MACQUARIE University

Department of Computing

ITEC625 Fundamentals of Computer Science
Workshop - Arrays

Learning outcomes

This weeks workshop is about understanding how to create an use arrays.

1. **Creating arrays**

   Create arrays that can hold values for,

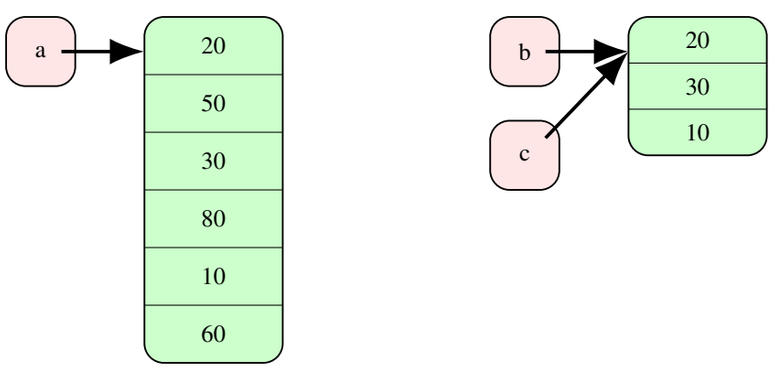
   - number of cars sold be each of the 8 employees.
   - GPAs of 50 students.
   - number of days it rained in each month.
   - the state of each of the 100 switches in the house (whether it's on or off).

```
int[] carSales = new int[8];
double[] gpas = new double[50];
int[] daysRained = new int[12];
boolean[] switchStates = new boolean[100];
```

2. **Memory representation**

   Based on the convention used in the lecture, draw a memory diagram for the following code:

```
int[] a = {20, 50, 30, 80, 10, 60};
int[] b = {a[0], a[2], a[4]};
int[] c = b;
```



3. **Tracing code involving arrays - 1** What is the state of the array `arr` when the following code is executed?

```
int[] arr = new int[8];
for(int i=0; i < arr.length/2; i++) {
        arr[i] = 2*i+1;
```

```
}
for(int i=arr.length/2; i < arr.length; i++) {
        arr[i] = 2*(arr.length - i) - 1;
}
```

`arr = {1,3,5,7,7,5,3,1}`

4. **Tracing code involving arrays - 2** What is the state of the array `arr` at the end of each iteration of the outer loop in the following code?

```
int[] arr = {12, 30, 8, 2, 7, 15};
for(int i=0; i < arr.length; i++) {
        int dodo = i;
        for(int k=i+1; k < arr.length; k++) {
                if(arr[k] < arr[dodo]) {
                        dodo = k;
                }
        }
        int temp = arr[i];
        arr[i] = arr[dodo];
        arr[dodo] = temp;
        //arr = ???
}
```

The code sorts the array.

`arr = {2, 7, 8, 12, 15, 30}`

5. **Basic operations on an array** Assume that there is an array `data` that has at least one item. Write a piece of code that stores the number of negative integers in the array.

```
int countNegs = 0;
for(int i=0; i < data.length; i++) {
        if(data[i] < 0) {
                countNegs++;
        }
}
```

6. **Methods operating on arrays** Complete the following method's definition based on the javadoc.

```
public class ArrayService {
        /**
        * @param arr
        * @param min
        * @param max: assume max >= min
        * @return true if there is any value in array arr that lies
        * in the range [min, max], false otherwise
        * for example,
        * if arr = {10, 70, 20, 90}, min = 30, max = 70, return true
        * if arr = {10, 70, 20, 90}, min = 30, max = 60, return false
        */
        public static boolean contains(int[] arr, int min, int max) {
                return false; //to be completed
        }
}
```

Test you code using the following JUnit test case:

```
@Test
public void testContains() {
        int[] a = {20, -50, 30, -40, 90};
        assertTrue(ArrayService.contains(a, 10, 20));
        assertTrue(ArrayService.contains(a, 30, 50));
        assertTrue(ArrayService.contains(a, -60, -20));
        assertTrue(ArrayService.contains(a, -50, -45));
        assertTrue(ArrayService.contains(a, -45, -40));

        assertFalse(ArrayService.contains(a, -49, -41));
        assertFalse(ArrayService.contains(a, 31, 89));
        assertFalse(ArrayService.contains(a, -39, 19));
}
```

```
public static boolean contains(int[] arr, int min, int max) {
        if(arr == null)
                return false;
        for(int i=0; i < arr.length; i++) {
                if(arr[i] >= min && arr[i] <= max) {
                        return true;
                }
        }
        return false;
}
```

