[2]

Declare a constant:

final int ARRAY_SIZE = 10;

Declare and initialize the array:

double[] fractions = new double[ARRAY_SIZE];

Refer to element 4:

double fourthElement = fractions[3];

Assign value to element 9 and 6:

fractions[9] = 1.667; fractions[6] = 3.333;

Sum all elements:

double sum = 0; for (int x = 0; x < fractions.length; x++) { sum += fractions[x]; }

[3]

Create array of five integers:

int[] numbers = new int[5];

Assign values from keyboard (no loop):

Scanner sc = new Scanner(System.in); numbers[0] = sc.nextInt(); numbers[1] = sc.nextInt(); numbers[2] = sc.nextInt(); numbers[3] = sc.nextInt(); numbers[4] = sc.nextInt();

Assign values with a loop:

for (int i = 0; i < numbers.length; i++) { numbers[i] = sc.nextInt(); }

Print without loop:

System.out.println(numbers[0] + " " + numbers[1] + " " + numbers[2] + " " + numbers[3] + " " + numbers[4]);

Print with loop:

for (int num : numbers) { System.out.print(num + " "); }

[4]

float min = Float.MAX_VALUE, max = Float.MIN_VALUE;

```
for (int i = 0; i < w.length; i++) {
   if (w[i] < min) min = w[i];
   if (w[i] > max) max = w[i];
}
System.out.println("Min: " + min + ", Max: " + max);
```

[5]

```
// Odd numbers
for (int num : nums) {
   if (num % 2 != 0) System.out.print(num + " ");
}
// Even numbers
for (int num : nums) {
   if (num % 2 == 0) System.out.print(num + " ");
}
```

[6]

```
int sum = 0, max = Integer.MIN_VALUE, min = Integer.MAX_VALUE, evenCount = 0, oddCount = 0;
for (int num : nums) {
   sum += num;
   if (num > max) max = num;
   if (num < min) min = num;
   if (num % 2 == 0) evenCount++;
   else oddCount++;
}
System.out.println("Sum: " + sum);
System.out.println("Max: " + max);
System.out.println("Min: " + min);
System.out.println("Even Count: " + evenCount);
```

```
System.out.println("Odd Count: " + oddCount);
// Even indices
for (int i = 0; i < nums.length; i += 2) {
   System.out.print(nums[i] + " ");
}
System.out.println();

// Odd indices
for (int i = 1; i < nums.length; i += 2) {
   System.out.print(nums[i] + " ");
}
```

[7]

```
System.out.println(Arrays.toString(ar));
for (int i = 0; i < ar.length; i++) {
   ar[i]++;
}
System.out.println(Arrays.toString(ar));
if (ar.length == br.length) {
   System.out.println("Both arrays are the same size");
}
for (int i = 0; i < ar.length; i++) {
   ar[i] += br[i];
}
System.out.println(Arrays.toString(ar));
System.arraycopy(br, 0, ar, 0, br.length);
System.out.println(Arrays.toString(ar));
```

[10]

int[] a;: Correct. Declares an array of integers.

int []b;: Correct. Different syntax, but still valid.

int e[5];: Incorrect. Java does not specify the size in the declaration.

int c[];: Correct. Similar to int[] a.

int [d];: Incorrect. The array size should not be declared this w

[11]

int[] a = new int[5];: Correct. Initializes an array of size 5.

int[] b = new int[];: Incorrect. Size or data must be specified.

int[] c = [10, 20, 30, 40, 50];: Incorrect. Missing new int.

int[] d = {10, 20, 30, 40, 50};: Correct. Array is implicitly initialized.

int[] e = new int[]{10, 20, 30, 40, 50};: Correct. Explicit array allocation.

int[] f = new int[5]{10, 20, 30, 40, 50};: Incorrect. Can't declare both size and data.

int[] g = new int[0];: Correct. Initializes an empty array.

[12]

array = new int[5];: Correct. Assigns a 5-element array.

array = new int[10];: Correct. Assigns a 10-element array.

array = new int[-5];: Incorrect. Negative array size will cause an error.

array = {10, 20, 30, 40, 50};: Incorrect. Array cannot be initialized like this after declaration.

array = new int[]{10, 20, 30, 40, 50};: Correct. Assigns an array explicitly.

array = new int[]{};: Correct. Initializes an empty array.

[13]

boolean: false

char: \u0000

byte, short, int, long: 0

float, double: 0.0

Object, Arrays: null

[14]

int[] array = {5, 4, 3, 2, 6, 7, 8, 9, 0, 1};: Declares an array.

array.length;: Correct. Retrieves array length.

array.length();: Incorrect. .length is not a method.

array.size();: Incorrect. .size() does not exist for arrays.

array.size;: Incorrect. Arrays don't have a size attribute.

array.length - 1;: Valid but not the accurate array length.

[15]

Int a = new int[10];: Incorrect. Int is capitalized and not recognized as a primitive type.

int b = new int[10].length;: Correct. This will set b to 10.

int c = {10, 20, 30, 40}.length;: Incorrect. Cannot use curly brackets this way.

int d = new int[]{10, 20, 30, 40}.length;: Correct. This will set d to 4.

int e = new double[]{1.1, 1.2, 1.5, 1.4}.length;: Incorrect. int cannot store floating-point lengths.

int f = new int[]{10, 20, 30, 40}[2];: Correct. Sets f to 30.

int[] g = new int[]{10, 20, 30, 40}[2];: Incorrect. Indexing directly doesn't provide an array.

int h = new double[]{1.1, 1.2, 1.5, 1.4}[2];: Correct, but data type mismatch.

