Array Modification via Parameter Passing

Mitsu Ogihara

Department of Computer Science
University of Miami

Table of Contents

args = arguments

- The method main has this `String[] args` as argument
- This is a String array
- This is the sequence of characters separated by a sequence of white spaces that are entered in the command line
  `java PRORGRAM-NAME`

Args

A program that prints the arg elements

```
public class Args {
  public static void main( String[] args ) {
    System.out.printf( "The array args has length %d.%n", args.length );
    for ( int index = 0; index < args.length; index ++ ) {
      System.out.printf( "args[%d] = %s%n", index, args[ index ] );
    }
  }
}
```

Print the element

Table of Contents

Passing

- If you pass an array to a method, the method can access the array
- Suppose a method `fooBar` requires an int array parameter. Then it is declared with

  ```
  TYPE fooBar(int[] data) { ...  }
  ```

  where `data` is the local name of the array.

Accessing Arrays Given as a Parameter

- Suppose `fooBar` executes the following two lines:
  - `data[0] = 0;` and
  - `data = new int[12];`
- Suppose `fooBar(myData)` is called where `myData` is an array with 10 elements { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
- Then the first operation is reflected on `myData` but not the second one

Accessing Arrays Given as a Parameter

- Suppose `fooBar` executes the following two lines:
  - `data[0] = 0;` and
  - `data = new int[12];`
- Suppose `fooBar(myData)` is called where `myData` is an array with 10 elements { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
- Then the first operation is reflected on `myData` but not the second one
- You must use a method that returns an array to change array size

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

Method header - returns an array
parameters are an input array, insertion position, and the new element

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

New array creation

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

Copying the elements before the insertion point

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

Place the insertion element

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

Copying the elements after the insertion point

Insertion and Deletion Example

```java
import java.util.Scanner;

public class ModifyArray {
  // class variable for the array
  public static String[] insert( String[] data, int p, String w ) {
    String[] newArray = new String[ data.length + 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    newArray[ p ] = w;
    for ( int index = p; index < data.length; index ++ ) {
      newArray[ index + 1 ] = data[ index ];
    }
    return newArray;
  }
```

Return the array

Insertion and Deletion Example

```
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

Method header - returns an array
parameters are an input array, insertion position, and the new element

Insertion and Deletion Example

```java
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

New array creation

Insertion and Deletion Example

```
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

Copying the elements before the deletion point

Insertion and Deletion Example

```
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

Copying the elements after the deletion point

Insertion and Deletion Example

```
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

Return the array

Insertion and Deletion Example

```java
  public static String[] remove( String[] data, int p) {
    String[] newArray = new String[ data.length - 1 ];
    for ( int index = 0; index < p; index ++ ) {
      newArray[ index ] = data[ index ];
    }
    for ( int index = p + 1; index < data.length; index ++ ) {
      newArray[ index - 1 ] = data[ index ];
    }
    return newArray;
  }
  public static void print( String[] data ) {
    for ( int index = 0; index < data.length; index ++ ) {
      System.out.printf( "%3d: %s%n", index, data[ index ] );
    }
  }
```

Print the data

Initial Array Creation

```
  public static String[] initial() {
    String[] data;
    Scanner keyboard = new Scanner( System.in );
    System.out.print( "Enter size: " );
    data = new String[ keyboard.nextInt() ];
    for ( int pos = 0; pos < data.length; pos ++ ) {
      System.out.print( "Enter element " + pos + " :" );
      data[ pos ] = keyboard.next();
    }
    return data;
  }
```

Method header - returns an array
parameters are an input array, insertion position, and the new element

Initial Array Creation

```java
  public static String[] initial() {
    String[] data;
    Scanner keyboard = new Scanner( System.in );
    System.out.print( "Enter size: " );
    data = new String[ keyboard.nextInt() ];
    for ( int pos = 0; pos < data.length; pos ++ ) {
      System.out.print( "Enter element " + pos + " :" );
      data[ pos ] = keyboard.next();
    }
    return data;
  }
```

New array creation

Initial Array Creation

