Ch. 5 – pt2

Arrays

Insert Item in Array

```
// Insert item into array
public class InsertItem{
   public static void main(String[] args){
      int[] oldArray = {1, 1, 2, 3, 5, 13, 21};

      // we notice our fibonacci sequence is missing 8
      // we add it in at index 5
      int index = 5;
      int num = 8;

      int[] newArray = new int[oldArray.length + 1];

      for(int i = 0; i<index; i++)
         newArray[i] = oldArray[i];

      newArray[index] = num;
      for(int i = index + 1; i<newArray.length; i++)
         newArray[i] = oldArray[i – 1];
}
```

`main(String[] args)`

- args is an array of Strings passed at the command line with the java command

Copying Arrays

Element by element

```
// Assume destination and source are arrays of the same length
for(int i = 0; i < destination.length; i++){
   destination[i] = source[i];
}
```

Copying reference

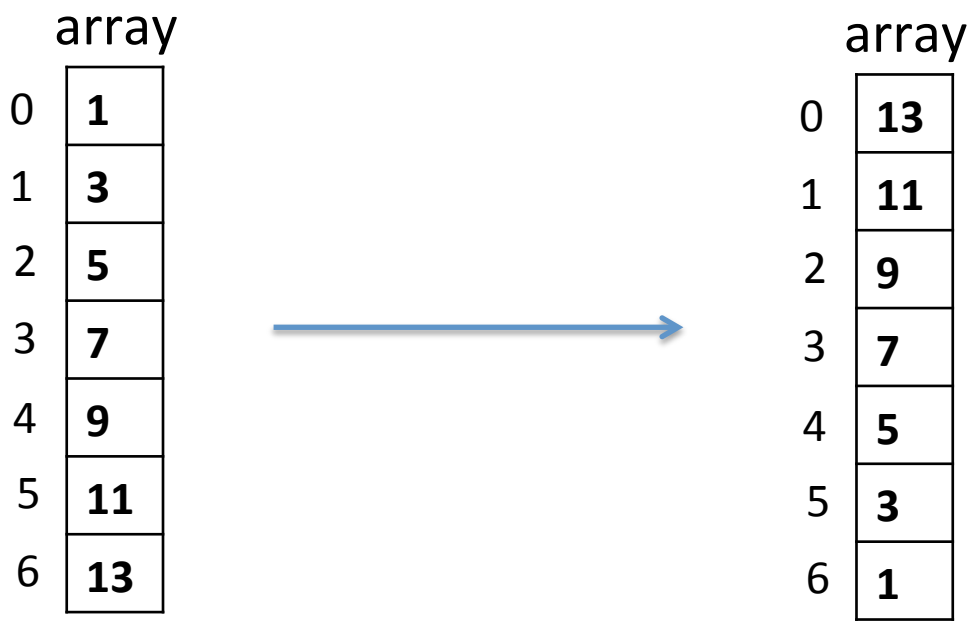
```
// destination and source don't need to be the same length

source = destination;
```

Copying Arrays

```
// What is printed out here?
public class TestCopy{
   public static void main(String[] args){
      int[] to = {1, 2, 3, 4};
      int[] from = {5, 6, 7, 8};
      for(int i = 0; i<from.length; i++){
         from[i] = to[i];
      }
      to[0] = 10;
      System.out.println(from[0]);
   }
}
```

Reversing Arrays



Passing and returning arrays

```
// Demo call by ref in action
public class PassArray{
   public static void main(String[] args){
      int[] ar = {0, 2, 4, 6, 8};
      alterArray(ar);
      for(int i = 0; i<ar.length; i++){
         System.out.println(i ": " + ar[i]);
      }
   }
   public static void alterArray(int[] a){
      if(a.length>0){
         for(int i = 0; i<a.length; i++){
            a[i] = a[i] * 2;
         }
      }
   }
}
```

Passing and returning arrays

```
// Demo returning an array
import java.util.Scanner;
public class ReturnArray{
   public static void main(String[] args){
      int[] ar = createArray();

      for(int i = 0; i<ar.length; i++){
         System.out.println(i + ": " + ar[i]);
      }
   }
   public static int[] createArray(){
      Scanner keyboard = new Scanner(System.in);
      int[] a = new int[4];
      for(int i = 0; i<a.length; i++)
         a[i] = keyboard.nextInt();
      return a;
   }
}
```

Call by val vs. Call by ref

- **Call by value**: With primitive data types, the literal value is passed as an argument to a method.
- **Call by reference**: With abstract data types, the pointer (memory address) is passed as an argument to a method.

Data Structures and Arrays

- Stack
  - Last in, first out (LIFO)
- Queue
  - First in, first out (FIFO)
- Push
- Pop
- Insert
- Remove

2 dimensional arrays

```
// Can represent rows by columns or columns by rows
import java.util.Scanner;
public class TwoDimensions{
   public static void main(String[] args){
      Scanner keyboard = new Scanner(System.in);
      int[][] twoD = new int[3][4];
      // Populate array
      for(int i = 0; i<twoD.length; i++){
         for(int j = 0; j<twoD[i].length; j++){
            twoD[i][j] = keyboard.nextInt();
         }
      }
      // Print array
      for(int i = 0; i<twoD.length; i++){
         for(int j = 0; j<twoD[i].length; j++){
            System.out.println("i: " + i + ", j: " + j + " " +
                                    twoD[i][j]);
         }
      }
   }
}
```

2 dimensional arrays

```
// Initializing 2-d array directly
public class TwoDimensionsDirect{
   public static void main(String[] args){
      int[][] ragged = {{1, 2, 3}, {4, 5}, {6, 7, 8, 9}};
      // Print array
      for(int i = 0; i<ragged.length; i++){
         for(int j = 0; j<ragged[i].length; j++){
            System.out.println("i: " + i + ", j: " + j + " " +
                                    ragged[i][j]);
         }
      }
   }
}
```

2 dimensional arrays

```
// Initializing 2-d array directly
public class TwoDimensionsDirect{
   public static void main(String[] args){
      int[][] ragged = {{1, 2, 3}, {4, 5}, {6, 7, 8, 9}};
      // Print array
      for(int i = 0; i<ragged.length; i++){
         for(int j = 0; j<ragged[i].length; j++){
            System.out.println("i: " + i + ", j: " + j + " " +
                                    ragged[i][j]);
         }
      }
   }
}
```

What is the value stored at **`ragged[0][2]`**?

N-dimensional arrays

```
// Maximum number of dimensions is 256
public class NDimensions{
   public static void main(String[] args){
      int[][][]...[] nDimArray = new int[5][5][5]...[5];
      ...
   }
}
```

Matrix Addition

```
// Write a method to accept two 2-dimensional arrays and
// return a 2-d array
// You can assume both arrays have the same dimensions
```

Matrix Multiplication

```
// Write a method to accept two 2-dimensional arrays and
// return a 2-d array
// You can assume both arrays have the same dimensions
```

Bubble Sort

```
// A method to accept an array and return an array in sorted
// order.
```