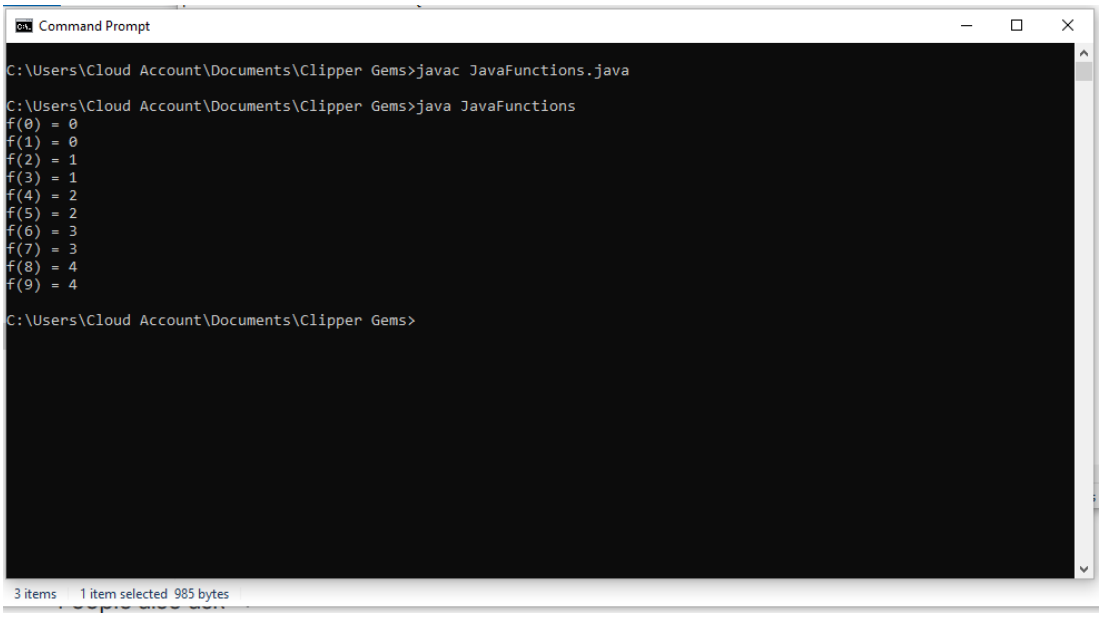
Activity: Java functions (1)

Type the following program in a file named “JavaFunctions.java”,
Then compile and run it. What does the program do?
Observe the output of this program. Is the output correct?

- It gets the f(x), but it’s not accurate because you are using an integer which only results to whole numbers



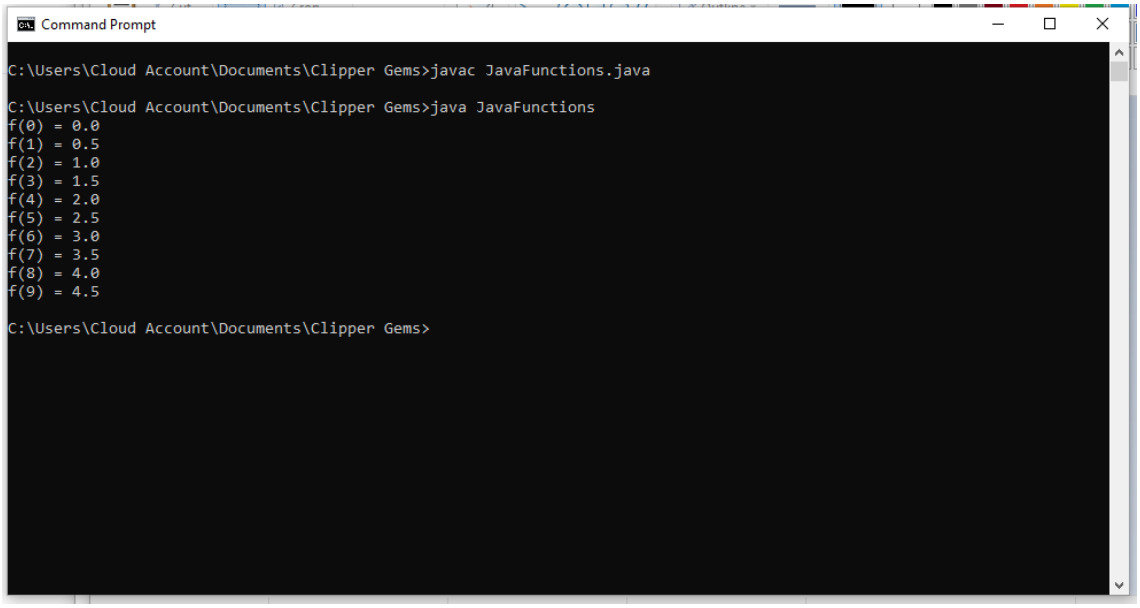
Activity: Java Function (2)

The previous program does not appear to give correct results: Change it so that it will appear to work correctly.