```java
  public static String[] initial() {
    String[] data;
    Scanner keyboard = new Scanner( System.in );
    System.out.print( "Enter size: " );
    data = new String[ keyboard.nextInt() ];
    for ( int pos = 0; pos < data.length; pos ++ ) {
      System.out.print( "Enter element " + pos + " :" );
      data[ pos ] = keyboard.next();
    }
    return data;
  }
```

Prompt the user and receive the size; create the array

Initial Array Creation

```
  public static String[] initial() {
    String[] data;
    Scanner keyboard = new Scanner( System.in );
    System.out.print( "Enter size: " );
    data = new String[ keyboard.nextInt() ];
    for ( int pos = 0; pos < data.length; pos ++ ) {
      System.out.print( "Enter element " + pos + " :" );
      data[ pos ] = keyboard.next();
    }
    return data;
  }
```

Receive data for each element

Initial Array Creation

```
  public static String[] initial() {
    String[] data;
    Scanner keyboard = new Scanner( System.in );
    System.out.print( "Enter size: " );
    data = new String[ keyboard.nextInt() ];
    for ( int pos = 0; pos < data.length; pos ++ ) {
      System.out.print( "Enter element " + pos + " :" );
      data[ pos ] = keyboard.next();
    }
    return data;
  }
```

Return the array

Joining Two Arrays into One

Receive from the user two String arrays and create the arrays that join them

For example, if the two arrays are

"Bill Evans", "Keith Jarrett", "Chick Corea"
and
"Herbie Hancock", "Red Garland", "Wynton Kelley"

then the joined array has six elements:

"Bill Evans", "Keith Jarrett", "Chick Corea", "Herbie Hancock", "Red Garland", "Wynton Kelley"

Coding Idea

- Create a method that interacts with the user to receive the dimension of the array, receives the elements, and then returns the array thus generated
- Create a method that receives two String arrays and returns the array generated by connecting them
- Create an array to print the contents of a String array with position value

Connect.java the main method

```
import java.util.*;
public class Connect {
  public static void main( String[] args ) {
    System.out.println( "The first array." );
    String[] first = generateArray();
    System.out.println( "The second array." );
    String[] second = generateArray();
    System.out.println( "Joined." );
    String[] joined = connect( first, second );
    printWithIndex( joined );
  }
```

The first array: use the method `generateArray`

Connect.java the main method

```
import java.util.*;
public class Connect {
  public static void main( String[] args ) {
    System.out.println( "The first array." );
    String[] first = generateArray();
    System.out.println( "The second array." );
    String[] second = generateArray();
    System.out.println( "Joined." );
    String[] joined = connect( first, second );
    printWithIndex( joined );
  }
```

The second array: use the method `generateArray`

`Connect.java` the main method

```java
import java.util.*;
public class Connect {
  public static void main( String[] args ) {
    System.out.println( "The first array." );
    String[] first = generateArray();
    System.out.println( "The second array." );
    String[] second = generateArray();
    System.out.println( "Joined." );
    String[] joined = connect( first, second );
    printWithIndex( joined );
  }
```

The joined array: use the method `connect` by supplying the two arrays as parameters

`Connect.java` connect and print

```
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

The declaration: return type is `String[]`

`Connect.java` connect and print

```
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

Create an array of the dimension equal to the sum of the two dimensions

`Connect.java` connect and print

```
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

Copy the elements from the first array to the joined array at the same positions

Connect.java connect and print

```
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

Copy the elements from the second array to the joined array by moving the positions by the length of the first array

Connect.java connect and print

```java
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

Return the result

Connect.java connect and print

```java
  public static String[] connect( String[] pre, String[] post ) {
    String[] joined = new String[ pre.length + post.length ];
    for ( int i = 0; i <= pre.length - 1; i ++ ) {
      joined[ i ] = pre[ i ];
    }
    for ( int i = 0; i <= post.length - 1; i ++ ) {
      joined[ i + pre.length ] = post[ i ];
    }
    return joined;
  }
  public static void printWithIndex( String[] theArray ) {
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "%02d: %s%n", i, theArray[ i ] );
    }
  }
```

The print method. Print the elements of the array given as the parameter with the position values

Connect.java generate

```
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements: " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d: ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

The declaration: return type is `String[]`

Connect.java generate

```
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements: " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d: ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

Create a scanner

Connect.java generate

```
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements:  " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d:  ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

Receive dimension until the dimension is $>= 0$
The initial value must be set to a value for which the loop condition holds
Need to use "nextLine" to be consistent with the other actions

Connect.java generate

```
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements: " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d: ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

Create an array of the given dimension. Can be a 0-element array.