[16]

byte a = 10;: Correct.

short a = 10;: Correct.

int a = 10;: Correct.

long a = 10;: Correct.

float a = 10;: Incorrect. Floats require 10f.

double a = 10;: Correct.

char a = 'A';: Correct.

int[] a = new int[10];: Incorrect. Conflicting variable types.

[17]

```
class Example {
   public static void increment(int x, int[] y) {
      x++; // Increment local copy of primitive `x`
      y[0]++; // Increment actual first element of array `y`
   }

   public static void main(String[] args) {
      int x = 100;
      int[] y = {200};
      System.out.println(x + " " + y[0]); // Output: 100 200
      increment(x, y);
      System.out.println(x + " " + y[0]); // Output: 100 201
   }
}
```

[18]

```
public static char[] merge(char[] array1, char[] array2) {
   char[] merged = new char[array1.length + array2.length];
   System.arraycopy(array1, 0, merged, 0, array1.length);
   System.arraycopy(array2, 0, merged, array1.length, array2.length);
   return merged;
}
```

[19]

```
import java.util.Arrays;
class Example {
   public static void main(String[] args) {
      int[] array = {100, 200, 300};
```

```
        System.out.println(Arrays.toString(array)); // Output: [100, 200, 300]

        // First loop: does not modify the actual array
        for (int a : array) { a++; }
        System.out.println(Arrays.toString(array)); // Output: [100, 200, 300]

        // Second loop: modifies the actual array
        for (int i = 0; i < array.length; i++) {
            array[i]++;
        }
        System.out.println(Arrays.toString(array)); // Output: [101, 201, 301]
    }
}
```

[20]

printArray(a);: Correct.

printArray(b);: Correct.

printArray(c);: Correct.

printArray(d);: Correct.

printArray(new int[]{});: Correct.

printArray(new int[5]);: Correct.

printArray(new int[]{10, 20, 30, 40});: Correct.

Others: Invalid syntax or arguments.

[21]

x = xr[0];: Correct.

xr[0] = x;: Correct.

x = xr;: Incorrect. int[] to int.

xr = x;: Incorrect. int to int[].

dr[0] = xr[0];: Incorrect. Incompatible types.

xr[0] = dr[0];: Incorrect. Cannot convert double

[22]

```
import java.util.Scanner;

public class UniqueValues {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int[] uniqueNumbers = new int[5];
      int count = 0;

      for (int i = 0; i < uniqueNumbers.length; i++) {
         System.out.print("Enter a number between 10 and 100: ");
         int num = scanner.nextInt();

         boolean isDuplicate = false;
         for (int j = 0; j < count; j++) {
            if (uniqueNumbers[j] == num) {
               isDuplicate = true;
               break;
            }
         }

         if (!isDuplicate) {
            uniqueNumbers[count] = num;
            count++;
            System.out.print("Unique numbers so far: ");
            for (int k = 0; k < count; k++) {
```

```
            System.out.print(uniqueNumbers[k] + " ");
          }
          System.out.println();
        }
      }
    }
}
```

[23]

```
public static void copyRange(int[] a1, int[] a2, int i1, int i2, int length) {
  System.arraycopy(a1, i1, a2, i2, length);
}
```

[24]

```
public static void main(String[] args) {
  int[] sourceArray = {1, 2, 3, 4, 5, 6, 7, 8};
  int[] destArray = new int[10];
  copyRange(sourceArray, destArray, 2, 3, 4);
  System.out.println(Arrays.toString(destArray)); // Output: [0, 0, 0, 3, 4, 5, 6, 0, 0, 0]
}
```

[25]

```
// Insert a number at the end
public static int[] insert(int[] array, int number) {
  int size = size(array);
  if (size == array.length) {
    array = Arrays.copyOf(array, size + 1);
  }
  array[size] = number;
```

```
    return array;
}

// Print all numbers in the list
public static void printList(int[] array) {
    System.out.println(Arrays.toString(array));
}

// Remove the last number
public static int[] remove(int[] array) {
    int size = size(array);
    if (size > 0) {
        array[size - 1] = 0;
    }
    return array;
}

// Remove a specific index
public static int[] remove(int[] array, int index) {
    int size = size(array);
    if (index >= 0 && index < size) {
        System.arraycopy(array, index + 1, array, index, size - index - 1);
        array[size - 1] = 0;
    }
    return array;
}

// Insert at a specific index
public static int[] insert(int[] array, int number, int index) {
```

```
    int size = size(array);
    if (index >= 0 && index <= size) {
        if (size == array.length) {
            array = Arrays.copyOf(array, size + 1);
        }
        System.arraycopy(array, index, array, index + 1, size - index);
        array[index] = number;
    }
    return array;
}

// Get the size of the list
public static int size(int[] array) {
    int count = 0;
    for (int i : array) {
        if (i != 0) {
            count++;
        }
    }
    return count;
}

// Check if the list is empty
public static boolean isEmpty(int[] array) {
    return size(array) == 0;
}

// Check if the list is full
public static boolean isFull(int[] array) {
```

```
    return size(array) == array.length;
}

// Clear the list
public static void clear(int[] array) {
    Arrays.fill(array, 0);
}

// Remove duplicates from the list
public static int[] removeDuplicates(int[] array) {
    Set<Integer> set = new HashSet<>();
    int[] result = new int[array.length];
    int index = 0;
    for (int number : array) {
        if (number != 0 && set.add(number)) {
            result[index++] = number;
        }
    }
    return Arrays.copyOf(result, index);
}

// Search for a specific number
public static int search(int[] array, int number) {
    for (int i = 0; i < array.length; i++) {
        if (array[i] == number) {
            return i;
        }
    }
    return -1;
```

```
}

// Check if a specific number exists
public static boolean isExist(int[] array, int number) {
    return search(array, number) != -1;
}
```