7. **Method operating on arrays - 2** Complete the following method's definition based on the javadoc.

```
public class ArrayService {
        /**
        * @param a
        * @param b
        * @return true if a and b are identical, false otherwise
        * for example,
        * if a = {10, 70, 20}, b = {10, 70, 20}, return true
        * if a = {10, 70, 20}, b = {10, 70, 20, 90}, return false
        * if a = {10, 70, 20, 90}, b = {10, 70, 20}, return false
        * if a = {10, 70, 20}, b = {10, 70, 30}, return false
        * if a = {10, 70, 20}, b = {30, 70, 20}, return false
        */
        public static boolean areIdentical(int[] a, int[] b) {
                return false; //to be completed
        }
}
```

Test you code using the following JUnit test case:

```
@Test
public void testAreIdentical() {
        int[] a = {20, -50, 30};
        int[] b = {20, -50, 30};
        assertTrue(ArrayService.areIdentical(a, b));

        int[] c = {20, -50, 30};
        int[] d = {20, -50, 30, 60};
        assertFalse(ArrayService.areIdentical(c, d));
```

```
        int[] e = {20, -50, 30, 60};
        int[] f = {20, -50, 30};
        assertFalse(ArrayService.areIdentical(e, f));

        int[] g = {20, -50, 30};
        int[] h = {20, -50, 40};
        assertFalse(ArrayService.areIdentical(g, h));

        int[] i = {20, -50, 30};
        int[] j = {10, -50, 30};
        assertFalse(ArrayService.areIdentical(i, j));
}
```

```
public static boolean areIdentical(int[] a, int[] b) {
        if(a==null || b==null)
                return false;
        if(a.length != b.length)
                return false;
        for(int i=0; i < a.length; i++) {
                if(a[i] != b[i]) {
                        return false;
                }
        }
        return true;
}
```

8. **Methods returning arrays**

   Define a method `getPositiveItems` that when passed an integer array, returns an array containing the positive items from that array. For example,

   a. if the array passed = `{-6, 0, 5, -2, -9, 1, 0, 0, 3}`, return the array `{5, 1, 3}`.

   b. if the array passed = `{-6, 0, -2, -9, 0, 0}`, return the array `{}`.

```
public static int[] getPositiveItems(int[] a) {
        if(a==null)
                return null;
        int count = 0;
        for(int i=0; i < a.length; i++) {
                if(a[i] > 0) {
                        count++;
                }
        }
        int[] result = new int[count];
        int destIndex = 0;
        for(int i=0; i < a.length; i++) {
                if(a[i] > 0) {
                        result[destIndex] = a[i];
                        destIndex++;
                }
        }
        return result;
```

```
}
```

9. **(Advanced)**

   - Define a method that when passed two arrays (assume that all items within an array are different), returns an array containing items that exist in both arrays, in the order they exist in the first of the two arrays.
   - Define a method that when passed an array, returns an array holding the longest streak of items that are in ascending order.

```
public static int[] intersection(int[] a, int[] b) {
        int count = 0;
        for(int i=0; i < a.length; i++) {
                int originalCount = count;
                for(int k=0; k < b.length && count == originalCount; k++) {
                        if(a[i] == b[k]) {
                                count++;
                        }
                }
        }
        int[] result = new int[count];
        int destIndex = 0;
        for(int i=0; i < a.length; i++) {
                int before = destIndex;
                for(int k=0; k < b.length && destIndex == before; k++) {
                        if(a[i] == b[k]) {
                                result[destIndex] = a[i];
                                destIndex++;
                        }
                }
        }
        return result;
}

public static int[] getLongestAscendingStreak(int[] arr) {
        int[] lengths = new int[arr.length];
        for(int i=0; i < arr.length; i++) {
                lengths[i] = 1;
                boolean ascending = true;
                for(int k=i+1; k < arr.length && ascending; k++) {
                        if(arr[k] >= arr[k-1]) {
                                lengths[i]++;
                        }
                        else {
                                ascending = false;
                        }
                }
        }
        int max = 0;
        for(int i=1; i < lengths.length; i++) {
                if(lengths[i] > lengths[max]) {
                        max = i;
                }
        }
        int[] result = new int[lengths[max]];
        for(int i=0; i < result.length; i++) {
                result[i] = arr[max+i];
        }
        return result;
}
```