The changes I made in the code are:

- Changing the data type of the function "JavaFunctions" from int to float. This will then return a floating value to give an accurate result.
- The second thing I changed is the constant number 2 to 2f which makes it a floating value. Simply having an integer divided by another integer will give an integer result. So, to get an accurate result, I change the data type to float to get an accurate answer.

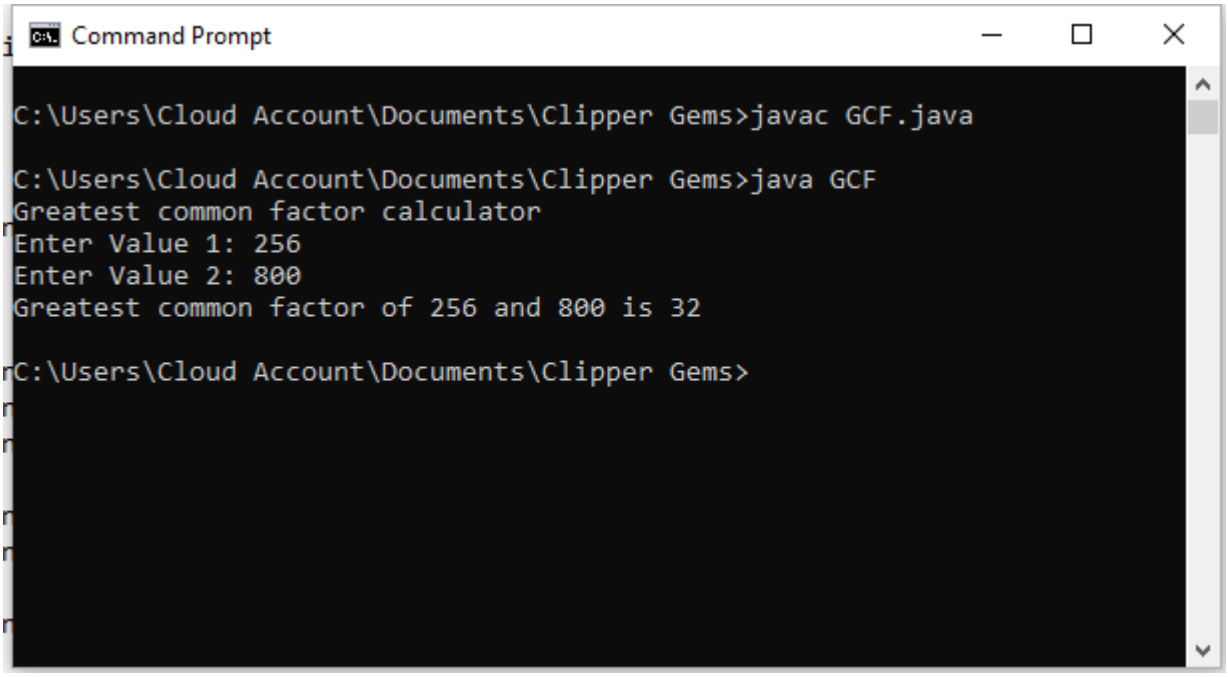
```
public class JavaFunctions{
        static float myFunction(int x){
                return (x/2f);
        }

        public static void main(String[] args){
                for(int x = 0; x < 10; x++){
                        System.out.println("f(" +  x + ") = " + myFunction(x));
                }
        }
}
```



Activity: GCF

- Create a java program that accepts two numbers.
- These two numbers are supplied as a parameter to a method.
- The method then returns the greatest common factor.
- Display the output like so: Greatest common factor of and is

