

Teaching Unit 6: ArrayList and Wrapper Classes

6.1 ArrayList Basics

Solutions

6.1.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
ArrayList<Integer> list = new ArrayList<Integer>();  
list.add(3); list.add(7);  
System.out.println(list.set(1, 9));  
System.out.println(list);
```

Output

```
7  
[3, 9]
```

set returns the old element and leaves the size unchanged.

6.1.2 - Write code

Build an ArrayList<String> containing "red", "green", and "blue". Replace "green" with "gold" and print the resulting list.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
ArrayList<String> colors = new ArrayList<String>();  
colors.add("red"); colors.add("green"); colors.add("blue");  
colors.set(1, "gold");  
System.out.println(colors);
```

Teaching Unit 6: ArrayList and Wrapper Classes

6.2 ArrayList Insert and Remove

Solutions

6.2.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
ArrayList<Integer> list = new ArrayList<Integer>();  
list.add(10); list.add(20); list.add(30);  
list.remove(1);  
System.out.println(list);
```

Output

[10, 30]

The integer argument selects index 1, so the value 20 is removed.

6.2.2 - Write code

Write public static void removeZeros(ArrayList<Integer> values). Remove every zero in place, including consecutive zeros, without skipping elements.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static void removeZeros(ArrayList<Integer> values) {  
    for (int i = values.size() - 1; i >= 0; i--) {  
        if (values.get(i) == 0) values.remove(i);  
    }  
}
```

Check your code

```
ArrayList<Integer> a = new ArrayList<Integer>(Arrays.asList(0,0,3,0));  
removeZeros(a); System.out.println(a);  
ArrayList<Integer> b = new ArrayList<Integer>(); removeZeros(b); System.out.println(b);
```

Expected output

[3]
[]

Teaching Unit 6: ArrayList and Wrapper Classes

6.3 ArrayList Traversal

Solutions

6.3.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
ArrayList<Integer> list = new ArrayList<Integer>(Arrays.asList(1,2,3));
int sum = 0;
for (int value : list) sum += value;
System.out.println(sum);
```

Output

6

Read-only enhanced traversal visits each element once.

6.3.2 - Write code

Write public static int totalLength(ArrayList<String> words). Assume all elements are non-null. Return the total number of characters, including 0 for an empty list.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static int totalLength(ArrayList<String> words) {
    int sum = 0;
    for (String word : words) sum += word.length();
    return sum;
}
```

Check your code

```
ArrayList<String> a = new ArrayList<String>(Arrays.asList("go", "Java", ""));
System.out.println(totalLength(a));
System.out.println(totalLength(new ArrayList<String>()));
```

Expected output

6
0

Teaching Unit 6: ArrayList and Wrapper Classes

6.4 ArrayList and Wrapper Classes

Solutions

6.4.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
Integer a = 150;  
Integer b = 150;  
System.out.println(a.equals(b));  
System.out.println(a.intValue() == b.intValue());
```

Output

```
true  
true
```

`equals` compares the Integer values; explicitly unboxed primitive values also compare numerically.

6.4.2 - Write code

Write public static double `mean(ArrayList<Integer> values)`. Assume a nonempty list of non-null values. Return an average that preserves the fractional part.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static double mean(ArrayList<Integer> values) {  
    int sum = 0;  
    for (int value : values) sum += value;  
    return (double) sum / values.size();  
}
```

Check your code

```
System.out.println(mean(new ArrayList<Integer>(Arrays.asList(2,3))));  
System.out.println(mean(new ArrayList<Integer>(Arrays.asList(-4))));
```

Expected output

```
2.5  
-4.0
```