hi

Building Java Programs

Chapter 7
Lecture 7-2: Arrays

reading: 7.1

Quick array initialization

type`[]` **name** = {**value**, **value**, … **value**};

- Example:

```
int[] numbers = {12, 49, -2, 26, 5, 17, -6};
```

index	0	1	2	3	4	5	6
value	12	49	-2	26	5	17	-6

- Useful when you know what the array's elements will be
- The compiler figures out the size by counting the values

"Array mystery" problem

- **traversal**: An examination of each element of an array.
- What element values are stored in the following array?

```
int[] a = {1, 7, 5, 6, 4, 14, 11};
for (int i = 0; i < a.length - 1; i++) {
    if (a[i] > a[i + 1]) {
        a[i + 1] = a[i + 1] * 2;
    }
}
```

index	0	1	2	3	4	5	6
value	1	7	10	12	8	14	22

Limitations of arrays

- You cannot resize an existing array:

```
int[] a = new int[4];
a.length = 10;          // error
```

- You cannot compare arrays with == or `equals`:

```
int[] a1 = {42, -7, 1, 15};
int[] a2 = {42, -7, 1, 15};
if (a1 == a2) {  ... }          // false!
if (a1.equals(a2)) {  ... }     // false!
```

- An array does not know how to print itself:

```
int[] a1 = {42, -7, 1, 15};
System.out.println(a1);         // [I@98f8c4]
```


The `Arrays` class

- Class `Arrays` in package `java.util` has useful static methods for manipulating arrays:

Method name	Description
`binarySearch(`**array**, **value**`)`	returns the index of the given value in a *sorted* array (or < 0 if not found)
`copyOf(`**array**, **length**`)`	returns a new copy of an array
`equals(`**array1**, **array2**`)`	returns `true` if the two arrays contain same elements in the same order
`fill(`**array**, **value**`)`	sets every element to the given value
`sort(`**array**`)`	arranges the elements into sorted order
`toString(`**array**`)`	returns a string representing the array, such as `"[10, 30, -25, 17]"`

- Syntax: `Arrays.`**methodName**`(`**parameters**`)`

`Arrays.toString`

- `Arrays.toString` accepts an array as a parameter and returns a `String` representation of its elements.

```
int[] e = {0, 2, 4, 6, 8};
e[1] = e[3] + e[4];
System.out.println("e is " + Arrays.toString(e));
```

Output:

```
e is [0, 14, 4, 6, 8]
```

- Must `import java.util.*;`

bye

hi

Weather question 2

- Modify the weather program to print the following output:

```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.

Temperatures: [45, 44, 39, 48, 37, 46, 53]
Two coldest days: 37, 39
Two hottest days: 53, 48
```


Weather answer 2

```
// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather2 {
    public static void main(String[] args) {
        ...
        int[] temps = new int[days];        // array to store days' temperatures
        ...   (same as Weather program)

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");


        System.out.println("Temperatures: " + Arrays.toString(temps));
        Arrays.sort(temps);
        System.out.println("Two coldest days: " + temps[0] + ", " + temps[1]);
        System.out.println("Two hottest days: " + temps[temps.length - 1] +
                           ", " + temps[temps.length - 2]);
    }
}
```


Building Java Programs

Chapter 7
Lecture 7-2: Arrays as Parameters

reading: 7.1 – 7.3

Swapping values

```
public static void main(String[] args) {
    int a = 7;
    int b = 35;

    // swap a with b?
    a = b;
    b = a;

    System.out.println(a + " " + b);
}
```

- What is wrong with this code? What is its output?

- The red code should be replaced with:

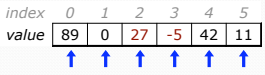
```
int temp = a;
a = b;
b = temp;
```


Array reversal question

- Write code that reverses the elements of an array.
  - For example, if the array initially stores:
    `[11, 42, -5, 27, 0, 89]`
  - Then after your reversal code, it should store:
    `[89, 0, 27, -5, 42, 11]`
    - The code should work for an array of any size.
    - Hint: think about swapping various elements...

Algorithm idea

- Swap pairs of elements from the edges; work inwards:



bye

hi

Flawed algorithm

- What's wrong with this code?

```
int[] numbers = [11, 42, -5, 27, 0, 89];
// reverse the array
for (int i = 0; i < numbers.length; i++) {
    int temp = numbers[i];
    numbers[i] = numbers[numbers.length - 1 - i];
    numbers[numbers.length - 1 - i] = temp;
}
```

- The loop goes too far and un-reverses the array! Fixed version:

```
for (int i = 0; i < numbers.length / 2; i++) {
    int temp = numbers[i];
    numbers[i] = numbers[numbers.length - 1 - i];
    numbers[numbers.length - 1 - i] = temp;
}
```


Array reverse question 2

- Turn your array reversal code into a `reverse` method.
  - Accept the array of integers to reverse as a parameter.

```
int[] numbers = {11, 42, -5, 27, 0, 89};
reverse(numbers);
```

  - How do we write methods that accept arrays as parameters?
  - Will we need to return the new array contents after reversal?
  - ...