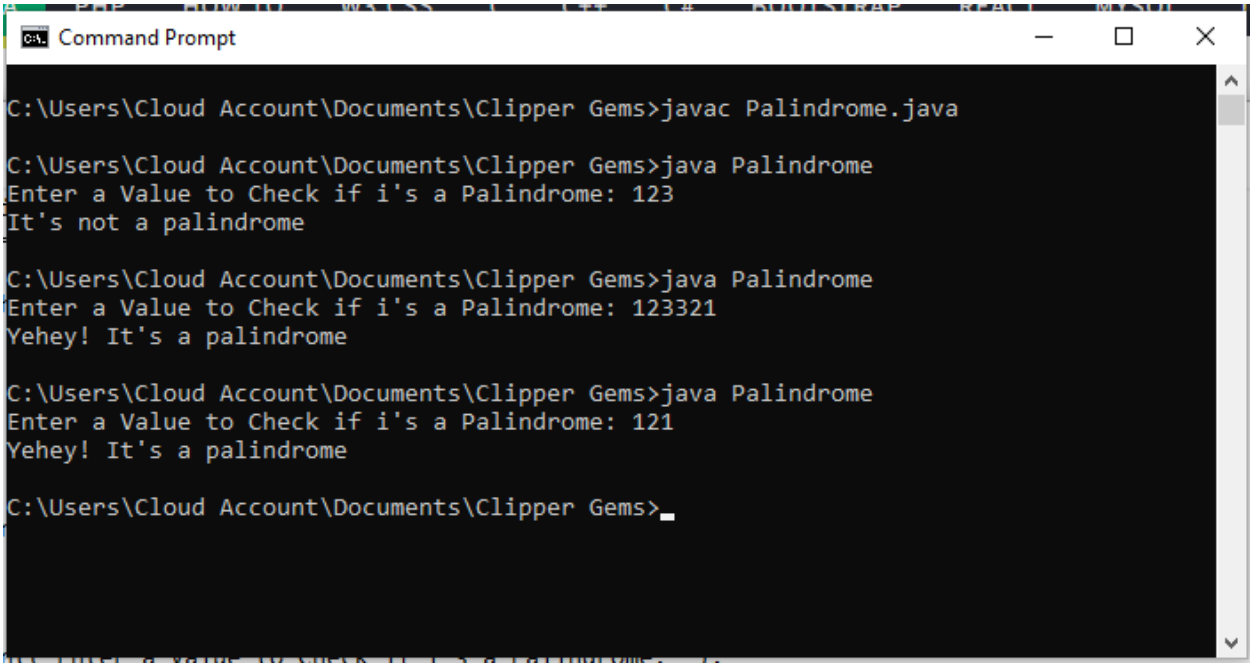
```
import java.util.Scanner;

public class GCF{
        static int gcf(int x, int y) {
           if (y==0) return x;
           return gcf(y,x%y);
        }

        public static void main(String[] args){
                Scanner scan = new Scanner(System.in);
                int x, y, gcf;

                System.out.println("Greatest common factor calculator");
                System.out.print("Enter Value 1: ");
                x = scan.nextInt();

                System.out.print("Enter Value 2: ");
                y = scan.nextInt();

                System.out.println("Greatest common factor of " + x + " and " + y + " is " + gcf(x, y));
        }
}
```



Activity: Palindrome

- Make a new Java program that asks the user to input a string.
- Create a function that checks if the string is a palindrome or not, this function must return a boolean value.
- The output should display either "It's not a palindrome" or "Yehey! It's a palindrome.

```
import java.util.Scanner;

public class Palindrome{
        static int checker(String nth) {
                int strLength = nth.length();
                for(int i = 0; i < strLength; i++){
                        if (nth.charAt(i) != nth.charAt((strLength - 1) - i)){
                                return 0;
                        }
                }
                return 1;
        }

        public static void main(String[] args){
                Scanner scan = new Scanner(System.in);
                String nth;

                System.out.print("Enter a Value to Check if i's a Palindrome: ");
                nth = scan.nextLine();

                switch(checker(nth)){
                        case 1: System.out.println("Yehey! It's a palindrome"); break;
                        case 0: System.out.println("It's not a palindrome"); break;
                        default: break;
                }
        }
}
```



Activity: FizzBuzz

- Write a Java program that prints the numbers from 1 to 100.
- But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz".
- For numbers which are multiples of both three and five print "FizzBuzz".

```
public class FizzBuzz{
        public static void main(String[] args){
                for(int i = 1; i <= 100; i++){
                        if (i % 3 == 0 && i % 5 == 0){
                                System.out.println("FizzBuzz");
                        } else if (i % 3 == 0){
                                System.out.println("Fizz");
                        } else if (i % 5 == 0){
                                System.out.println("Buzz");
                        } else {
                                System.out.println(i);
                        }
                }
        }
}
```

```
Command Prompt

C:\Users\Cloud Account\Documents\Clipper Gems>javac FizzBuzz.java

C:\Users\Cloud Account\Documents\Clipper Gems>java FizzBuzz
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
```

Activity: Control Structures

Type, compile and run the following java program using a file
Named "MyIfElseProgram.java". What does this program do?

- It utilizes the control structure to print specific prompts based on the if statement

```
C:\Users\Cloud Account\Documents\Clipper Gems>javac MyIfElseProgram.java

C:\Users\Cloud Account\Documents\Clipper Gems>java MyIfElseProgram
First one: 0
Less than 5: 1
Less than 5: 2
Less than 5: 3
Less than 5: 4
Not less than 5: 5
Not less than 5: 6
Not less than 5: 7
Not less than 5: 8
Not less than 5: 9

C:\Users\Cloud Account\Documents\Clipper Gems>
```

The program prints out all a prompt depending on which condition was met:

