MODIFYING ARRAYS

zyBook 6.6, zyBook 6.7, zyBook 6.8

@ Dr. Jessica Young Schmidt and NCSU Computer Science Faculty

MODIFYING ARRAYS

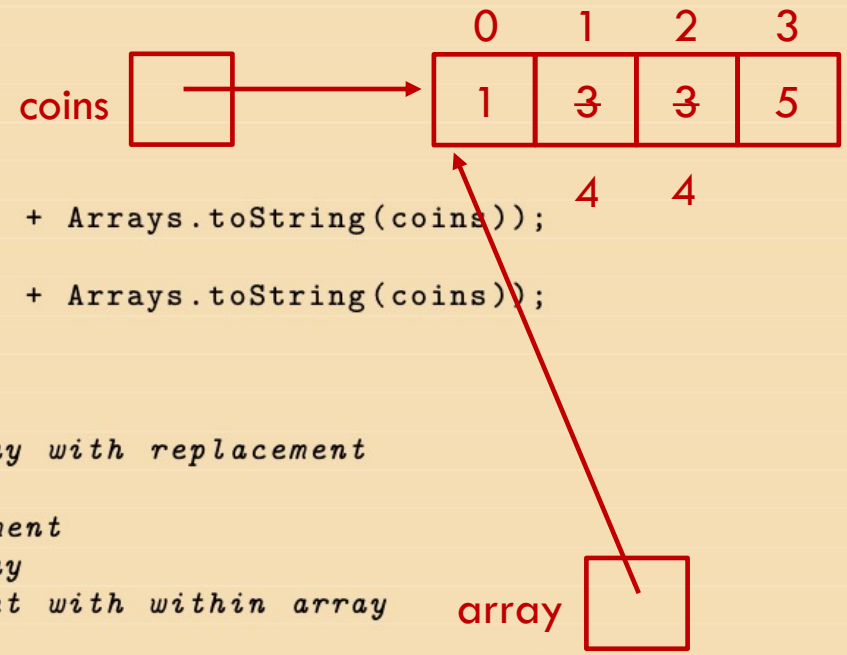
We can update and modify arrays in the same way that we can update and modify any variables.

```
int x = 0;
x = 3;
x++;
x--;
x *= 2;

int[] list = new int[3];
list[1] = 3;
list[2]++;
list[0]--;
list[0] *= 2;
```

REPLACING

```
/**
 * Replace all instances of target in array with replacement
 *
 * @param array int array use for replacement
 * @param target int to search for in array
 * @param replacement int to replace target with within array
 */
public static void replaceAll(int[] array, int target, int replacement) {
  // Traverse array to search for target element & replace
  for (int i = 0; i < array.length; i++) {
    if (array[i] == target) {
      // We have found the target element.
      // We replace the element with replacement.
      array[i] = replacement;
    }
  }
}
```

```java
import java.util.Arrays;
/**
 * Replaces a given element value
 * @author Reges & Stepp - Building Java Programs: A Back to Basics Approach
 * @author Jessica Young Schmidt
 */
public class ReplaceElements {
  /**
   * Starts program
   * @param args command line arguments
   */
  public static void main(String[] args) {
    int[] coins = { 1, 3, 3, 5 };
    System.out.println("Before replaceAll: " + Arrays.toString(coins));
    replaceAll(coins, 3, 4);
    System.out.println("After replaceAll:  " + Arrays.toString(coins));
  }

  /**
   * Replace all instances of target in array with replacement
   *
   * @param array int array use for replacement
   * @param target int to search for in array
   * @param replacement int to replace target with within array
   */
  public static void replaceAll(int[] array, int target, int replacement) {
    for (int i = 0; i < array.length; i++) { // traverse array to search for target element & replace
      if (array[i] == target) { // We have found the target element.
        array[i] = replacement;  // We replace the element with replacement.
      }
    }
  }
}
```




```
$ java -cp bin ReplaceElements
Before replaceAll: [1, 3, 3, 5]
After replaceAll:  [1, 4, 4, 5]
```