Connect.java generate

```
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements: " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d: ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

Receive elements

Connect.java generate

```java
  public static String[] generateArray() {
    Scanner console = new Scanner(  System.in );
    int size = -1;
    while ( size < 0 ) {
      System.out.print( "Enter # of elements: " );
      size = Integer.parseInt( console.nextLine() );
    }
    String[] theArray = new String[ size ];
    for ( int i = 0; i <= theArray.length - 1; i ++ ) {
      System.out.printf( "Enter elent %02d: ", i );
      theArray[ i ] = console.nextLine();
    }
    return theArray;
  }
}
```

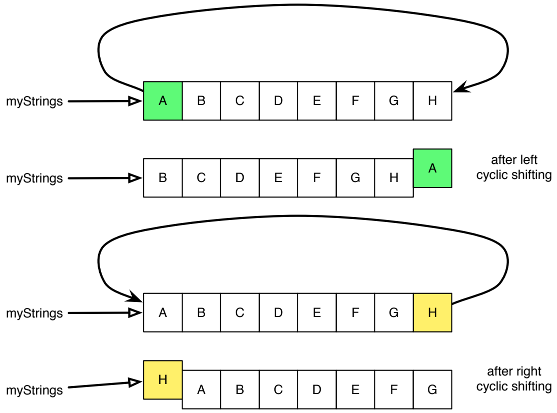
Return the array

Table of Contents

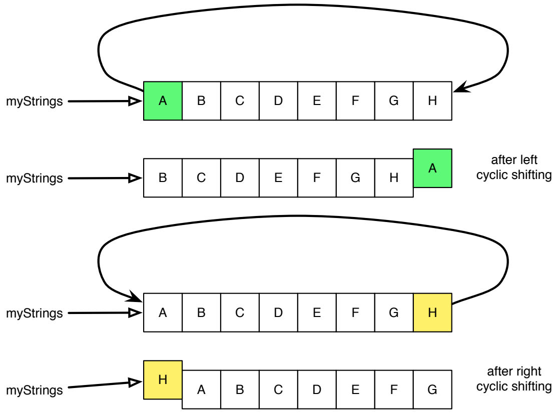
Cyclic Shifting Array Elements

- Left cyclic shift: shift elements by one lower position with element 0 moving to the last position
- Right cyclic shift: shift elements by one higher position with the last element moving to position 0

Cyclic Shifting Array Elements

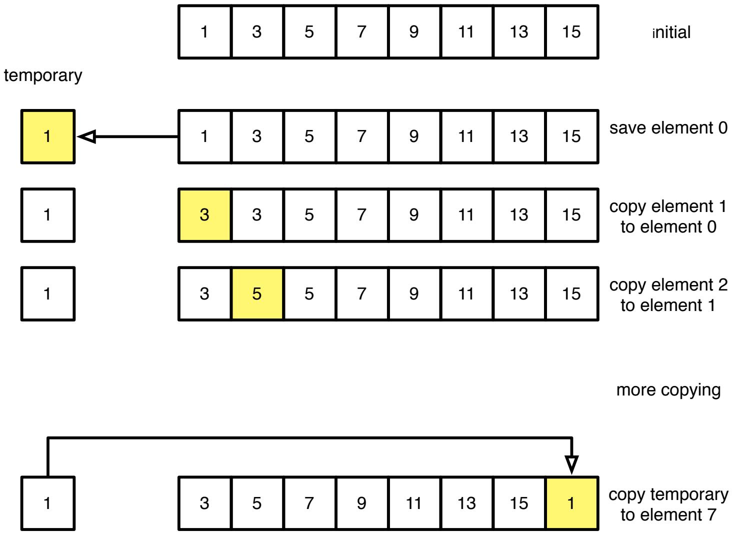


Cyclic Shifting Array Elements



- We can use the idea of temporary storage in this situation

Idea


1 3 5 7 9 11 13 15
initial
temporary
1
1 3 5 7 9 11 13 15
save element 0
1
3 3 5 7 9 11 13 15
copy element 1 to element 0
1
3 5 5 7 9 11 13 15
copy element 2 to element 1
more copying
1
3 5 7 9 11 13 15 1
copy temporary to element 7

Demonstrating Cyclic Shifts

- The program uses an array of non-negative integers
- The user enters the length of an array and the maximum for the elements of the array
- Thr program uses a `Random` object to generate elements of the array
- The user selects left or right cyclic shift and the program performs the requested change and prints the result

Demonstrating Cyclic Shifts

- The program uses an array of non-negative integers
- The user enters the length of an array and the maximum for the elements of the array
- Thr program uses a `Random` object to generate elements of the array
- The user selects left or right cyclic shift and the program performs the requested change and prints the result
- The program uses a special method for print a given array with a fixed number `WIDTH` of elements per row

ArrayCyclicShift.java