- First condition is checking if the n is Zero, then if true then it prints "First one: " and value of n
- Second condition is checking if n is less than 5, then if true then it prints "Less than 5: " and value of n
- The last statement is triggered if everything else is false, then it prints "Not less than 5: " and value of n

```java
// What's the difference?

int n = 10;
while(n < 10) {
    System.out.println("n is " + n);
    n++;
}

int n = 10;
do
{
    System.out.println("n is " + n);
    n++;
}
while(n < 10);
```

Activity: Control structures

- Enhance your previous program.
- Instead of counting from 0 to 9, let it count from 0 to 20.
- For numbers 10 and 11, the program should say: "Ten or Eleven: (number)

```java
public class MyIfElseProgram{
        public static void main(String[] args){
                int n;
                for(n = 0; n <= 20; n++){
                        if (n == 0){
                                System.out.println("First one: " + n);
                        } else if (n < 5){
                                System.out.println("Less than 5: " + n);
                        } else if (n == 10) {
                                System.out.println("Ten: " + n);
                        } else if (n == 11){
                                System.out.println("Eleven: " + n);
                        } else{
                                System.out.println("Not less than 5: " + n);
                        }
                }
        }
}
```

```
Command Prompt
Not less than 5: 7
Not less than 5: 8
Not less than 5: 9

C:\Users\Cloud Account\Documents\Clipper Gems>javac MyIfElseProgram.java

C:\Users\Cloud Account\Documents\Clipper Gems>java MyIfElseProgram
First one: 0
Less than 5: 1
Less than 5: 2
Less than 5: 3
Less than 5: 4
Not less than 5: 5
Not less than 5: 6
Not less than 5: 7
Not less than 5: 8
Not less than 5: 9
Ten: 10
Eleven: 11
Not less than 5: 12
Not less than 5: 13
Not less than 5: 14
Not less than 5: 15
Not less than 5: 16
Not less than 5: 17
Not less than 5: 18
Not less than 5: 19
Not less than 5: 20

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: Sample Program

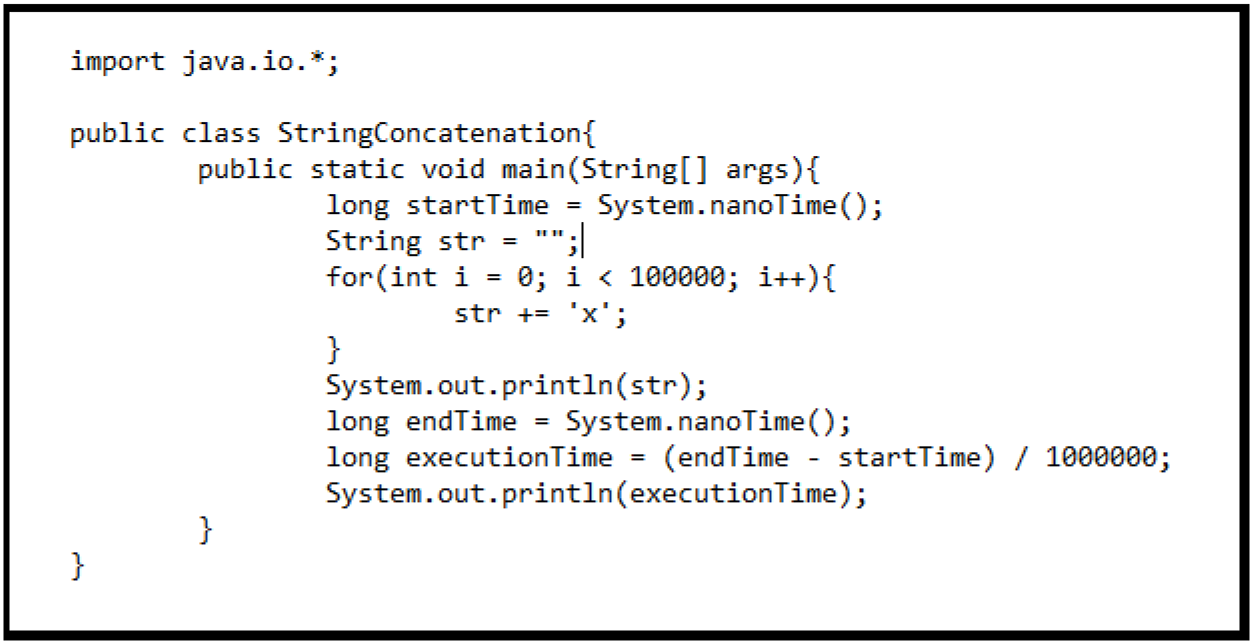
What is the output of this program? Why?

- The first print is “First, the value is 2000000000” because the value declared was 2billion
- The second print is “After adding, the value is 3000000000” because the initial value of was added to 1 billion, and the total was returned to n.
- The code could be shortened to n+=1000000000 as you’re adding the value directly to the n.

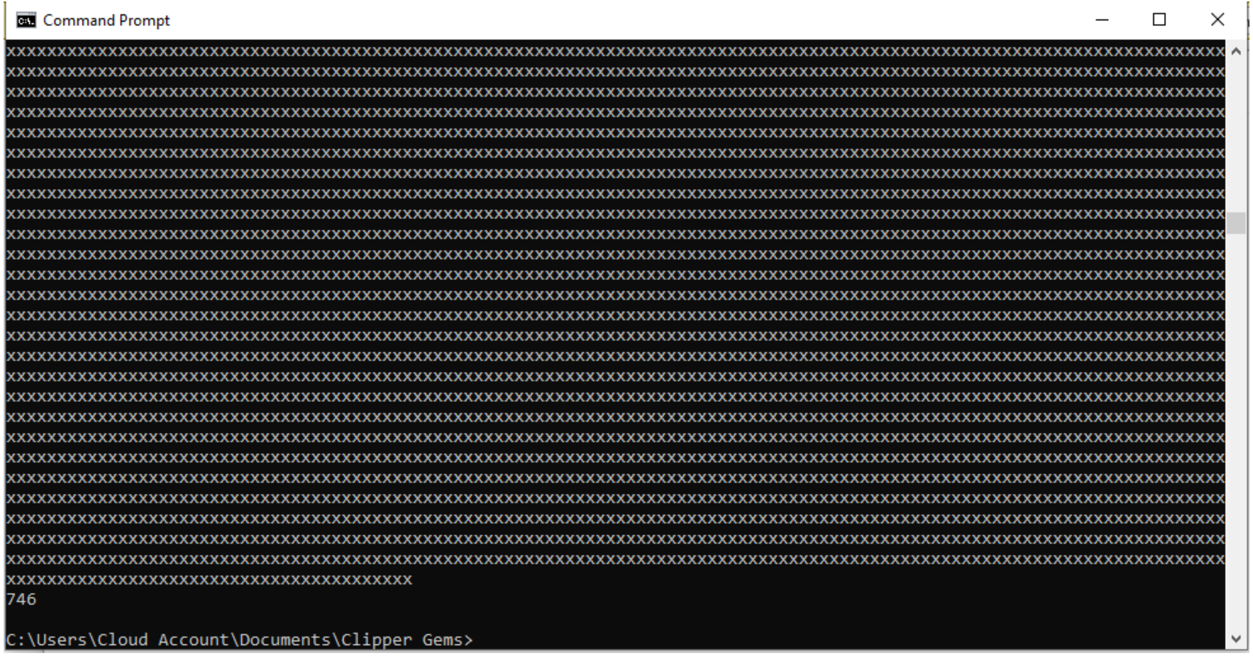
```java
// Type in this program and run it:

