Introduction to Computer Science

PTI CS 103 Lecture Notes

The Prison Teaching Initiative

This version: April 4, 2019

Acknowledgements

TK.

Contents

1

Arrays

Consider this snippet of code:

```
if      (day ==  0) System.out.println("Monday");
else if (day ==  1) System.out.println("Tuesday");
else if (day ==  2) System.out.println("Wednesday");
else if (day ==  3) System.out.println("Thursday");
else if (day ==  4) System.out.println("Friday");
else if (day ==  5) System.out.println("Saturday");
else if (day ==  6) System.out.println("Sunday");
```

What does this code do? It prints the day of the week after conditioning on the value of an integer `day`. But this code is repetitive. It would be useful if we had some way of creating a list of days of the week, and then just specifying which of those days we wanted to print. Something like this:

```
System.out.println(DAYS\_OF\_WEEK[day]);
```

To achieve this in Java, we need arrays.

Definition 1.0.1. An *array* is an ordered and fixed-length list of values that are of the same type. We can access data in an array by *indexing*, which means referring to specific values in the array by number. If an array has `n` values, then we think of it as being numbered from `0` to `n-1`.

To *loop* or *iterate* over an array means that our program accesses every value in the array, typically in order. For example, if we looped over the array in the diagram, that would mean that we looked at the value at the 0th index, then the value at the 1st index, then the value at the 2nd index, and so on.

When we say that the array is "ordered" is that the relationship between an index and its stored value is unchanged (unless we explicitly modify it). If we loop over an unchanged array multiple times, we will always access the same values.

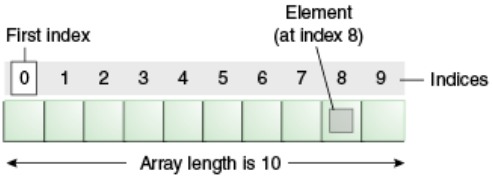

Figure 1.1: Diagram of an array (Credit: https://www.geeksforgeeks.org/arrays-in-java/)

Arrays are *fixed-length*, meaning that after we have created an array, we cannot change its length. We will see in the next chapter [TK: confirm] that `ArrayLists` are an array-like data structure that allows for changing lengths.

Finally, all the values in an array must be of the same type. For example, an array can hold all floating point numbers or all characters or all strings. But an array cannot hold values of different types.

1.1 Creating arrays

The syntax for creating an array in Java has three parts:

1. Array type
2. Array name
3. Either: array size or specific values

For example, this code creates an array of size `n = 10` and fills it with all `0.0`s

```
double[] arr;                       // Declare array
arr = new double[n];                // Initialize the array
for (int i = 0; i < n; i++) {       // Iterate over array
    arr[i] = 0.0;                   // Initialize elements to 0.0
}
```

The key steps are: we first declare and initialize the array. We then loop over the array to initialize specific values. We can also initialize the array at compile time, for example

```
String[] DAYS_OF_WEEK = {
//  Indices:
//  0      1      2      3      4      5      6
    "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"
};
```

Notice the difference in syntax. When creating an empty array, we must specify a size. When initialize an array at compile time with specific values, the size is implicit in the number of values provided.

Finally, in Java, it is acceptable to move the brackets to directly after the type declaration to directly after the name declaration. For example, these two declarations are equivalent:

```
int arr[];
int[] arr;
```

1.2 Indexing

Consider the array `DAYS_OF_WEEK` from the previous section. We can *index* the array using the following syntax:

```
System.out.println(DAYS_OF_WEEK[3]);  // Prints "Thu"
```

In Java, array's are said to use *zero-based indexing* because the first element in the array is accessed with the number `0` rather than `1`.

Example 1.2.1: What does `System.out.println(DAYS_OF_WEEK[1]);` print?

Example 1.2.2: What does this code do? What number does it print?

```
double sum = 0.0;
double[] arr = { 1, 2, 2, 3, 4, 7, 9 }
for (int i = 0; i < arr.length; i++) {
    sum += arr[i];
}
System.out.println(sum / arr.length);
```

1.3 Array length

As mentioned previously, arrays are *fixed-length*. After you have created an array, it's length is unchangeable. You can access the length of an array `arr[]` with the code `arr.length`.

