.length();i++){
                if (keywordSet.contains(new_word[i])){
                    count_vowel++;
                }
                else {
                    count_consonant++;
                }
            }
        }
        int[] result = {count_vowel, count_consonant};
        return result;
    }
}
```

# Question 3 Output:

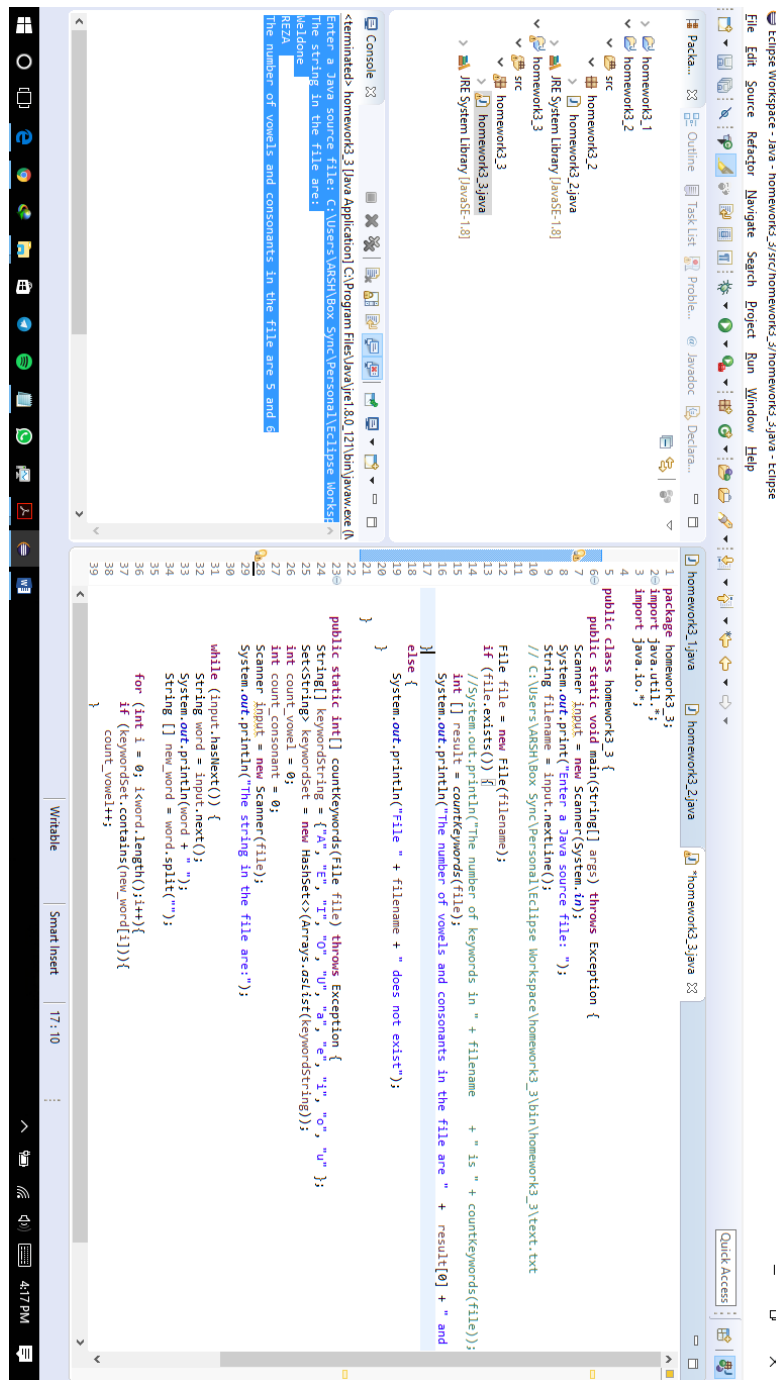
Enter a Java source file: C:\Users\ARSH\Box Sync\Personal\Eclipse Workspace\homework3\_3\bin\homework3\_3\text.txt

The string in the file are:

Weldone

REZA

The number of vowels and consonants in the file are 5 and 6



The screenshot shows the Eclipse IDE interface. The main editor displays a Java file named `homework3_3.java` with the following code:

