Tutorial 11

Arrays:

Exercise 1:

A. Write a code snippet to define the following arrays:

1. An int array named nums of size 10.
2. A double array named dobs of size 5.
3. A string array named names of size 100.

B. Which of the following array definitions is right and which is wrong?

1. `int i = new int(30);`
2. `double d[] = new double[30];`
3. `char[] r = new char(1..30);`
4. `int i[] = (3, 4, 3, 2);`
5. `float f[] = {2.3, 4.5, 6.6};`
6. `char[] c = new char();`

C. Given the following array definition,
`double[] arr = new double[5];`
Write code snippets (if possible) to answer the following questions:

1. Access the 1st element.
2. Access the element at index 0.
3. Access the last element.
4. Access the element at index 4.
5. Access the element before the last.
6. Access the element at index 6.
7. Given an integer variable $i < 5$, access the element at index $i – 1$
8. Assign the sum of the first two elements to the 4th element.
9. Given an integer variable i, assign the result of dividing the ith element by the element before it to last element. Your code must have full checks to avoid runtime errors.

D. Write a code snippet that uses an array initializer to create an array of characters r that contains the characters of the word “Riyadh”.

E. What is wrong with the following code? Is it a compile-time error or a runtime error?
```
int[] a = new int[-1];
```

F. What is wrong with the following code? Is it a compile-time error or a runtime error?
```
int[] a;
a[2] = 10;
```

G. Given a non-empty array a of integers and a Scanner object s, what is wrong with the following code? Does it have a compile-time error or runtime error?
```
for (int i = 0; i <= a.length; i++) {
  a[i] = s.nextInt();
}
```

H. Write a code snippet to create a boolean array b of size N, where N is entered by the user (assume user will enter a positive integer greater than zero). Then fill out the array such that elements with even index get true and elements with odd index get false (Element with index zero gets true).

I. Write a code snippet that shifts the elements of an array myList of size N where N > 0, one element to the left.

Exercise 2:

Design and implement a class ArrayOps that provides the following services:

A. `findMax` takes an array of int as parameter and returns the maximum value of its components
B. `findMin` takes an array of int as parameter and returns the minimum value of its components
C. `displayAllMax` takes an array of int as parameter and prints the index(es) of its maximum values
D. `displayAllMin` takes an array of int as parameter and prints the index(es) of its minimum values
E. `calcAvg` takes an array of int as parameter and returns the average value of its components
F. `contains` takes an array of int and an int key as parameters and returns true if key exists in the array, otherwise returns false
G. `find` takes an array of int and an int key as parameters and returns the index of key if it exists, otherwise returns -1
H. `displayAll` takes an array of int and an int key as parameters and displays all the indexes of key
I. modify the previious method to print "Not found" if the key does not exist.
J. `findAll` takes an array of int and an int key as parameters and returns all the indexes of key (How ?)
K. `hasDuplicates` takes an array of int and returns true if the array has duplicates, and false otherwise

Tutorial 09 Solutions

Exercise 1:

A.
```
1. int[] nums = new int[10];
2. double[] dobs = new double[5];
3. String[] names = new String[100];
```
B. `Correct: 2`

C.
```
1. arr[0]
2. arr[0]
3. arr[arr.length - 1]
4. arr[4]
5. arr[arr.length - 2]
6. can not access element at index 6
7. arr[i - 1]
8. arr[3] = arr[0] + arr[1]
9. if (i >= 2 && i <= arr.length && arr[i - 2] != 0)
     arr[arr.length -1] = arr[i - 1] / arr[i - 2];
```
D.
```
char[] r = {'R', 'i', 'y', 'a', 'd', 'h};
```
E. Can not create an array with a negative size. It will give you the following runtime error (exception): NegativeArraySizeException

F. Can not use an array without initializing it. This is a compile time error: "local variable may not have been initialized"

G. The loop at the last iteration will try to access the element at index a.length, which is outside the array. This will cause the following runtime error (exception): ArrayIndexOutOfBoundsException

H. Since boolean arrays are initialized by Java to false, we just need to fill out elements with even index with value true
```
boolean[] b = new boolean[N];
for (int i = 0; i < b.length; i++){
  if (i % 2 == 0)
  b[i] = true;
}
```
I.
```
int temp = myList[0];
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}
myList[myList.length - 1] = temp;
```

Exercise 2:

```
class ArrayOps {
```

A.

```
public int findMax(int[] x) {
  int result = Integer.MIN_VALUE;
      // or  = x[0];
  for (int i=0; i < x.length; i++)
    if (result < x[i])
      result = x[i];
  return result;
}
```

B.

```
public int findMin(int[] x) {
  int result = Integer.MAX_VALUE;
      // or  = x[0];
  for (int i=0; i < x.length; i++)
    if (result > x[i])
      result = x[i];
  return result;
}
```

C.

```
public void displayAllMax(int[] x) {
  int max = findMax(x);
  for (int i=0; i < x.length; i++)
    if (max == x[i])
      System.out.println(i);
}
```

D.

```
public void displayAllMin(int[] x) {
  int min = findMin(x);
  for (int i=0; i < x.length; i++)
    if (min == x[i])
      System.out.println(i);
}
```

E.

```
public double calcAvg(int[] x) {
  double sum = 0.0;
  for (int i=0; i < x.length; i++)
    sum += x[i];
  return sum / x.length;
}
```

F.

```
public boolean contains(int[] a, int k) {
  boolean result = false;
  for (int i=0; i < x.length; i++)
    if (a[i] == k)
      result = true;
  return result ;
}
```

G.
```
public int find(int[] x, int key) {
  for (int i=0; i < x.length; i++)
    if (x[i] == key)
      return i;
  return -1;
}
```

H.
```
public void displayAll(int[] x, int k) {
  for (int i=0; i < x.length; i++)
    if (x[i] == key)
      System.out.println(i);
}
```

I.
```
/* Modified version
public void displayAll(int[] x, int k) {
  boolean found = false;
  for (int i=0; i < x.length; i++)
    if (x[i] == key) {
      System.out.println(i);
      found = true
    }
   if (!found)
    System.out.println("Not found");
} */
```

J.
```
public int[] findAll(int[] x, int key ) {
  int count = 0;
  for (int i=0; i < x.length; i++)
    if (x[i] == key)
      count++;
  if (count < 0)
    return null;
  int[] result = new int[count];
  int j = 0;
  for (int i=0; i < a.length; i++)
    if (x[i] == key)
      result[j++] = x[i];
  return result;
}
```

K.
```
public boolean hasDublicate(int[] x){
  for (int i=0; i < x.length-1; i++)
    for (int j=i+1, j<x.length, j++)
      if (x[i] == x[j])
        return true;
  return false;
 }
}
```