SWAPPING VALUES

Would this work to swap the values of x and y?

```
int x = 5;
int y = 1;

x = y;

y = x;
```

x	~~5~~ 1
y	1

```
int x = 5;
int y = 1;

int temp = x;

x = y;

y = temp;
```

x	~~5~~ 1
y	~~1~~ 5
temp	5

SWAP METHOD FOR INT

- Method does not work due to scope and int primitive type:
  - Parameters are copies of integer values passed to method
  - Only changes values of parameters

```
public class SwapInts {
  public static void main(String[] args) {
    int x = 1;
    int y = 16;
    System.out.println("Before swap (main): " + x + ", " + y);
    swap(x, y);
    System.out.println("After swap (main): " + x + ", " + y);
  }

  public static void swap(int x, int y) {
    System.out.println("--Before swap (swap): " + x + ", " + y);
    int temp = x;
    x = y;
    y = temp;
    System.out.println("--After swap (swap): " + x + ", " + y);
  }
}
```

```
$ java -cp bin SwapInts
Before swap (main): 1, 16
--Before swap (swap): 1, 16
--After swap (swap): 16, 1
After swap (main): 1, 16
```

x = 1
y = 16
temp = 1
x = 16
y = 1

SWAP METHOD FOR ARRAY ELEMENTS

- Method works with arrays because arrays are stored as objects
- When you pass an array as a parameter to a method, the method has the ability to change the contents of the array

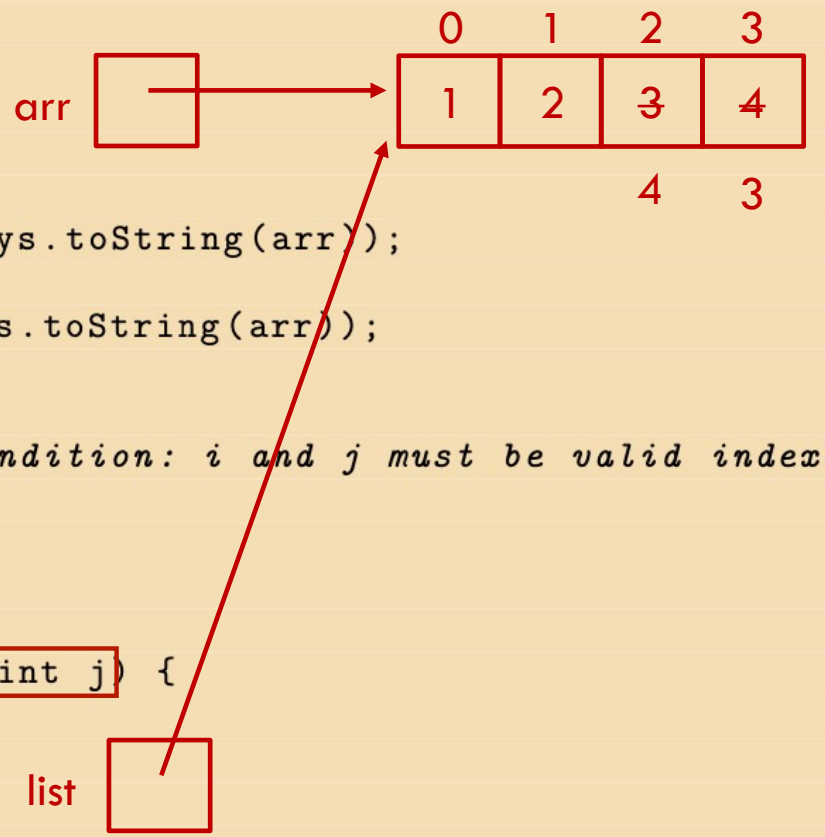
```
/**
 * Swaps elements at indexes i and j. Precondition: i and j must be valid
 * indexes of list
 *
 * @param list array of integers
 * @param i index of first element in swap
 * @param j index of second element in swap
 */
public static void swap(int[] list, int i, int j) {
  int temp = list[i];
  list[i] = list[j];
  list[j] = temp;
}
```

```java
import java.util.Arrays;
/**
 * Array swapping algorithm
 * @author Reges & Stepp
 * @author Jessica Young Schmidt
 */
public class SwapArray {
  /**
   * Starts program
   * @param args command line arguments
   */
  public static void main(String[] args) {
    int[] arr = { 1, 2, 3, 4 };
    System.out.println("Before swap: " + Arrays.toString(arr));
    swap(arr, 2, 3);
    System.out.println("After swap: " + Arrays.toString(arr));
  }
  /**
   * Swaps elements at indexes i and j. Precondition: i and j must be valid indexes
   * @param list array of integers
   * @param i index of first element in swap
   * @param j index of second element in swap
   */
  public static void swap(int[] list, int i, int j) {
    int temp = list[i];
    list[i] = list[j];
    list[j] = temp;
  }
}
```

i = 2

j = 3

temp = 3




```
$ java -cp bin SwapArray
Before swap: [1, 2, 3, 4]
After swap: [1, 2, 4, 3]
```

