# Assess Yourself

Write a program that accepts three inputs from the user:

- `exchangeFile`, the name of a file containing exchange rates **from** AUD
- `audValues`, the name of a file containing a number of values in AUD
- `currency`, the currency to convert **to**, from AUD

Your program must read the contents of `exchangeFile` to find the value of `currency`, then process all the items in `audValues` and output their converted value.

The contents of `exchangeFile` will be in csv format, with two columns:
`Currency acronym (AUD, USD),Exchange rate to AUD`

The `audValues` file will have one AUD value per line.

Example:

```
exchange.txt
values.txt
USD
34.00AUD = 26.52USD
29.43AUD = 22.96USD
568.30AUD = 443.27USD
```

# Assess Yourself

```
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

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

        Scanner scanner = new Scanner(System.in);

        String exchangeFile = scanner.nextLine();
        String audValues = scanner.nextLine();
        String currency = scanner.nextLine();

        try (Scanner exchange = new Scanner(new FileReader(exchangeFile));
            Scanner values = new Scanner(new FileReader(audValues))) {

            ...

        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

# Assess Yourself

```
double exchangeRate = 0.0;

while (exchange.hasNextLine()) {
    String[] line = exchange.nextLine().split(",");

    if (line[0].equals(currency)) {
        exchangeRate = Double.parseDouble(line[1]);
        break;
    }
}

while (values.hasNextLine()) {
    double value = Double.parseDouble(values.nextLine());

    System.out.format("%.2fAUD = %.2f%s\n",
                value, value*exchangeRate, currency);
}
```

# Assess Yourself

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

**Hint:** Google how to check character case/category.

Input:

```
"Winter is coming. I am Iron Man!"
```

Output: `"20 4 7"`

# 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 nPunctuation = 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 (Character.getType(c) == Character.OTHER_PUNCTUATION) {
                nPunctuation++;
            }
        }

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

SWEN20003
Object Oriented Software Development

# Arrays

Semester 1, 2019

# The Road So Far

- OOP Foundations
  - Classes and Objects
  - Strings and Wrappers
  - Formatting
  - Methods and Abstraction
  - Input and Output

# Lecture Objectives

After this lecture you will be able to:

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

In-class code found here

# Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store `n` integer values

```
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`

```
int[] ints;
```

How many elements does this array have?

# 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};
Superhero[] heroes = {new Superhero("Tony Stark", "Iron Man")};
```

- How many elements?
- What are their values?

```
int[] ints = new int[100];
Superhero[] heroes = new Superhero[100];
```

- How many elements?
- What are their values?

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

- How many elements?
- What are their values?

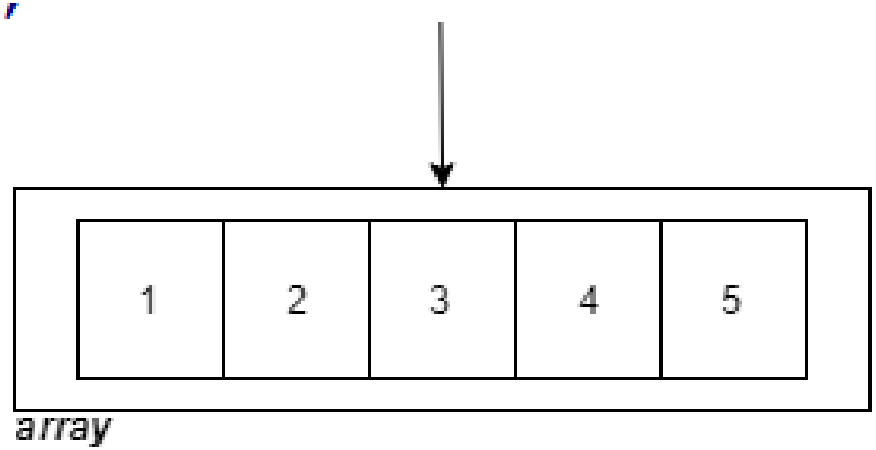
# Assess Yourself

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

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

ints1[0] = 15;

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

Output:
10
15

# Pitfall: Array Assignment

```
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 `double`s 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

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

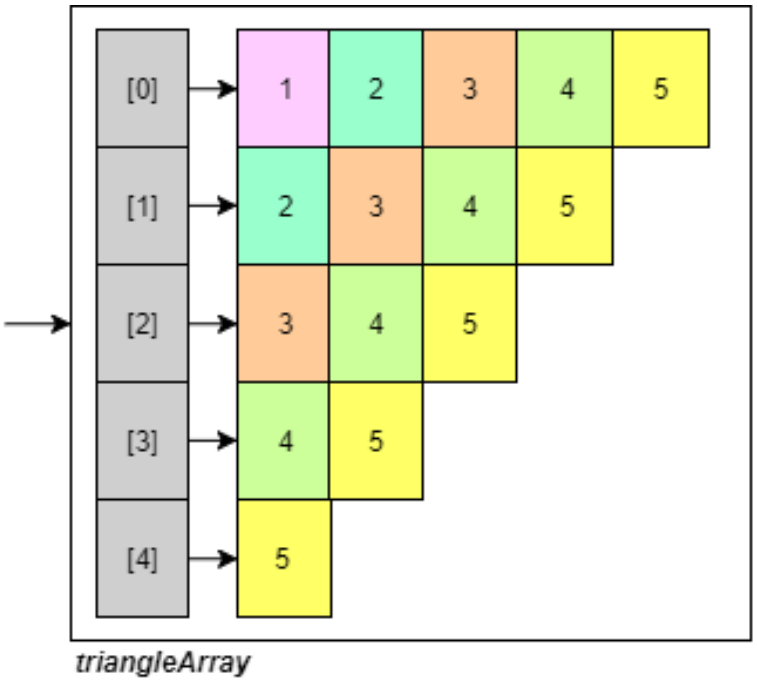
- Initialising irregular arrays slightly more complicated

```
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},
};
```




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] = i + j + 1;
            }
        }
    }
}
```

# 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]
```

- 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 (Person p : people) {
    System.out.println(p.isHungry());
}
```

# Assess Yourself

Write a program that asks the user for a single input `n`, and then asks for $2 \times n$ more `String` inputs, representing a superhero's public and secret identities. Each pair of inputs should be then be used to add a `Superhero` object to an array of size `n`.

Once the array has been filled, your program should then print the array.

# Assess Yourself

```
Input a number: 2
Input the identities of a hero: Tony Stark,Iron Man
Input the identities of a hero: Natasha Romanoff,Black Widow
[Tony Stark is really Iron Man!, Natasha Romanoff is really Black Widow!]
```

# 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);

        //Ask for number Input
        System.out.print("Input a number: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        //Intialise array of n length
        Superhero heroes[] = new Superhero[n];

        //Ask for 2*n string inputs
        //Create new objects and put values into array
        for (int i=0; i<n; i++){
            System.out.print("Input the identities of a hero: ");
            String[] identities = scanner.nextLine().split(",");
            heroes[i] = new Superhero(identities[0], identities[1]);
        }

        //Print array
        System.out.print(Arrays.toString(heroes));

    }
}
```

# 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.

# End of Foundation Block!