public class JavaIntegerOverflow
{
    public static void main(String[] args) {
        int n = 2000000000; // 2 billion: 9 zeroes
        System.out.println("First, the value is " + n);
        n = n + 1000000000; // 1 billion: 9 zeroes
        System.out.println("After adding, the value is " + n);
    }
}

// What is the output of this program? Why?
```

Activity: String Concatenation (1)

- Create a new Java class (name it any way you like), and create a main method like in our previous programs.
- In the main method, declare a single variable named "str" of type "String“.
- Let the initial value upon declaration be "" (empty string).
- Create a for loop that loops 100000 times.
- On every iteration of the loop, it adds the string "x" to the existing string "str" (ultimately creating a string with 100000 "x" characters).
- Compile and run the program.
- Measure how long it takes for the program to execute
    - 746ms



```java
import java.io.*;

public class StringConcatenation{
        public static void main(String[] args){
                long startTime = System.nanoTime();
                String str = "";
                for(int i = 0; i < 100000; i++){
                        str += 'x';
                }
                System.out.println(str);
                long endTime = System.nanoTime();
                long executionTime = (endTime - startTime) / 1000000;
                System.out.println(executionTime);
        }
}
```

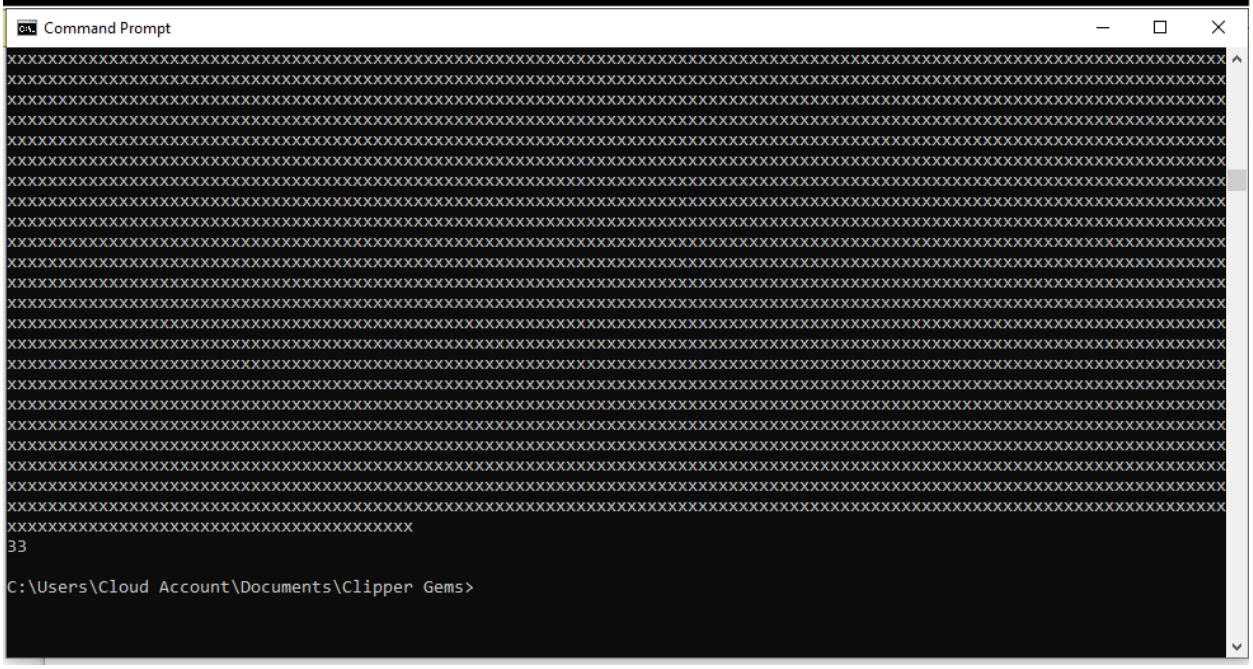


```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
746

C:\Users\Cloud Account\Documents\Clipper Gems>
```

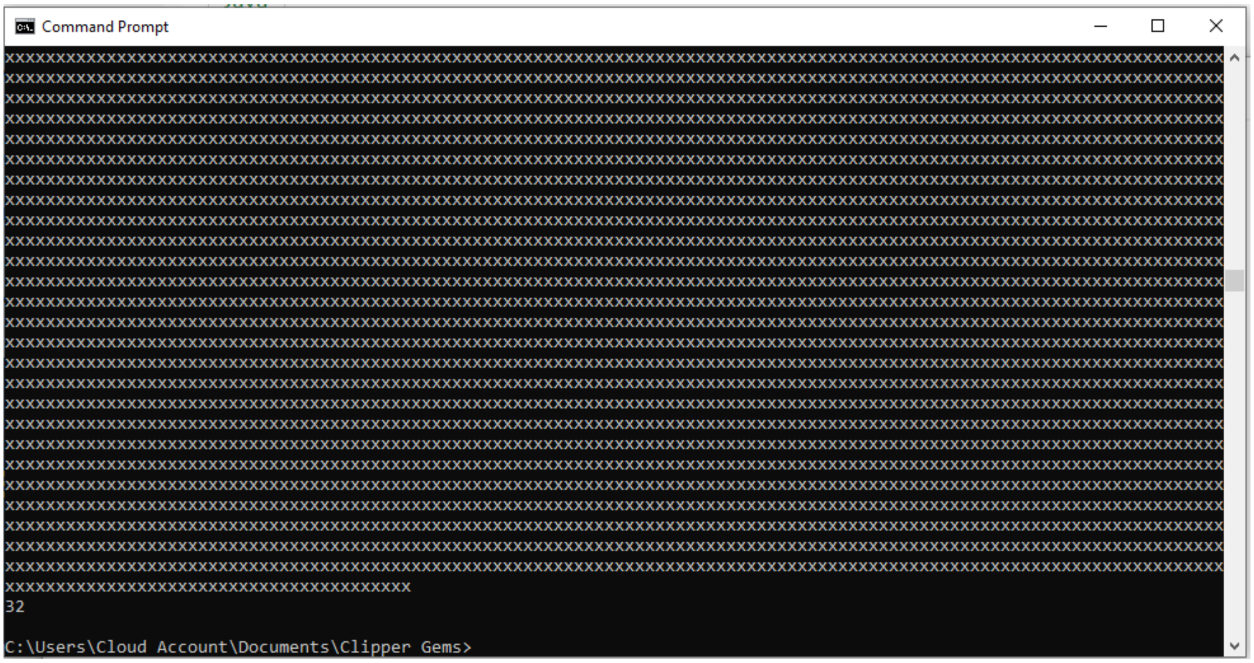
Activity: String Concatenation (2)

- Modify your previous program: Instead of concatenating String objects, use a StringBuilder object to construct the 100000 characters long string.
- How does this change affect the execution speed?
    - 33ms