Example 1.3.1: What does `System.out.println(DAYS_OF_WEEK.length);` print?

Example 1.3.2: Write a `for` loop to print the days of the week in order (Monday through Sunday) using an array rather than seven `System.out.println` function calls.

1.4 Default initialization

In Java, the default initial values for numeric primitive types is `0` and `false` for the `boolean` type.

Example 1.4.1: Consider this code from earlier:

```
double[] arr;
arr = new double[n];
for (int i = 0; i < n; i++) {
    arr[i] = 0.0;
}
```

Rewrite this code to be a single line.

1.5 Bounds checking

Consider this snippet of code.

Example 1.5.1: Where is the bug?

```
int[] arr = new int[100];
for (int i = 0; i <= 100; ++i) {
    System.out.println(arr[i]);
```

```
}
```

The issue is that the program attempts to access the value `arr[100]`, while the last element in the array is `arr[99]`.

This kind of bug is called an "off-by-one error" and is so common it has a name. In general, an off-by-one-error is one in which a loop iterates one time too many or too few.

Example 1.5.2: Where is the off-by-one-error?

```
int[] arr = new int[100];
for (int i = 0; i < array.length; i++) {
    arr[i] = i;
}
for (int i = 100; i > 0; --i) {
    System.out.println(arr[i]);
}
```

Example 1.5.3: Fill in the missing code in this `for` loop to print the numbers in reverse order, i.e. `5, 4, 3, 2, 1`:

```
int[] arr = { 1, 2, 3, 4, 5 };
for (???) {
    System.out.println(arr[i]);
}
```

1.6 Empty arrays

This code prints five values, one per line, but we never specified which values. What do you think it prints?

```
int[] arr = new int[5];
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

In Java, an unitialized or empty array is given a default value:

- For `int`, `short`, `byte`, or `long`, the default value is `0`.
- For `float` or `double`, the default value is `0.0`
- For `boolean` values, the default value is `false`.
- For `char`, the default value is the null character `'\0000'`.

Note that an array can be partially initialized.

Example 1.6.1: What does this code print?

```
char[] alphabet = new char[26];
alphabet[0] = 'a';
alphabet[1] = 'b';
```

```
for (int i = 0; i < alphabet.length; i++) {
    System.out.println(alphabet[i]);
}
```

1.7 Enhanced for loop

So far, we have seen how to iterate over arrays by indexing each element with a number:

```
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
for (int i = 0; i < vowels.length; ++ i) {
    System.out.println(vowels[i]);
}
```

We can perform the same iteration without using indices using an "enhanced `for` loop" or `for-each` loops:

```
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
for (char item: vowels) {
    System.out.println(item);
}
```

1.8 Exchanging and shuffling

Two common tasks when manipulating arrays are *exchanging two values* and *shuffling* values. (*Sorting* is more complicated and will be address later.)

To exchange to values, consider the following code:

```
double[] arr = { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 };
int i = 1;
int j = 4;
double tmp = arr[i];
arr[i] = arr[j];
arr[j] = tmp;
```

Example 1.8.1: What are the six values in the array, in order?

To shuffle the array, consider the following code:

```
int n = arr.length;
for (int i = 0; i < n; i++) {
    int r = i + (int) (Math.random() * (n-i));
    String tmp = arr[r];
    arr[r] = arr[i];
    arr[i] = tmp;
}
```

Example 1.8.2: What does this code do:

```
for (int i = 0; i < n/2; i++) {
    double tmp = arr[i];
    arr[i] = arr[n-1-i];
```

```
4       arr[n-i-1] = tmp;
5   }
```

Exercises

1.1 Write a program that reverses the order of values in an array.

1.2 What is wrong with this code snippet?

