Chapter 8
Arrays

Lesson page 8-1. Introduction to arrays

Activity 8-1-1 Introduction to arrays

Question 1. The base type of an array declared as below is <*type*>:

<*type*> `[]...[] b;`

Question 2. `T[]` is a class whose objects are arrays of elements of type `T`.

Question 3. The name of a newly created array of 30 elements whose base type is **long**.

Question 4. `String[] strArray; strArray= new String[40];`

Question 5. `String[] strArray= new String[5];`

Question 6. No. The elements of an array must have the same type.

Activity 8-1-2 The length of an array

Question 7. `temps.length` .

Activity 8-1-3 Referencing array elements

Question 8. A subscripted variable has the form

<*variable*> `[` <*expression*> `]`

where <*variable*> is the name of an array (or any expression that yields the name of an array) and <*expression*> has type **int** .

Question 9. In a reference `d[i]` to an element of array `d`, the value of expression `i` is called the index of that array element. The plural of index is "indexes" or "indices".

Question 10. Because the first element is numbered 0. For example, an array `g` of size `2` has elements numbered `0` and `1`, so the last element has index `1`, which equals `2-1`, which is `g.length-1`.

Question 11. `for (int i= 0; i != arr.length; i= i+1)`

Question 12. `System.out.println(arr[1]+arr[3]+arr[2]+arr[0]);`

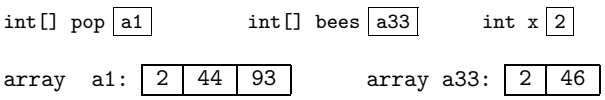
Question 13. The output is: `ace`. The two statement-comments are:

```
// store first 5 alphabet letters in array arr
// print every other position of arr, starting at 0
```

Activity 8-1-4 Array initializers

Question 14. Line 1: 'a' is a `char`, not a `String`.
Line 2: There is a period, not a comma, after `6`.
Line 3: The delimiters are () instead of {} and the third item is not a `String`.

Question 15. Here is a snapshot of the variables after execution:



Activity 8-1-5 Where can array initializers be used?

Question 16. An array initalizer `{...}` can be used only in a declaration-assignment, and no nowhere else.

Lesson page 8-2. Talking about array segments

Activity 8-2-1 The notation `b[i..j]`

Question 1. The elements in `b[1..4]`: `44, 2, 45, 2`.

Question 2. The subsection `b[1..5]` has 5 elements.

Question 3. `(j-1)+1-i = j-i`.

Question 4. `b[3..4]` or `b[3..b.length-2]`.

Activity 8-2-2 Picturing array elements

Question 5. `x[c..d]` and `x[c..e-1]`.

Question 6. The underlined expressions: `0`; `4`; `x.length`; `e-1`.

Question 7. The underlined parts: `d+1-c`; `e-3`; `e-2`; `e-c = 0` (or `e = c`); `d-c = -1` (or `d = c-1`).

Activity 8-2-3 Using pictures to present relations about arrays

Question 8. False; it has length 3.

Question 9. False; it is `b[m..m]`

Question 10. False: if `n = length-1` the second segment has 1 element.

Question 11. False; which is smallest depends on `b.length`. If `b.length=n`, the last segment is smallest because it is empty; otherwise, the middle segment is smallest.

Question 12. Write ".≤3" or just "≤3"

Question 13. Write ".>b[3..length-1]" or just ">b[3..]"

Lesson page 8-3. Some programs that use arrays

Activity 8-3-1 A schema for processing array segments

Question 1. (1) The size is 0. (2) The program segment is:

```
// inv: start <= i <= end+1; z[start..i-1] has been printed.
for (int i= start; i <= end; i++) {
    System.out.println(z[i]);
}
```

Activity 8-3-2 Processing array segments in reverse order

Question 2. The two schemata are:

```
// Process c[h..k-1], from end to beginning
    // invariant P: h-1 <= i < k and
    //              c[i+1..k-1] has been processed
    for (int i= k-1; i != h-1; i--) {
        Process c[i]
    }
```

```
// Process c[h..k], from end to beginning
    // invariant P: h-1 <= i <= k and
    //              c[i+1..k] has been processed
    for (int i= k; i != h-1; i--) {
        Process c[i]
    }
```

Question 3. The elements with even indices in z[0..i] have been printed.