```
1 package homework3_3;
2 import java.util.*;
3 import java.io.*;
4
5 public class homework3_3 {
6     public static void main(String[] args) throws Exception {
7         Scanner input = new Scanner(System.in);
8         System.out.print("Enter a Java source file: ");
9         String filename = input.nextLine();
10        // C:\Users\ARSH\Box Sync\Personal\Eclipse Workspace\homework3_3\bin\homework3_3\text.txt
11
12        File file = new File(filename);
13        if (file.exists()) {
14            //System.out.println("The number of keywords in " + filename
15            int [] result = countKeywords(file);
16            System.out.println("The number of vowels and consonants in the file are " + result[0] + " and
17
18            }
19        } else {
20            System.out.println("File " + filename + " does not exist");
21        }
22    }
23
24    public static int[] countKeywords(File file) throws Exception {
25        String[] keywordString = {"a", "e", "i", "o", "u", "A", "E", "I", "O", "U"};
26        Set<String> keywordSet = new HashSet<>(Arrays.asList(keywordString));
27        int count_vowel = 0;
28        int count_consonant = 0;
29        Scanner input = new Scanner(file);
30        System.out.println("The string in the file are:");
31
32        while (input.hasNext()) {
33            String word = input.next();
34            System.out.println(word + " ");
35            String [] new_word = word.split(" ");
36            for (int i = 0; i<new_word.length;i++){
37                if (keywordSet.contains(new_word[i])){
38                    count_vowel++;
39                }
40            }
41        }
42    }
43 }
```

The console window at the bottom shows the following output:

```
<terminated> homework3_3 [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\java.exe(A
Enter a Java source file: C:\Users\ARSH\Box Sync\Personal\Eclipse Workspace\homework3_3\bin\homework3_3\text.txt
The string in the file are:
Weldone
REZA
The number of vowels and consonants in the file are 5 and 6
```



## Question 4:

```
package homework3_4;
import java.util.*;

public class homework3_4 {
    public static void main(String[] args) {
        Map<String, String> linkedHashMap = new LinkedHashMap<>(4, 0.75f,
true);

        linkedHashMap.put("Ohio", "Columbus");
        linkedHashMap.put("West Virginia", "Charleston");
        linkedHashMap.put("California", "Sacramento");
        linkedHashMap.put("Texas", "San Antonio");

        Set set = linkedHashMap.entrySet();
        Iterator i = set.iterator();

        Scanner input = new Scanner(System.in);
        int correct = 0;
        int notcorrect = 0;

        while(i.hasNext()) {
            Map.Entry me = (Map.Entry)i.next();
            String State = (String)me.getKey();
            String Capital = (String)me.getValue();
            System.out.println("Enter the capital of " + State + ":" );
            String new_capital = input.nextLine();

            if (Capital.equalsIgnoreCase(new_capital)){
                System.out.print("Bravo! ");
                correct++;
            } else {
                System.out.print("No Correct! ");
                notcorrect++;
            }
            System.out.println("The capital of " + State + " is " + Capital );
        }
        System.out.println("The number of correct answers are: " + correct);
        System.out.println("The number of false answers are: " + notcorrect);
    }
}
```

# Question 4 Output:

Enter the capital of Ohio:

columbus

Bravo! The capital of Ohio is Columbus

Enter the capital of West Virginia:

CHARLESTON

Bravo! The capital of West Virginia is Charleston

Enter the capital of California:

SaCRAMenTO

Bravo! The capital of California is Sacramento

Enter the capital of Texas:

Huoston

No Correct! The capital of Texas is San Antonio

The number of correct answers are: 3

The number of false answers are: 1

The screenshot shows the Eclipse IDE with a project named 'homework3\_4'. The package explorer on the left shows the project structure. The main editor displays the source code for 'homework3\_4.java'. The code defines a 'main' method that uses a 'LinkedHashMap' to store state-capital pairs. It iterates through these pairs, prompts the user for the capital of each state, and checks if the input matches the stored capital (case-insensitive). It keeps track of the number of correct and incorrect answers. The console on the bottom left shows the execution output, which matches the text provided in the previous blocks. The status bar at the bottom indicates the file is 'Writable', 'Smart Insert' is active, and the cursor is at line 16, column 10.