```
1   int[] arr;
2   for (int i = 0; i < 10; i++) {
3       arr[i] = i;
4   }
```

1.3 Rewrite this snippet using an enhanced `for-each` loop (for now, it is okay to re-define the array):

```
1   char[] vowels = {'a', 'e', 'i', 'o', 'u'};
2   for (int i = array.length; i >= 0; i--) {
3       char letter = vowels[i];
4       System.out.println(letter);
5   }
```

1.4 Write a program that uses `for` loops to print the following pattern:

```
1   1********
2
3   12*******
4
5   123******
6
7   1234*****
8
9   12345****
10
11  123456***
12
13  1234567**
14
15  12345678*
16
17  123456789
```

1.5 Write a program `HowMany.java` that takes an arbitrary number of command line arguments and prints how many there are.

References

Computer Science: An Interdisciplinary Approach, Robert Sedgewick and Kevin Wayne.

2

ArrayLists

A *collection* is a group of objects. Today, we'll be looking at a very useful collection, the `ArrayList`. A *list* is an ordered collection, and an `ArrayList` is one type of list.

Create a class `NameTracker` and follow along in it.

Before we can use an ArrayList, we have to import it:

```
import java.util.ArrayList;
```

Next, we call the constructor; but we have to declare the type of object the :code:‘ArrayList‘ is going to hold. This is how you create a new :code:‘ArrayList‘ holding :code:‘String‘ objects.

```
ArrayList<String> names = new ArrayList<String>();
```

Notice the word “String” in angle brackets. This is the Java syntax for constructing an ArrayList of String objects.

We can add a new String to names using the :code:‘add()‘ method.

```
names.add("Ana");
```

Example 2.0.1: Exercise: Write a program that asks the user for some names and then stores them in an ArrayList. Here is an example program:

```
Please give me some names:
Sam
Alecia
Trey
Enrique
Dave
Your name(s) are saved!
```

We can see how many objects are in our ArrayList using the size() method.

```
System.out.println(names.size()); // 5
```

Example 2.0.2: Modify your program to notify the user how many words they have added.

```
Please give me some names:
Mary
Judah
Your 2 name(s) are saved!
```

Remember how the `String.charAt()` method returns the `char` at a particular index? We can do the same with names. Just call `get()`:

```
names.add("Noah");
names.add("Jeremiah");
names.add("Ezekiel");
System.out.println(names.get(2)); // ``Ezekiel''
```

Example 2.0.3: Update your program to repeat the names back to the user in reverse order. Your solution should use a for loop and the size() method. For example:

```
Please give me some names:
Ying
Jordan

Your 2 name(s) are saved! They are:
Jordan
Ying
```

Finally, we can ask our names `ArrayList` whether or not it has a particular string.

```
names.add(``Veer'');
System.out.println(names.contains(``Veer'')); // true
```

Example 2.0.4: Update your program to check if a name was input by the user. For example:

```
Please give me some names:
Ying
Jordan

Search for a name:
Ying
Yes!
```

An `ArrayList` can hold any type of object! For example, here is a constructor for an `ArrayList` holding an instance of a `Person` class:

```
ArrayList<Person> people = new ArrayList<Person>();
```

where `Person` is defined as

```java
public class Person {

    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return this.name;
    }

    public int getAge() {
        return this.age;
    }
}
```

Example 2.0.5: Modify our program to save the user's input names as Person instances. Rather than storing String objects in the ArrayList, store Person objects by constructing them with the input name. You'll need to use the Person constructor to get a Person instance!

Exercises

2.1

Write a class BlueBook that tells the user the price of their car, depending on the make, model, and year. You should use Car.java and the stencil file provided, BlueBook.java.

Your program depends on what cars your BlueBook supports, but here is an example program:

```
What is your car's make?
Toyota
What is your Toyota's model?
Corolla
What is your Toyota Corolla's year?
1999

Your 1999 Toyota Corolla is worth \$2000.
```

2.2 Notify the user if the car is not in your BlueBook.

2.3 Clean up main by putting your code for creating the ArrayList in a separate method. What type should the method return?

2.4 If the car is not in the BlueBook, ask the user to input the relevant data, construct a new Car instance, add it to your ArrayList.

References

1. https://github.com/accesscode-2-1/unit-0/blob/master/lessons/week-3/2015-03-24_arraylists.md