```java
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
           ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

WIDTH is the number of elements per line, defined as a constant

ArrayCyclicShift.java

```java
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
          ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

The method header for printing the elements of an `int` array; the parameter is `int[]`

ArrayCyclicShift.java

```java
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
          ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

Use a for loop for going through the indices, 0 .. `numbers.length - 1`

ArrayCyclicShift.java

```java
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
          ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

Simply print the element at `index`

ArrayCyclicShift.java

```
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
           ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

If `index` is NOT equal to `WIDTH - 1` module `WIDTH` and NOT equal to `LENGTH - 1`, print `",  "`

ArrayCyclicShift.java

```java
import java.util.*;
public class ArrayCyclicShift {

  public static final int WIDTH = 8;
  public static void printArray( int[] numbers ) {
    for ( int index = 0; index < numbers.length; index ++ ) {
      System.out.print( numbers[ index ] );
      if ( index != numbers.length - 1 &&
          ( index % WIDTH ) != WIDTH - 1 ) {
        System.out.print( ", " );
      }
      else {
        System.out.println();
      }
    }
  }
```

Otherwise, print the newline

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

The left cyclic shift method

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Announce that it is the left cyclic method

ArrayCyclicShift.java (part 2)

```
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Save `numbers[0]` to `temporary`

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Iterate index values from 1 to `numbers.length - 1` and copy element from position `index` to `index-1`

ArrayCyclicShift.java (part 2)

```
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Copy the saved element to position `numbers.length - 1`

ArrayCyclicShift.java (part 2)

```
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

The right cyclic shift method

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Announce that it is the right cyclic method

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Save `numbers[numbers.length - 1]` to `temporary`

ArrayCyclicShift.java (part 2)

```java
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Iterate index values from `numbers.length - 1` to 1 and copy element from position `index-1` to `index`

ArrayCyclicShift.java (part 2)

```
  public static void leftCyclicShift( int[] numbers ) {
    System.out.println( "Left Cyclic Shift" );
    int temporary = numbers[ 0 ];
    for ( int index = 1; index < numbers.length; index ++ ) {
      numbers[ index - 1 ] = numbers[ index ];
    }
    numbers[ numbers.length - 1 ] = temporary;
  }

  public static void rightCyclicShift( int[] numbers ) {
    System.out.println( "Right Cyclic Shift" );
    int temporary = numbers[ numbers.length - 1 ];
    for ( int index = numbers.length - 1; index >= 1; index -- ) {
      numbers[ index ] = numbers[ index - 1 ];
    }
    numbers[ 0 ] = temporary;
  }
```

Copy the saved element to `numbers[ 0 ]`

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Creating the console input

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Receive from the user the number of elements

The main method

```java
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

If the number of elements is invalid, throw a runtime error

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Create a `Random` object `rand`

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Receive from the user the maximum

The main method

```
public static void main( String[] args ) {
  Scanner console = new Scanner( System. in );
  System.out.print( "Enter array size > 0: " );
  int length = console.nextInt();
  if ( length <= 0 ) {
    throw new IllegalArgumentException( "Not positive" );
  }
  Random rand = new Random();
  System.out.print( "Enter maximum > 0: " );
  int maximum= console.nextInt();
  if ( maximum <= 0 ) {
    throw new IllegalArgumentException( "Not positive" );
  }
  int[] numbers = new int[ length ];
  for ( int index = 0; index < length; index ++ ) {
    numbers[ index ] = rand.nextInt( maximum );
  }
  printArray( numbers );
```

If the maximum is invalid, throw a runtime error

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Create an `int` array of the given length

The main method