```
1 package homework3_4;
2 import java.util.*;
3
4 public class homework3_4 {
5     public static void main(String[] args) {
6         Map<String, String> linkedHashMap = new LinkedHashMap<>(4, 0.75f, true);
7         linkedHashMap.put("Ohio", "Columbus");
8         linkedHashMap.put("West Virginia", "Charleston");
9         linkedHashMap.put("California", "Sacramento");
10        linkedHashMap.put("Texas", "San Antonio");
11
12        Set set = linkedHashMap.entrySet();
13        Iterator i = set.iterator();
14
15        Scanner input = new Scanner(System.in);
16        int correct = 0;
17        int notcorrect = 0;
18
19        while(i.hasNext()) {
20            Map.Entry me = (Map.Entry)i.next();
21            String State = (String)me.getKey();
22            String Capital = (String)me.getValue();
23            System.out.println("Enter the capital of " + State + ":");
24            String new_capital = input.nextLine();
25
26            if (Capital.equalsIgnoreCase(new_capital)){
27                System.out.print("Bravo! ");
28                correct++;
29            } else {
30                System.out.print("No Correct! ");
31                notcorrect++;
32            }
33            System.out.println("The capital of " + State + " is " + Capital );
34        }
35        System.out.println("The number of correct answers are: " + correct);
36        System.out.println("The number of false answers are: " + notcorrect);
37    }
38 }
```

Console Output:

```
<terminated> homework3_4 [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (h
Enter the capital of Ohio:
Columbus
Bravo! The capital of Ohio is Columbus
Enter the capital of West Virginia:
CHARLESTON
Bravo! The capital of West Virginia is Charleston
Enter the capital of California:
SaCRAMenTO
Bravo! The capital of California is Sacramento
Enter the capital of Texas:
Huoston
No Correct! The capital of Texas is San Antonio
The number of correct answers are: 3
The number of false answers are: 1
```

# Question 5:

## 5.1. Greater Common Divisor (GCD) using Euclidean Method:

What is essential about the Euclid method is that it works recursively and invokes itself until it hits 1. Every time it takes the GCD of  $n$  and the remainder of " $m \% n$ " which happens " $k$ " times.

The remainder of two number can't be greater than the half of the largest one which can be show as follows

$$10 \% 1 = 0$$

$$10 \% 2 = 0$$

$$10 \% 3 = 1$$

$$10 \% 4 = 2$$

$$10 \% 5 = 0$$

$$10 \% 6 = 4$$

$$10 \% 7 = 3$$

$$10 \% 8 = 2$$

$$10 \% 9 = 1$$

which are all less than  $n / 2 = 5$ .

Thus every time the " $m \% n$ " is invoked, the value gets halved.

$$\text{gcd}(m, n)$$

$$\text{gcd}(n, m \% n)$$

$$\text{gcd}(m \% n, n \% (m \% n))$$

as shown above every two iteration the first value gets halved thus the value of " $k$ " should be  $\frac{n}{2^{k/2}} \leq 1$ , which is  $k \geq 2 \log n$

Thus the time complexity is  $O(\log n)$

## 5.2. prime number using Sieve of Eratosthenes:

The Sieve of Eratosthenes's method eliminates the products of all numbers with itself in an increasing order to the end and counts the remaining unaffected number as prime numbers. If the number is  $n$ , the method gets invoked  $n/2$  times for 2,  $n/3$  times for 3 and so on, thus the complexity is:

$$O(n/2 + n/3 + n/5 + n/11 + \dots) = n * O(1/2 + 1/3 + 1/5 + 1/11 + \dots) \leq O(nF(n)) = O\left(\frac{n\sqrt{n}}{\log n}\right)$$