```
import java.io.*;

public class StringConcatenation{
        public static void main(String[] args){
                long startTime = System.nanoTime();
                StringBuilder str = new StringBuilder();
                for(int i = 0; i < 100000; i++){
                        str.append("x");
                }
                System.out.println(str);
                long endTime = System.nanoTime();
                long executionTime = (endTime - startTime) / 1000000;
                System.out.println(executionTime);
        }
}
```



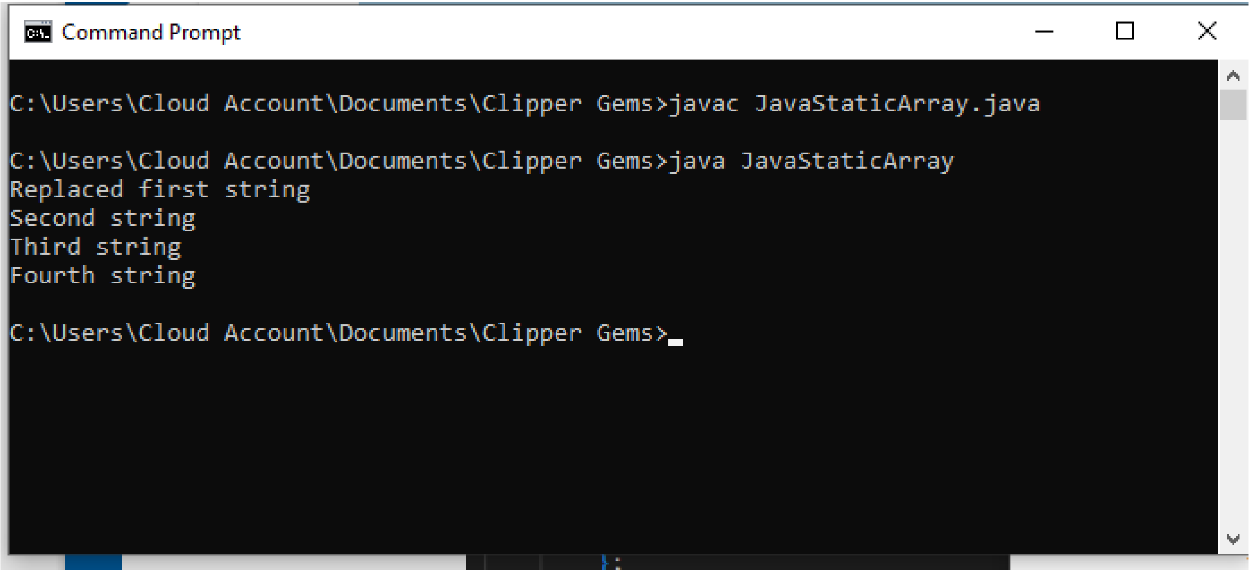
- How about if you use StringBuffer instead of StringBuilder?
    - 32ms (1ms difference)



```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
32

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: A static array example

The following program demonstrates the use of a Simple array to store strings. Try it.



```

C:\Users\Cloud Account\Documents\Clipper Gems>javac JavaStaticArray.java

C:\Users\Cloud Account\Documents\Clipper Gems>java JavaStaticArray
Replaced first string
Second string
Third string
Fourth string

C:\Users\Cloud Account\Documents\Clipper Gems>_
```

Activity: Static Array (1)

Change the previous program (JavaStaticArray) so that it will print the strings in reverse order (starting from the last, going up towards the first)

```
public class JavaStaticArray{
        public static void main(String[] args){
                String[] strings;
                strings = new String[]{
                        "First string",
                        "Second string",
                        "Third string",
                        "Fourth string",
                };
                strings[0] = "Replaced first string";
                for(int n=strings.length-1; n>=0; n--){
                        System.out.println(strings[n]);
                }
        }
}
```

Command Prompt

```
C:\Users\Cloud Account\Documents\Clipper Gems>javac JavaStaticArray.java

C:\Users\Cloud Account\Documents\Clipper Gems>java JavaStaticArray
Fourth string
Third string
Second string
Replaced first string

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: Static Array (2)

- Change the program so that the user must specify the order in which the strings are printed when running the program:
- If the program is executed as "java JavaStaticArray normal", the strings are printed in the order in which they were declared.
- If the program is executed as "java JavaStaticArray reverse", the strings are printed in reverse order.
- If neither "reverse" or "normal" are supplied, then the program should exit with an error message.