```
  public static void main( String[] args ) {
    Scanner console = new Scanner( System. in );
    System.out.print( "Enter array size > 0: " );
    int length = console.nextInt();
    if ( length <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    Random rand = new Random();
    System.out.print( "Enter maximum > 0: " );
    int maximum= console.nextInt();
    if ( maximum <= 0 ) {
      throw new IllegalArgumentException( "Not positive" );
    }
    int[] numbers = new int[ length ];
    for ( int index = 0; index < length; index ++ ) {
      numbers[ index ] = rand.nextInt( maximum );
    }
    printArray( numbers );
```

Fill the array with random nonnegative integers less than the maximum

The main method

```
public static void main( String[] args ) {
  Scanner console = new Scanner( System. in );
  System.out.print( "Enter array size > 0: " );
  int length = console.nextInt();
  if ( length <= 0 ) {
    throw new IllegalArgumentException( "Not positive" );
  }
  Random rand = new Random();
  System.out.print( "Enter maximum > 0: " );
  int maximum= console.nextInt();
  if ( maximum <= 0 ) {
    throw new IllegalArgumentException( "Not positive" );
  }
  int[] numbers = new int[ length ];
  for ( int index = 0; index < length; index ++ ) {
    numbers[ index ] = rand.nextInt( maximum );
  }
  printArray( numbers );
```

Print the array

The main method (cont'd)

```
    String response = "";
    do {
      System.out.print( "Enter (l)eft (r)ight or (q)uit: " );
      response = console.next();
      if ( response.startsWith( "l" ) ) {
        leftCyclicShift( numbers );
        printArray( numbers );
      }
      else if ( response.startsWith( "r" ) ) {
        rightCyclicShift( numbers );
        printArray( numbers );
      }
    } while ( !response.startsWith( "q" ) );
  }
}
```

Use this string for storing user response

The main method (cont'd)

```
    String response = "";
    do {
      System.out.print( "Enter (l)eft (r)ight or (q)uit: " );
      response = console.next();
      if ( response.startsWith( "l" ) ) {
        leftCyclicShift( numbers );
        printArray( numbers );
      }
      else if ( response.startsWith( "r" ) ) {
        rightCyclicShift( numbers );
        printArray( numbers );
      }
    } while ( !response.startsWith( "q" ) );
  }
}
```

do-while loop that stops when the response starts with `"q"`

The main method (cont'd)

```
    String response = "";
    do {
      System.out.print( "Enter (l)eft (r)ight or (q)uit: " );
      response = console.next();
      if ( response.startsWith( "l" ) ) {
        leftCyclicShift( numbers );
        printArray( numbers );
      }
      else if ( response.startsWith( "r" ) ) {
        rightCyclicShift( numbers );
        printArray( numbers );
      }
    } while ( !response.startsWith( "q" ) );
  }
}
```

Prompt the user to receive input

The main method (cont’d)

```
    String response = "";
    do {
      System.out.print( "Enter (l)eft (r)ight or (q)uit: " );
      response = console.next();
      if ( response.startsWith( "l" ) ) {
        leftCyclicShift( numbers );
        printArray( numbers );
      }
      else if ( response.startsWith( "r" ) ) {
        rightCyclicShift( numbers );
        printArray( numbers );
      }
    } while ( !response.startsWith( "q" ) );
  }
}
```

If the response starts with `"l"` perform left cyclic shift and print the array

The main method (cont'd)

```
    String response = "";
    do {
      System.out.print( "Enter (l)eft (r)ight or (q)uit: " );
      response = console.next();
      if ( response.startsWith( "l" ) ) {
        leftCyclicShift( numbers );
        printArray( numbers );
      }
      else if ( response.startsWith( "r" ) ) {
        rightCyclicShift( numbers );
        printArray( numbers );
      }
    } while ( !response.startsWith( "q" ) );
  }
}
```

If the response starts with `"r"` perform right cyclic shift and print the array