Solutions to Quick Check Questions

10

Arrays

10.1 Array Basics

Quick Check

1. Which of the following statements are invalid?

```
a. float  number[23];

b. float  number = { 1.0f, 2.0f, 3.0f };

c. int    number;
   number = new Array[23];

d. int[]  number = [ 1, 2, 3, 4 ];
```

a. Invalid. You cannot set the size of an array at declaration. You do that when you create an array.

b. Invalid. The data type declaration must be float[ ].

c. Invalid. Array is an invalid term. It should be int[23] *for creating an array of* int.

d. Invalid. The parentheses are used for designating array elements as { 1, 2, 3, 4 }.

2. Write a code fragment to compute the sum of all positive real numbers stored in the following array.

```
double[] number = new double[25];
```

Answer:

```
double sum = 0;
for (int count = 0; count < number.length; count++) {
   if (number[count] > 0) {
      sum += number[count];
   }
}
```

3. Describe the difference between the following two code fragments.

```
//code fragment 1
for (int i = 0; i < number.length; i++) {
   if ( i % 2 == 0 ) {
      outputBox.printLine( number[i] );
   }
}

//code fragment 2
for (int i = 0; i < number.length; i++) {
   if ( number[i] % 2 == 0 ) {
      outputBox.printLine( number[i] );
   }
}
```

Fragment 1 prints out elements stored in the even index, i.e., number[0], number[2], number[4], and so forth.
Fragment 2 prints out even numbers in the array, regardless of the position they are stored in the array.

10.2 Arrays of Objects

1. Which of the following statements are invalid?

invalid → a. `Person[25] person;`

b. `Person[ ] person;`

c. `Person person[] = new Person[25];`

invalid → d. `Person person[25] = new Person[25];`

2. Write a code fragment to print out the name of those who are older than 20. Assume the following declaration and that the array is already set up correctly.

```
Person[ ]  friend = new Person[100];
```

Answer:

```
for (int count = 0; count < friend.length; count++) {
   if (friend[count].getAge() > 20) {
      System.out.println(friend[count].getName());
   }
}
```

10.3 The For-Each Loop

1. Rewrite the following for loop by using a for-each loop.

```
for (int i = 0; i < number.length; i++) {
   System.out.println(number[i]);
}
```

Answer:

```
for (int val : number) {
   System.out.println(val);
}
```

2. Rewrite the following for loop by using the standard for loop.

```
for (Person p : person) {
   System.out.println(p.getName());
}
```

Answer:

```
for (int i = 0; i < person.length; i++) {
   System.out.println(person[i].getName());
}
```

3. Why can't the following for loop be expressed as a for-each loop?

```
for (int i = 0; i < number.length; i++) {
   number[i] = number[i] + 50;
}
```

The for-each loop is a read-only loop in which you can only access the values in the collection. You are not allowed to modify the values so the given code that attempts to modify the array elements is not allowed.

10.4 Passing Arrays to Methods

1. What will be an output from the following code?

```
int[] list = {10, 20, 30, 40 };
myMethod( list );
System.out.println( list[1] );
System.out.println( list[3] );
...
public void myMethod(int[] intArray)
{
   for (int i = 0; i < intArray.length; i+=2) {
      intArray[i] = i;
   }
}
```

Answer:

```
20
40
```

list[0] and list[2] are changed to 0 and 2, but the calling side is printing list[1] and list[3] and therefore we did not see any changes. If we change the output statement to print out list[0] and list[2], then we have

```
0
2
```

as the output.

2. If we replace myMethod of question 1 with the following, what will be an output?

```
public void myMethod(int[] intArray)
{
    int[] local = intArray;
    for (int i = 0; i < local.length; i+=2) {
        local[i] = i;
    }
}
```

Answer:

```
20
40
```

Creating a local array does not create a duplicate array. All we're doing here is setting another reference to the same array. If we changed the output statement to print out list[0] and list[2], the output will be

```
0
2
```

10.5 Two-Dimensional Arrays

1. Write a code fragment to compute the average pay of the pays stored in the payScaleTable array.

Answer:

```
int count = 0;
double sum = 0;
for (int row = 0; row < payScaleTable.length; row++) {
    for (int col = 0; col < payScaleTable[row].length;
                                                  col++) {
       count++;
       sum += payScale[row][col];
    }
}
double average = sum / count;
```

2. Write a code fragment that finds the largest integer in the following two-dimensional array.

```
int[][] table = new int[10][10];
```

Answer:

```
int max = table[0][0];
for (int row = 0; row < table.length; row++) {
   for (int col = 0; col < table[row].length; col++) {
      if (max < table[row][col]) {
         max = table[row][col];
      }
   }
}
```

3. What is an output from the following code.

```
int[][] table = new int[10][5];

System.out.println(table.length);
System.out.println(table[4].length);
```

Answer:

```
10
5
```

10.6 Lists and Maps

1. What is the output from the following code?

```
List<String> list = new ArrayList<String>(  );
```

```
for (int i = 0; i < 6; i++ ) {
   list.add( "element " + i );
   System.out.println( list.size( ));
}
```

Answer

```
1
2
3
4
5
6
```

2. What is the output from the following code?

```
List<String> list = new ArrayList<String>( );

for (int i = 0; i < 6; i++ ) {
   list.add( "element " + i );
}

list.remove( 1 );
list.remove( 3 );

System.out.println( list.get( 2 ) );
```

Answer

```
element 3
```

Notice that after the item at position 1 is removed, the list is shorted by one. So at this point, the item at position 3 is [elememt 4].

3. Identify all errors in the following code.

```
1 ──► List<String> list = new ArrayList<Integer>();
2 ──► List<Person> people = new List<Person>();
3 ──► Map<String> table = new Map();
```

1. Parametized type String and Integer are not compatible.

2. The List is an interface. We cannot create an instance of an interface..

3. The Map is an interface. We cannot create an instance of an interface. The Map interface requires two parametized types for the key and the value.

10.7 Sample Development: The Address Book

No Quick Check questions.