Module 5 Quiz

LATEST SUBMISSION GRADE

100%

1. Of the following, which is the correct syntax to declare an array of 24 **boolean** elements named **arr**?   **1 / 1 point**

   ○ boolean arr[] = new boolean;

   ◉ boolean[] arr = new boolean[24];

   ○ boolean
     arr = new boolean[24];

   ○ boolean arr[24];

   ○ boolean[24] arr = new boolean[24];

2. Given the declaration of the 24 element **boolean** array **arr** above, what is the index of the last element of the array?   **1 / 1 point**

   ○ 25

   ◉ 23

   ○ 24

   ○ 22

   ✓ **Correct**
   Java arrays use zero-based
   indexing, where the index of the first array element is zero.

3. If you wanted to initialize the **boolean** array **arr** so that all elements contain the value **true**, you could use the following: **1 / 1 point**
   arr.fill(true);

   ◉ FALSE

   ○ TRUE

   ◉ FALSE

   ○ TRUE

   ✓ **Correct**
   Array objects do not have
   many methods, and in particular they do not have a fill() method. Please review
   the methods available in the Arrays class.

4. Arrays of class type require a two-phase initialization. One phase to create the array and another phase to create the objects in the array.   **1 / 1 point**

   ○ FALSE

   ◉ TRUE

   ✓ **Correct**
   Arrays of class type only
   contain references to the actual objects.

5. Given an integer array named **arr**, the number of elements in the array can be determined by the following method call:
arr.length()

`1 / 1 point`

- ◉ FALSE
- ○ TRUE

✓ **Correct**
Please review what methods
are available in array objects and what data fields are available in them.

6. Which of the following statements is true (pick only one):

`1 / 1 point`

- ◉ An array can be sized dynamically, but it cannot be resized without allocating a new array.
- ○ An array cannot be sized dynamically when the program is running.
- ○ An array can be sized dynamically, and it can be resized without allocating a new array.

7. What is the value of the variable s after the following section of code executes?

`1 / 1 point`

```
1    int s = 0;
2    int [] a = {12, 23, 34, 45, 56};
3    for (int i=1; i<a.length; i++) {
4        s += a[i];
5    }
6
```

- ○ 0
- ○ 114
- ○ 170
- ◉ 158

✓ **Correct**
Carefully examine the for-loop and where it starts and stops.

8. What type of collection would we use if we wanted no duplicates?

`1 / 1 point`

- ○ Queue
- ○ List
- ◉ Set
- ○ Map

✓ **Correct**
Please review the various containers available in the Java Collections Framework in lecture 4 part 1.

9. Examine the following code:

`1 / 1 point`

```
1    ArrayList<String> list = new ArrayList<String>() ;
2
3    list.add("alpha");
4    list.add("bravo");
5    list.add("charlie");
6    list.add("delta");
```

```
 6    iist.auu( ueita );
 7    list.add("echo");
 8
```

Which of the following will replace the element "**charlie**" with "**foxtrot**"?

◯ list.set("charlie","foxtrot");

◯ list[2] = "foxtrot";

◉ list.set(list.indexOf("charlie"),"foxtrot");

◯ list.add("foxtrot", list.indexOf("charlie"));

◯ list.set("foxtrot","charlie");

✓ **Correct**
In this situation, we want to replace an element rather than add a new element. Review the methods of the ArrayList class in lecture 5.

10. Examine the following code: `1 / 1 point`

```
 1    ArrayList<String> list = new ArrayList<String>() ;
 2
 3    list.add("alpha");
 4    list.add("bravo");
 5    list.add("charlie");
 6    list.add("delta");
 7    list.add("echo");
 8
```

Which of the following will change the list so that it looks like:

```
 1    alpha
 2    bravo
 3    charlie
 4    delta
```

◯ list.clear("echo");

◯ list removellist sizell·

◯ list.clear("echo");

◯ list.remove(list.size());

◉ list.remove(list.size()-1);

◯ list.empty("echo");

◯ list.remove(5);

✓ **Correct**
In this case we want to delete an element from the ArrayList. There are several ways to do it. Review the methods available in the class.

11. Examine   `1 / 1 point`
the following code:

```
 1    ArrayList<String> list = new ArrayList<String>() ;
 2
 2    list sdd("slnhs").
```

11. Examine
the following code:

```
1    ArrayList<String> list = new ArrayList<String>() ;
2
3    list.add("alpha");
4    list.add("bravo");
5    list.add("charlie");
6    list.add("delta");
7    list.add("echo");
8
9    for ( _____ name : _____ ) {
10        out.println( _____ );
11    }
12
```

Fill in the blanks so that all the elements in the ArrayList are printed.

○
```
1    iterator()    hasNext()    next()
```

Fill in the blanks so that all the elements in the ArrayList are printed.

○
```
1    iterator()    hasNext()    next()
```

○
```
1    String    list    next()
```

◉
```
1    String    list    name
```

○
```
1    int    String    name
```

○
```
1    String    iterator()    next()
```