```
public class JavaStaticArray{
        public static void main(String[] args){
                String[] strings;
                strings = new String[]{
                        "First string",
                        "Second string",
                        "Third string",
                        "Fourth string",
                };
                strings[0] = "Replaced first string";
                if (args[0].equals("normal")){
                        for(int n=0; n<strings.length; n++){
                                System.out.println(strings[n]);
                        }
                } else if (args[0].equals("reverse")){
                        for(int n=strings.length-1; n>=0; n--){
                                System.out.println(strings[n]);
                        }
                }
        }
}
```

Command Prompt

```
C:\Users\Cloud Account\Documents\Clipper Gems>javac JavaStaticArray.java

C:\Users\Cloud Account\Documents\Clipper Gems>java JavaStaticArray normal
Replaced first string
Second string
Third string
Fourth string

C:\Users\Cloud Account\Documents\Clipper Gems>java JavaStaticArray reverse
Fourth string
Third string
Second string
Replaced first string

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: Dynamic Array example

The following program demonstrates the use of a
Dynamic array to store strings. What is the difference
To the static array? Try IT

- The difference in both is that Static is fixed size, meanwhile dynamic isn't.

```
import java.util.ArrayList;

public class JavaDynamicArray{
        public static void main(String[] args){
                ArrayList<String> dynamicArray = new ArrayList<String>();
                dynamicArray.add("First string");
                dynamicArray.add("Second string");
                dynamicArray.add("Third string");
                dynamicArray.add("Fouth string");
                for(int n = 0; n<dynamicArray.size(); n++){
                        System.out.println(dynamicArray.get(n));
                }
        }
}
```

Command Prompt

```
C:\Users\Cloud Account\Documents\Clipper Gems>javac JavaDynamicArray.java

C:\Users\Cloud Account\Documents\Clipper Gems>java JavaDynamicArray
First string
Second string
Third string
Fouth string

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: Multi-dimensional array

Try it, What does this program do?

- This program loops every array on 2d. And adding the value of c that also increase by 1 per loop.

```
public class MultiArraySample{
        public static void main(String[] args){
                int[][] array = new int[10][5];
                int c = 0;
                for(int n1 = 0; n1 < 10; n1++){
                        for(int n2 = 0; n2 < 5; n2++){
                                array[n1][n2] = c++;
                        }
                }
        }
}
```

Activity: Multi-dimensional array

Add to the previous program: Change it so that the program will also print the contents of the array in the following format

MultiArraySample.java - Notepad

File Edit Format View Help

```
ss MultiArraySample{
blic static void main(String[] args){
      int[][] array = new int[10][5];
      int c = 0;
      for(int n1 = 0; n1 < 10; n1++){
              for(int n2 = 0; n2 < 5; n2++){
                      array[n1][n2] = c++;
                      System.out.print(array[n1][n2] + ",");
              }
              System.out.println("");
      }
```

Ln 10, Col 22 | 100% | Windows (CRLF) | UTF-8

```
Command Prompt

C:\Users\Cloud Account\Documents\Clipper Gems>javac MultiArraySample.java

C:\Users\Cloud Account\Documents\Clipper Gems>java MultiArraySample
0,1,2,3,4,
5,6,7,8,9,
10,11,12,13,14,
15,16,17,18,19,
20,21,22,23,24,
25,26,27,28,29,
30,31,32,33,34,
35,36,37,38,39,
40,41,42,43,44,
45,46,47,48,49,

C:\Users\Cloud Account\Documents\Clipper Gems>
```

Activity: Simple Product Database

• Make a new Java program (class) named SimpleProductDatabase: In the main method, introduce a dynamic string array.

• In the array, insert several strings in the following format: ":" (eg: "Mitsubishi Adventure:800000", "Brewed Coffee (Venti):110", "Candy bar:50", etc.)

• Then let the program loop through the array using a for loop, printing out the product information for each record in the following format: "Product: , Price: ".

• (eg: "Product: Mitsubishi Adventure, Price: 800000", "Product: Brewed Coffee (Venti), Price: 110", etc.)

```java
import java.lang.*;
import java.util.ArrayList;

public class SimpleProductDatabase{
        static int getColon(String n){
                for(int i = 0; i < n.length(); i++){
                        String str = "";
                        str += n.charAt(i);
                        if (str.equals(":")){
                                return i;
                        }
                }
                return -1;
        }

        public static void main(String[] args){
                ArrayList<String> dynamicArray = new ArrayList<String>();
                dynamicArray.add("Mitsubishi Adventure:800000");
                dynamicArray.add("Brewed Coffee(Venti):110");
                dynamicArray.add("Candy bar:50");
                for(int i = 0; i <dynamicArray.size(); i++){
                        String str = dynamicArray.get(i);
                        int nth = getColon(str);
                        System.out.println("Product: " + str.substring(0, nth) + ", Price: " + str.substring(nth+1, str.length()));
                }
        }
}
```

Command Prompt

```
C:\Users\Cloud Account\Documents\Clipper Gems>javac SimpleProductDatabase.java

C:\Users\Cloud Account\Documents\Clipper Gems>java SimpleProductDatabase
Product: Mitsubishi Adventure, Price: 800000
Product: Brewed Coffee(Venti), Price: 110
Product: Candy bar, Price: 50

C:\Users\Cloud Account\Documents\Clipper Gems>
```