Question 4. z[0]

Question 5. If z.length is even, z[z.length-2]; otherwise, z[z.length-1].

Question 6. If b.length is odd, the middle element belongs to the last half:

```
// Print the first half of z.
    // invariant: z[0..i-1] has been printed and
    //            0 <= i <= z.length/2
    for(int i= 0; i != z.length/2; i++) {
        System.out.println(z[i]);
    }
```

Question 7. If b.length is odd, the middle element belongs to the last half:

```
// Print the last half of z in reverse order.
    // invariant: z[i+1..length-1] has been printed and
    //            z.length/2-1 <= i <= z.length-1
    for(int i= z.length-1; z.length/2 <= i; i--) {
        System.out.println(z[i]);
    }
```

Activity 8-3-3 Printing input values in reverse order

Question 8. Here's the program:

```
// Read 5 integers and print their squares
// in reverse order.
import java.io.*;
public class ReverseSquares {
    public static void main(String[] gopher)
            throws Exception {

        int[] numbers= new int[5]; // the read ints

        // Link to the keyboard and prompt the user.
            BufferedReader br= new BufferedReader(
                new InputStreamReader(System.in));
            System.out.println("Enter 5 integers " +
                "and press return after each one:");

        // Read 5 ints from the keyboard and store them
        // in numbers in the order they are read in.
            // inv: 0 <= i <= length and
            //      numbers[0..i-1] has been processed
            for (int i= 0; i != numbers.length; i= i+1) {
                numbers[i]= (new Integer(
                    br.readLine())).intValue();
            }

        // Write the squares of number[0..],
        // in reverse order
            // inv: -1 <= i < length and
            //      numbers[i+1..length-1] has been
            //      processed
            for (int i= numbers.length-1; i>=0; i--) {
                System.out.println(
                    numbers[i]*numbers[i]);
            }
    }
}
```

Activity 8-3-4 Finding the number of smaller elements

Question 9. Here is the method:

```
// = number of elements of b that are less than v
public static int numberLess(int[] b, int v) {
    int c= 0;
    // c= (no. elements of b[0..b[length-1]] less than v)
    // inv: 0 <= i <= b.length and
    //      c = (no.elements of b[0..i-1] less than v)
    for (int i= 0; i != b.length; i++) {
        if (b[i] < v) { c= c+1; }
    }
    return c;
}
```

Question 10. If the `double`s represented $, a cheaper price would be good.

Activity 8-3-5 Testing function `numberLess`

Question 11. These are not the only errors, just a few of the tricky ones.

1. The initialization of the `String` array is missing braces ({}).
2. In the **for** loop, if `i = b.length` and the program tries to access `b[i]`, there will be an array-out-of-bounds exception.
3. Also in the **for** loop, what are the commas doing there?
4. In the conditions for the while loops, there is this mysterious "`i`". Where did it come from? What is its initial value? What is the value of `i` after printing the second line and does that value change before printing the third line? And the fourth?
5. The `charAt` positions are wrong for the third and fourth lines.

```
public class TestA {
    public static void main(String[] args) {
        String[] lines= {"Jack", "Sage", "Theo"};
        pp(lines);
    }

    // On the first line, print the first char of
    // each element of b
    // On the second line, print the second char of
    // each element of b
    // On the third line, print the third char of
    // each element of b
    // On the fourth line, print the fourth char of
```

```
    // each element of b
    public static void pp(String[] b) {
        int i;

        // output the first line
            for (i= 0; i != b.length; i= i+1) {
                System.out.print(b[i].charAt(0));
            }
            System.out.println();

        // output the second line
            i= 0;
            while (i != b.length) {
                System.out.print(b[i].charAt(1));
                i= i+1;
            }
            System.out.println();

        // output the third line
            i= 0;
            while (i<b.length) {
                System.out.print(b[i].charAt(2));
                i= i+1;
            }
        System.out.println();

        // output the fourth line
            i= -1;
            while (i<b.length-1) {
                i= i+1;
                System.out.print(b[i].charAt(3));
            }
    }
}
```

