COMP90041
Programming and Software Development
2020 - Semester 2
Lab 6

Yuhao(Beacon) Song

yuhsong1@unimelb.edu.au

yuhsong@student.unimelb.edu.au

Introduction

- **Timetable**
  - Tue(18) 16:15-17:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/94854648719?pwd=WUY0NmR6MkI5UVZBUWhGNWFIU216Zz09
  - Wed(19) 17:15-18:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/93723434766?pwd=MGh4QkJuZnhJR21LZ0VqeXhJQU52UT09

- **GitHub Page (Tutorial Materials, Solutions, Additional Resources)**
  - https://github.com/Beaconsyh08/COMP90041-2020SEM2

- **PollEv**
  - https://pollev.com/yuhsong

Outline

- Assignment Updates
- Lecture Review
- Exercise & demo

Assignment Update - Authorship

4. **TASK: Make sure to add a file entitled '*authorship.txt*' to your repository and submit it along with your code. This applies to everyone, regardless whether you think you have correctly linked your accounts.** The *authorship.txt file* must contain the following lines:

```
Your full name:
Your Canvas username (i.e., yourname in
yourname@student.unimelb.edu.au):
Github repository link:
```

The Easiest Way

- Copy the required "Authorship.txt" file and add to your own **repository**, then **commit** & **push**

- Original one: https://github.com/COMP90041/sem2-2020-assignment-01

```
Make sure to fill in the following information regarding authorship:

Your full name:
Your username (from Canvas, e.g., yourname@student.unimelb.edu.au):
Github repository link:

By submitting work for assessment I hereby declare that I understand the University's policy on academic integrity and that the work submitted is original and solely my work, and that I have not been assisted by any other person (collusion) apart from where the submitted work is for a designated collaborative task, in which case the individual contributions are indicated. I also declare that I have not used any sources without proper acknowledgment (plagiarism). Where the submitted work is a computer program or code, I further declare that any copied code is declared in comments identifying the source at the start of the program or in a header file, that comments inline identify the start and end of the copied code, and that any modifications to code sources elsewhere are commented upon as to the nature of the modification.
```

Pitfall: Array Index Out of Bounds

- Array **indices** always start with 0, and always end with the integer that **is one less than the size of the array(arr.length - 1)**
  - The most common programming error made when using arrays is attempting to use a **nonexistent array index**

- When an index expression evaluates to some value other than those allowed by the array declaration, the index is said to **be out of bounds**
  - An out of bounds index will cause a program to terminate with **a run-time error** message
  - Array indices get out of bounds most commonly at the first or last iteration of a loop that processes the array: Be sure to test for this!

The "for each" Loop

- The general syntax for a **for-each loop** statement used with an array is

  for (ArrayBaseType VariableName : ArrayName)

  Statement

- The above for-each line should be read as "for each VariableName in ArrayName do the following:"
  - Note that **VariableName** must be declared within the foreach loop, not before
  - Note also that a **colon** (not a semicolon) is used after VariableName

Arrays with a Class Base Type

- The **base type** of an array can be a **class type**

```
Date[] holidayList = new Date[20];
// holidayList[0] = new Date(xxx);
```

- The above example creates 20 **indexed variables** of **type Date**
  - It does NOT create 20 objects of the class Date
  - Each of these indexed variables are automatically initialized to **null - placeholder**
  - Any attempt to reference any them at this point would result in a "**null pointer exception**" error message
- Like any other object, each of the indexed variables requires a separate invocation of a constructor using new (singly, or perhaps using a for loop) to create an object to reference

```
for (int i = 0; i < holidayList.length; i++)
    holidayList[i] = new Date();
```

Use of “==” with Arrays

- For the same reason, the **equality operator (==)** only tests two arrays to see if they are stored in the same location in the computer's **memory**
  - It does not test two arrays to see if they contain the same values (a == b)
  - The result of the above boolean expression will be true if a and b share the same **memory address** (and, therefore, reference the same array), and false otherwise

- Use Arrays.equals to test if they are stored same value

```
Arrays.equals(arr1, arr2)
```

Exercise

```
// Read in the array from keyboard
readArray(numbers);

// Display an array
display(numbers);

// Get maximum value of an array
int max = getMax(numbers);
System.out.println("Max value is: " + max);

// Get the sum of all elements in an array
int sum = getSum(numbers);
System.out.println("Sum is: " + sum);

// Sort array elements in descending order
sortArrayDescendingly(numbers);

// Display an array
display(numbers);

// Find the element with the largest number of appearances
// If there is a tie then return the smaller element
int mostFrequent = getMostFrequent(numbers);
System.out.println("Most frequent value is: " + mostFrequent);
```