Assess Yourself

Write a program that takes a single line of input, and counts the number of lowercase, uppercase, and space characters, and outputs them on the same line, separated by spaces.

Hint: Google how to check character case.

Input:

```
"Never gonna give you up. Never gonna let you down."
```

Output: `"37 2 9"`

Assess Yourself

Assess Yourself

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        String text = scanner.nextLine();

        int nLower = 0;
        int nUpper = 0;
        int nSpaces = 0;

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);

            if (Character.isLowerCase(c)) {
                nLower++;
            } else if (Character.isUpperCase(c)) {
                nUpper++;
            } else if (c == ' ') {
                nSpaces++;
            }
        }

        System.out.format("%d %d %d", nLower, nUpper, nSpaces);
    }
}
```

BIOMEDICAL RESEARCH VICTORIA
INNOVATING FOR HEALTH

Now Recruiting ***Computer Science** Students

Undergraduate Research Opportunities Program (UROP)

- UROP provides **casual paid employment** for undergraduate students (2nd year +) in biomedical research teams in Melbourne
- Do you have **PROGRAMMING SKILLS** & an **interest in biomedical science**?
- Applications open: **6 – 20 March 2018** for projects starting in winter semester break

www.biomedvic.org.au/urop

Principal Sponsor

CSL™

SWEN20003
Object Oriented Software Development

Arrays

Semester 1, 2018

The Road So Far

- Java basics
- Basic classes
- Formatted output
- Dynamic, “repeat use” programs
  - User input
  - Branching
- Iteration

Lecture Objectives

After this lecture you will be able to:

- Declare and create arrays
- Combine iteration with arrays to store and manipulate large datasets
- Use multiple approaches to fill an array with data

In-class code found here

Arrays

Motivation

- Store a single integer value

```
int x;
```

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store `n` integer values

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store n integer values

```
Uhh...
```

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store `n` integer values

```
Uhh...int[] ints;
```

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store `n` integer values

```
Uhh...int[] ints;
```

Keyword

Array: A sequence of elements *of the same type* arranged in order in memory

Array Declaration

```
basetype[] varName; OR
basetype varName[];
```

- **Declares** an array (`[]`)
- Each *element* is of type `basetype`

Array Declaration

```
basetype[] varName; OR
basetype varName[];
```

- **Declares** an array (`[]`)
- Each *element* is of type `basetype`

```
int[] ints;
```

Array Declaration

```
basetype[] varName; OR
basetype varName[];
```

- **Declares** an array (`[]`)
- Each *element* is of type `basetype`

```
int[] ints;
```

How many elements does this array have?

Pitfall: Array Declaration

```java
int[] ints;
int x = ints[0];
```

Pitfall: Array Declaration

```
int[] ints;
int x = ints[0];
```

```
Program.java:13: error: variable ints might not have been initialized
```

- Arrays must be initialised, just like any other variable
- Let's look at how

Array Assignment

```
int[] ints = {0, 1, 2, 3, 4};
```

- How many elements?
- What are their values?

Array Assignment

```java
int[] ints = {0, 1, 2, 3, 4};
```

- How many elements?
- What are their values?

```java
int[] ints = new int[100];
String[] strings = new String[100];
```

- How many elements?
- What are their values?

Array Assignment

```java
int[] ints = {0, 1, 2, 3, 4};
```

- How many elements?
- What are their values?

```java
int[] ints = new int[100];
String[] strings = new String[100];
```

- How many elements?
- What are their values?

```java
int[] ints1 = new int[n];
int[] ints2 = ints1;
```

- How many elements?
- What are their values?

Assess Yourself

```
int[] ints1 = {10, 20, 30, 40};
int[] ints2 = ints1;

System.out.println(ints2[0]);

ints1[0] = 15;

System.out.println(ints2[0]);
```

Output: ?

Assess Yourself

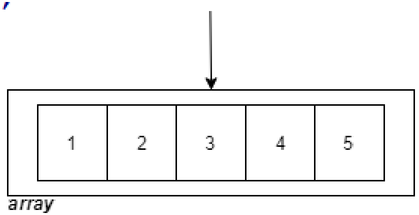
```
int[] ints1 = {10, 20, 30, 40};
int[] ints2 = ints1;

System.out.println(ints2[0]);

ints1[0] = 15;

System.out.println(ints2[0]);
```

Output:

```
10
15
```

Pitfall: Array Assignment

```java
int[] array = {
    1, 2, 3, 4, 5
};
```



- Arrays are *references*!
- Manipulating one reference affects all references

Assess Yourself

Write a program that accepts a single user input `n`, and creates an array of doubles of that size. Your program should then fill that array with increasing powers of two (starting from 1.0).

Assess Yourself

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();
        double[] nums = new double[n];

        for (int i = 0; i < n; i++) {
            nums[i] = Math.pow(2, i);
        }

        // For sanity checking
        for (int i = 0; i < n; i++) {
            System.out.println(nums[i]);
        }

    }
}
```

Multi-Dimensional Arrays

- Java permits “multi-dimensional” arrays
- Technically exist as “array of arrays”
- Declared just like 1D arrays

Multi-Dimensional Arrays

- Java permits “multi-dimensional” arrays
- Technically exist as “array of arrays”
- Declared just like 1D arrays

```java
int[][] nums = new int[10][10]; // Square array
int[][] nums = new int[10][];   // Irregular array
```