Question 12. There is no character `"Jo".charAt(2)`. The following program uses a simpler method `pp`.

```
public class TestB {
    public static void main(String[] args) {
        String[] lines= {"Mel", "Mary", "Jo"};
        pp(lines);
    }

    // On the first line, print the first char of each element
    // of b; on the second line, the second char; on the third
```

```
    // line, the third char, and on the fourth line; the fourth
    // char; print a blank for a nonexisting character
    public static void pp(String[] b) {
        for (int pos= 0; pos <= 3; pos++) {
            // Print line pos (the first line is line 0)
                for (int i= 0; i != b.length; i++) {
                    if (pos < b[i].length())
                        { System.out.print(
                            b[i].charAt(pos)); }
                    else { System.out.print(" "); }
                }
                System.out.println();
        }
    }
}
```

Activity 8-3-6 Checking for equality of arrays

Question 13. Here is the method:

```
    // = "arrays b and c are equal"
    public static boolean equals(Object[] a, Object[] b) {

        // return true if both refer to the same object
            if (a==b) return true;

        // return false if one is null or if
        // they are of different lengths
            if (a==null || b==null || a.length != b.length)
                return false;

        // return false if a[i]!=b[i] for some i
            for (int i=0; i!=a.length; i= i+1) {
                if (!(a[i].equals(b[i]))) return false;
            }

        // {a[0..a.length-1] = b[0..b.length-1]}
        return true;
    }
```

Activity 8-3-7 Returning an array

Question 14. The value of a variable that is an argument can not be changed by assignment to the corresponding parameter, because the "call by value" technique is used for parameter-argument correspondence. The VALUE of the argument is stored in the parameter.

Thus, whether the argument is a variable of a primitive type or a class type doesn't matter; it can't be changed by assignment to the corresponding parameter.

However, the object that is named by an argument CAN be changed by the method.

Lesson page 8-4. Arrays and classes

Activity 8-4-1 Class `Student`

Question 1. True. `getGrade` is not a static method. Only static methods can be called without creating an object of that class type.

Question 2. The name is an empty `String`.

Question 3. Here is the new constructor:

```
// Constructor: a student with name and grade
// supplied by the user
public Student() {
    read();
}
```

Activity 8-4-2 Class `StudentReport`

Question 4. Nothing would have to be changed.

Activity 8-4-3 Class `StudentReport` (continued)

Question 5. In class `StudentReport`:

```
// Read in the title of the report
public void getTitle() {
    System.out.print("Enter the title of the report: ");
    title= JLiveRead.readLineString();
}
```

In `main`, after declaring a new `StudentReport`: r.getTitle();
Add to the top of `printReport`:

```
System.out.println("\n" + title);
for (int i=0; i != title.length(); i++) {
      System.out.print("-");
}
System.out.println();
```

Activity 8-4-4 Private leaks

Question 6. `y` contains: 4, 2, 99, 1; `x` contains: 4, 2, 99, 1.

Activity 8-4-5 Using class DynamicArray

Question 7. Here is the program segment:

```
DynamicArray z= new DynamicArray();
for (int i=0; i != z.length(); i++) {
    z.set(i, i*i);
    System.out.print(z.get(i) + ", ");
}
```

Activity 8-4-6 Explaining class DynamicArray

Question 8. Here are the headers, with specifications:

```
// Constructor: an array with e (>0) elements allocated
public DynamicArray(int e)

// = the number of elements in array that are currently
// in use: 1+(maximum index into which a value was
// stored)
public int length()

// = the size of the allocated array
public int allocatedLength()

// = the element at index i (given 0 <= i < length())
public int get(int i)

// set the element at position i to v (0 <= i)
public void set(int i, int v)
```

Question 9. Here is the program segment:

```
DynamicArray z= new DynamicArray(20);
for (int i= 0; i != z.allocatedLength(); i++) {
    z.set(i, i*i);
    System.out.print(z.get(i) + ", ");
}
```

Question 10. Class DynamicArray allows the size of the array to change, so there is no need for a limit of 15 students.