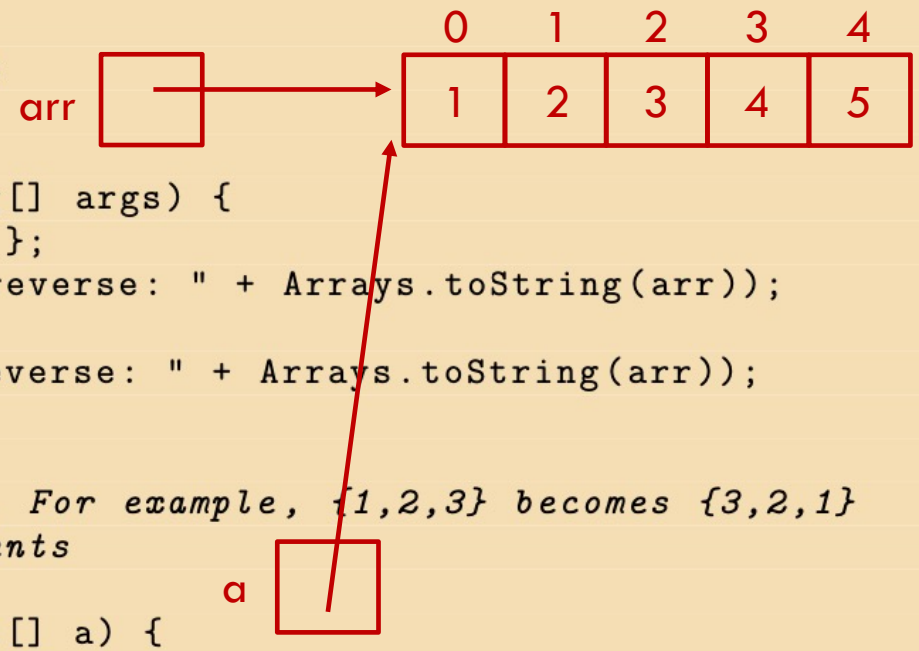
REVERSING ARRAY

Reverse order of elements stored in array. [3, 8, 7, -2, 14, 78] would become [78, 14, -2, 7, 8, 3].

```
Algorithm reverseArray
Input array, a, to reverse
Output array, a, with elements in reverse order of input

- swap elements at index 0 and index length-1
- swap elements at index 1 and index length-2
- continue until array is completely reversed
```

```java
/**
 * Reverses the elements of a. For example, {1,2,3} becomes {3,2,1}
 *
 * @param a int array of elements
 */
public static void reverse(int[] a) {
  // Traverse through first half of a. We only want to traverse through first half
  // because if we traverse through the whole array then we will reverse a twice
  // and be back to starting value.
  for (int i = 0; i < a.length / 2; i++) {
    // j is the index that we will swap with index i. 0 with (length-1), 1 with
    // (length-1)-1, ... i with (length-1)-i.
    int j = a.length - i - 1;
    swap(a, i, j); // swap elements at two indexes
  }
}

/**
 * Swaps elements at indexes i and j. Precondition: i and j must be valid
 * indexes of list
 *
 * @param list array of integers
 * @param i index of first element in swap
 * @param j index of second element in swap
 */
public static void swap(int[] list, int i, int j) {
  int temp = list[i];
  list[i] = list[j];
  list[j] = temp;
}
```

```java
import java.util.Arrays;
/**
 * Array reversing algorithm
 * @author Reges & Stepp
 * @author Jessica Young Schmidt
 */
public class ReverseArray {
  public static void main(String[] args) {
    int[] arr = { 1, 2, 3, 4, 5 };
    System.out.println("Before reverse: " + Arrays.toString(arr));
    reverse(arr);
    System.out.println("After reverse: " + Arrays.toString(arr));
  }
  /**
   * Reverses the elements of a. For example, {1,2,3} becomes {3,2,1}
   * @param a int array of elements
   */
  public static void reverse(int[] a) {
    for (int i = 0; i < a.length / 2; i++) {
      int j = a.length - i - 1;
      swap(a, i, j); // swap elements at two indexes
    }
  }
  public static void swap(int[] list, int i, int j) {
    int temp = list[i];
    list[i] = list[j];
    list[j] = temp;
  }
}
```




```
$ java -cp bin ReverseArray
Before reverse: [1, 2, 3, 4, 5]
After reverse: [5, 4, 3, 2, 1]
```

0	1	2	3	4
1	2	3	4	5

i = 0, j = 4

0	1	2	3	4
5	2	3	4	1

i = 1, j = 3

0	1	2	3	4
5	4	3	2	1

i = 2

0	1	2	3	4
5	4	3	2	1