- Initialising irregular arrays slightly more complicated

Multi-Dimensional Arrays

- Java permits “multi-dimensional” arrays
- Technically exist as “array of arrays”
- Declared just like 1D arrays

```
int[][] nums = new int[10][10]; // Square array
int[][] nums = new int[10][];   // Irregular array
```

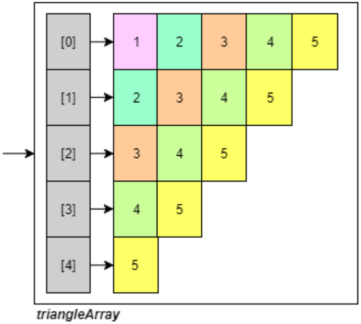
- Initialising irregular arrays slightly more complicated

```
for (int i = 0; i < nums.length; i++) {
    nums[i] = new int[i + 1];
}
```

Assess Yourself

Write a program that can generate the following 2D array:

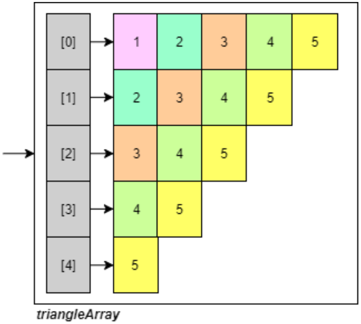
```
int[][] triangleArray = {
        {1, 2, 3, 4, 5},
        {2, 3, 4, 5},
        {3, 4, 5},
        {4, 5},
        {5},
};
```



Assess Yourself

Write a program that can generate the following 2D array:

```
int[][] triangleArray = {
        {1, 2, 3, 4, 5},
        {2, 3, 4, 5},
        {3, 4, 5},
        {4, 5},
        {5},
};
```



Can you write your program with as **few assumptions as possible**?

Assess Yourself

```
public class Main {
    public static void main(String[] args) {
        int HEIGHT = 5;
        int MAX_WIDTH = HEIGHT;

        int[][] triangleArray= new int[HEIGHT][];

        for (int i = 0; i < HEIGHT; i++) {
            triangleArray[i] = new int[HEIGHT - i];

            for (int j = 0; j < HEIGHT - i; j++) {
                triangleArray[i][j] = j + 1;
            }
        }
    }
}
```

Array Methods

- Indexing

```
int x = ints[0];
int x = ints[-1]; // Gives out of bounds error
```

Array Methods

- Indexing

```
int x = ints[0];
int x = ints[-1]; // Gives out of bounds error
```

- Length

```
int len = ints.length
```

Array Methods

- Indexing

```
int x = ints[0];
int x = ints[-1]; // Gives out of bounds error
```

- Length

```
int len = ints.length
```

- Equality

```
import java.util.Arrays;

int[] n1 = {1, 2, 3};
int[] n2 = {1, 2, 3};

Arrays.equals(n1, n2));
```

Array Methods

- Resizing

```
ints = new int[ints.length + 1]
```

Array Methods

- Resizing

```
ints = new int[ints.length + 1]
```

- Sorting (“ascending”)

```
Arrays.sort(n1);
```

Array Methods

- Resizing

```
ints = new int[ints.length + 1]
```

- Sorting (“ascending”)

```
Arrays.sort(n1);
```

- Printing

```
System.out.println(Arrays.toString(n1));
"[1, 2, 3]"
```

- Full Array documentation here

For Each Loop

```
for (<type> varName : <iterable object>) {
    <block of code to execute>
}
```

- More convenient method of iteration
- No indexing required
- Useful when operating with/on the *data*, and not the array

For Each Loop

```
for (<type> varName : <iterable object>) {
    <block of code to execute>
}
```

- More convenient method of iteration
- No indexing required
- Useful when operating with/on the *data*, and not the array

```
for (int i : ints) {
    System.out.println(i);
}
```

Assess Yourself

Write a program that asks the user for a single input `n`, and then asks for `n` more `String` inputs (words, lines, etc.). Each input should be added to an array of size `n`.

Once the array has been filled, your program should then print the array, **sort** the array, and then print the array once more.

Assess Yourself

```
Input n: 5
Input some text: Hello World
Input some text: this is some text
Input some text: SWEN20003
Input some text: Object Oriented Software Development
Input some text: More text
[Hello World, this is some text, SWEN20003,
        Object Oriented Software Development, More text]
[Hello World, More text, Object Oriented Software
        Development, SWEN20003, this is some text]
```

Assess Yourself

```java
import java.util.Scanner;
import java.util.Arrays;

public class Program {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Input n: ");
        int n = scanner.nextInt();

        // Make sure we eat the newline character
        scanner.nextLine();

        String[] text = new String[n];

        for (int count = 0; count < n; count++) {
            System.out.print("Input some text: ");
            text[count] = scanner.nextLine();
        }

        System.out.println(Arrays.toString(text));
        Arrays.sort(text);
        System.out.println(Arrays.toString(text));

    }
}
```

Metrics

Write a program that continually accepts a line of text from the user, and stores the frequency of the *length* of the words entered across all lines. For simplicity, you may assume that the maximum length of any word in the input is 10; your program may ignore longer words.

The output of your program should be a list in the following format:

```
"Length 01 words: 3"
"Length 02 words: 6"
...
"Length 10 words: 0"
```

Bonus: Customise your program so that the maximum word length is also provided by the user, before any text is read.