Lesson page 8-5. Some basic array algorithms

Activity 8-5-1 Finding the first value

Question 1. Here is the new linearSearch.

```
public class Test {
    public static void main(String args[]) {

        String[] testArray= "a", "b", "a", "c", "a";
        System.out.println(Test.linearSearch(testArray,
            0, 4, "a"));
    }

    /* = the index of the first occurrence of x in
       b[h..k-1], or k if x is not in b[h..k-1].
       */
    public static int linearSearch(Object[] b, int h,
        int k, Object x) {

        int i= h;
        // invariant P: h<=i<=k and x not in b[h..i-1]
        while (i!=k && !x.equals(b[i])) {
            i= i+1;
        }
        return i;
    }
}
```

Activity 8-5-2 Another version of linear search

Question 2. Here is function `lastLinearSearch`:

```
// = index of last occurrence of x in array b,
// or -1 if x is not in b.
// Precondition: b and x have method equals.
public static int reverseLinSearch(Object[] b, Object x) {
    // inv: -1<=i<b.length and x not in b[i+1..b.length-1]
    for (int i= b.length-1; 0 <= i; i--) {
        if (b[i].equals(x)) { return i; }
    }
    return -1;
}
```

Question 3. Here is the filled-in method:

```
// Print the index of each occurrence of x in b and return
// the index of the first occurrence of x (-1 if none)
public static int linearSearch(Object[] b, Object x) {
    // location of first occurrence (-1 if none)
    int first= -1;
    // inv: 0<=i<=b.length and location of all
    //      x in b[h..i-1] has been printed, and
```

```
    //      first= first position of x in b[h..i-1]
    //      (-1 if none).
    for (int i= 0; i != b.length; i++) {
        if (x.equals(b[i])) {
            System.out.println(i);
            if (first < 0)
                { first= i; }
        }
    }
    return first;
}
```

Activity 8-5-3 Finding the minimum value

Question 4. Here is the function that returns the maximum value:

```
// = the maximum value in b.
public static int maxValue(int[] b) {
    int max= b[0];
    // {inv: max is the largest value in b[0..i-1]}
    for (int i= 1; i!=b.length; i++) {
        if (max < b[i]) { max= b[i]; }
    }
    return max;
}
```

Activity 8-5-4 Inserting into a sorted array segment

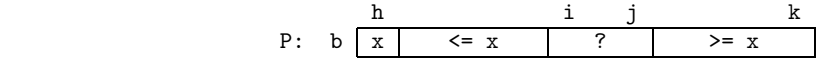
Question 5. Here is the rewritten method:

```
// b[h..k-1] is sorted in ascending order, and b[k]
// contains a value. Sort b[h..k]
public static void insertValue(String[] b, int h, int k) {
    String v= b[k];
    int i= k;
    /* inv: (1) Placing v in b[i] makes b[h..k] a
            permutation of its initial value
            (2) b[h..k] with b[i] removed is
            initial b[h..k-1]
            (3) v < b[i+1..k]
    */
    while ((i != h) && ( v.compareTo(b[i-1]) < 0 ) ) {
        b[i]= b[i-1];
        i= i-1;
    }
    b[i]= v;
}
```

Activity 8-5-5 Partitioning an array segment

Question 6. The possible values of `i` are 2 and 3.

Question 7. Here is the algorithm:




```
// Given Q true, truthify R1
    i= h+1;
    j= k;
    // {inv: b[h+1..i-1] <= b[h] = x <= b[j+1..k]}
    while (i<=j) {
         if (b[i] <= b[h]) { i= i+1; }
         else if (b[j] >= b[h]) { j= j-1; }
         else { // {b[j] < x < b[i]}
            int t1= b[i];  b[i]= b[j];  b[j]= t1;
            i= i+1; j= j-1;
        }
    }
}
```

Activity 8-5-6 Median of three

Question 8. The completed methods are:

```
// Swap b[i] and b[j].
public static void stwap(int[] b, int i, int j) {
    int temp= b[i];
    b[i]= b[j];
    b[j]= temp;
}

// Swap b[h], b[(h+k)/2], and b[k] (if necessary)
// so that b[h] <= b[(h+k)/2] <= b[k].
// Precondition: b[h..k] has at least three elements
public static void sort3(int[] b, int h, int k) {
    int mid= (h+k)/2;

    // Put the largest of b[h], b[mid], and b[k] in b[k]
        if (b[h] > b[k]) { swap(b, h, k); }
        if (b[mid] > b[k]) { swap(b, mid, k); }
    if (b[h] > b[mid]) { swap(b, h, mid); }
}
```