Array parameter (declare)

```
public static type methodName(type[] name) {
```

- Example:

```
// Returns the average of the given array of numbers.
public static double average(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    return (double) sum / numbers.length;
}
```

  - You don't specify the array's length (but you can examine it).

Array parameter (call)

```
methodName(arrayName);
```

- Example:

```
public class MyProgram {
    public static void main(String[] args) {
        // figure out the average TA IQ
        int[] iq = {126, 84, 149, 167, 95};
        double avg = average(iq);
        System.out.println("Average IQ = " + avg);
    }
    ...
```

  - Notice that you don't write the `[]` when passing the array.

Array return (declare)

```
public static type[] methodName(parameters) {
```

- Example:

```
// Returns a new array with two copies of each value.
// Example: [1, 4, 0, 7] -> [1, 1, 4, 4, 0, 0, 7, 7]
public static int[] stutter(int[] numbers) {
    int[] result = new int[2 * numbers.length];
    for (int i = 0; i < numbers.length; i++) {
        result[2 * i]     = numbers[i];
        result[2 * i + 1] = numbers[i];
    }
    return result;
}
```


Array return (call)

```
type[] name = methodName(parameters);
```

- Example:

```
public class MyProgram {
    public static void main(String[] args) {
        int[] iq = {126, 84, 149, 167, 95};
        int[] stuttered = stutter(iq);
        System.out.println(Arrays.toString(stuttered));
    }
    ...
```

- Output:

```
[126, 126, 84, 84, 149, 149, 167, 167, 95, 95]
```


bye

hi

Reference semantics

reading: 7.1, 3.3, 4.3

Value semantics

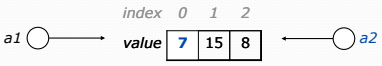
- **value semantics**: Behavior where values are copied when assigned, passed as parameters, or returned.
  - All primitive types in Java use value semantics.
  - When one variable is assigned to another, its value is copied.
  - Modifying the value of one variable does not affect others.

```
int x = 5;
int y = x;      // x = 5, y = 5
y = 17;         // x = 5, y = 17
x = 8;          // x = 8, y = 17
```


Reference semantics (objects)

- **reference semantics**: Behavior where variables actually store the address of an object in memory.
  - When one variable is assigned to another, the object is *not* copied; both variables refer to the *same object*.
  - Modifying the value of one variable *will* affect others.

```
int[] a1 = {4, 15, 8};
int[] a2 = a1;          // refer to same array as a1
a2[0] = 7;
System.out.println(Arrays.toString(a1)); // [7, 15, 8]
```



Arrays pass by reference

- Arrays are passed as parameters by *reference*.
  - Changes made in the method are also seen by the caller.

```
public static void main(String[] args) {
    int[] iq = {126, 167, 95};
    increase(iq);
    System.out.println(Arrays.toString(iq));
}
public static void increase(int[] a) {
    for (int i = 0; i < a.length; i++) {
        a[i] = a[i] * 2;
    }
}
```

- Output:

```
[252, 334, 190]
```

Array reverse question 2

- Turn your array reversal code into a `reverse` method.
  - Accept the array of integers to reverse as a parameter.

```
int[] numbers = {11, 42, -5, 27, 0, 89};
reverse(numbers);
```

- Solution:

```
public static void reverse(int[] numbers) {
    for (int i = 0; i < numbers.length / 2; i++) {
        int temp = numbers[i];
        numbers[i] = numbers[numbers.length - 1 - i];
        numbers[numbers.length - 1 - i] = temp;
    }
}
```


Array return question

- Write a method `merge` that accepts two arrays of integers and returns a new array containing all elements of the first array followed by all elements of the second.

```
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};

int[] a3 = merge(a1, a2);
System.out.println(Arrays.toString(a3));
// [12, 34, 56, 7, 8, 9, 10]
```

- Write a method `merge3` that merges 3 arrays similarly.

```
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};
int[] a3 = {444, 222, -1};

int[] a4 = merge3(a1, a2, a3);
System.out.println(Arrays.toString(a4));
// [12, 34, 56, 7, 8, 9, 10, 444, 222, -1]
```


bye

hi

Array return answer 1

```
// Returns a new array containing all elements of a1
// followed by all elements of a2.
public static int[] merge(int[] a1, int[] a2) {
    int[] result = new int[a1.length + a2.length];

    for (int i = 0; i < a1.length; i++) {
        result[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        result[a1.length + i] = a2[i];
    }

    return result;
}
```


Array return answer 2

```
// Returns a new array containing all elements of a1,a2,a3.
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    int[] a4 = new int[a1.length + a2.length + a3.length];

    for (int i = 0; i < a1.length; i++) {
        a4[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        a4[a1.length + i] = a2[i];
    }
    for (int i = 0; i < a3.length; i++) {
        a4[a1.length + a2.length + i] = a3[i];
    }

    return a4;
}

// Shorter version that calls merge.
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    return merge(merge(a1, a2), a3);
}
```


bye