Activity 8-5-7 Merging two sorted array segments

Question 9. Here is the method, with the missing code no longer missing:

```
// = a sorted array containing the elements of x and y.
// Precondition: Arrays x and y are in ascending order.
public static int[] arrayMerge(int[] x, int[] y) {
    int[] merge= new int[x.length + y.length];
    int m= 0;
    int i= 0;
    int j= 0;

    // {inv: merge[0..m-1] contains x[0..i-1] and
    //       y[0..j-1], and merge[0..m-1]
    //       is in ascending order}
    while (i != x.length && j != y.length) {
        if (x[i] < y[j]) { merge[m]= x[i]; i= i+1; }
        else { merge[m]= y[j]; j= j+1; }
        m= m+1;
    }

    // copy into merge the end of whichever array
    // has yet to be completely copied
        while (i != x.length)
            { merge[m]= x[i]; i= i+1; m= m+1; }
        while (j != y.length)
            { merge[m]= y[j]; j= j+1; m= m+1; }

    return merge;
}
```

Activity 8-5-8 Binary search

Question 10. Here is binary search:

```
// Assume virtual elements b[-1] = -infinity and
// b.[b.length] = +infinity. Return an index
// i that satisfies R: b[i] <= x < b[i+1]
public static int binarySearch(int[] b, int x) {
    int i= -1;
    int j= b.length;
    // {P:b[i] <= x < b[j] and -1 <= i < j <= b.length}
    while (j != i+1) {
        int e= (i+j)/2;
        // {-1 <= i < e < j <= b.length}
        if (b[e] <= x) { i= e; }
```

```
            else { j= e; }
        }
        return i;
    }
```

Question 11.

i	j	mid
-1	5	2
2		3
3		4
4		

i	j	e
-1	5	2
	2	0
0		1
1		

i	j	e
-1	5	2
2		3
	3	

Lesson page 8-6. Selection sort and insertion sort

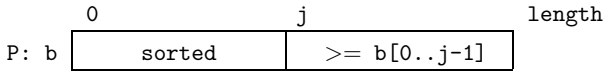
Question 1. False, sort of. Traditionally, "sorted" means in ascending order (e.g. the dictionary). It is correct to say "the array is sorted in descending order", but when you say "the array is sorted", most will assume you mean in ascending order.

Question 2. True.

Question 3. The array is sorted by the lengths of its elements.

Activity 8-6-1 Selection sort

Question 4. Precondition `Q` and postcondition `R` are fine. Invariant `P` is



The algorithm is:

```
int j= 0;
// {inv: P (see above)}
while (j != b.length) {
    int k= index of smallest value in b[j..];
    Swap b[k] and b[j];
    j= j+1;
}
```

Activity 8-6-2 Selection sort revisited

Question 5. Use a statement-comment to provide more structure to a program and to make it easier to read at different levels: read the statement-

comment to understand *what* is to be done and read its implementation to understand *how* the *what* is done.

Question 6. Here is the program segment:

```
// Store in p the index of the minimum of b[j..]
    int p= j;
    // {inv: b[p] is the minimum of b[j..i-1]}
    for (int i= j+1; j != b.length; i++) {
        if (b[i] < b[p]) {
            p= i;
        }
    }
```

Activity 8-6-3 Insertion sort

Question 7. Here is `insertionSort` and `insertValue`:

```
// Sort b into descending order
public static void insertionSortDescend(String[] b) {
    // {inv P: b[0..j-1] is in descending order}
    for (int j= 1; j < b.length; i= i+1) {
        insertValueDescend(b, 0, j);
    }
}

// Insert b[k] into b[h..k-1] in descending order
// Precondition: b[h..k-1] is in descending order.
public static void insertValueDescend(String[] b, int h, int k) {
    String v= b[k];
    int i= k;
    while ((i != h) && (v.compareTo(b[i-1]) > 0) ) {
        b[i]= b[i-1];
        i= i-1;
    }
    b[i]= v;
}
```