
Arrays

CSE 142, Summer 2002
Computer Programming 1

<http://www.cs.washington.edu/education/courses/142/02su/>

Readings and References

- Reading
 - » Section 22.1, *An Introduction to Programming and Object Oriented Design using Java*, by Niño and Hosch
 - » Chapters 18 and 19, *Introduction to Programming in Java*, Dugan
- Other References
 - » Section *Arrays* of the Java tutorial
 - » <http://java.sun.com/docs/books/tutorial/java/data/arrays.html>

Arrays

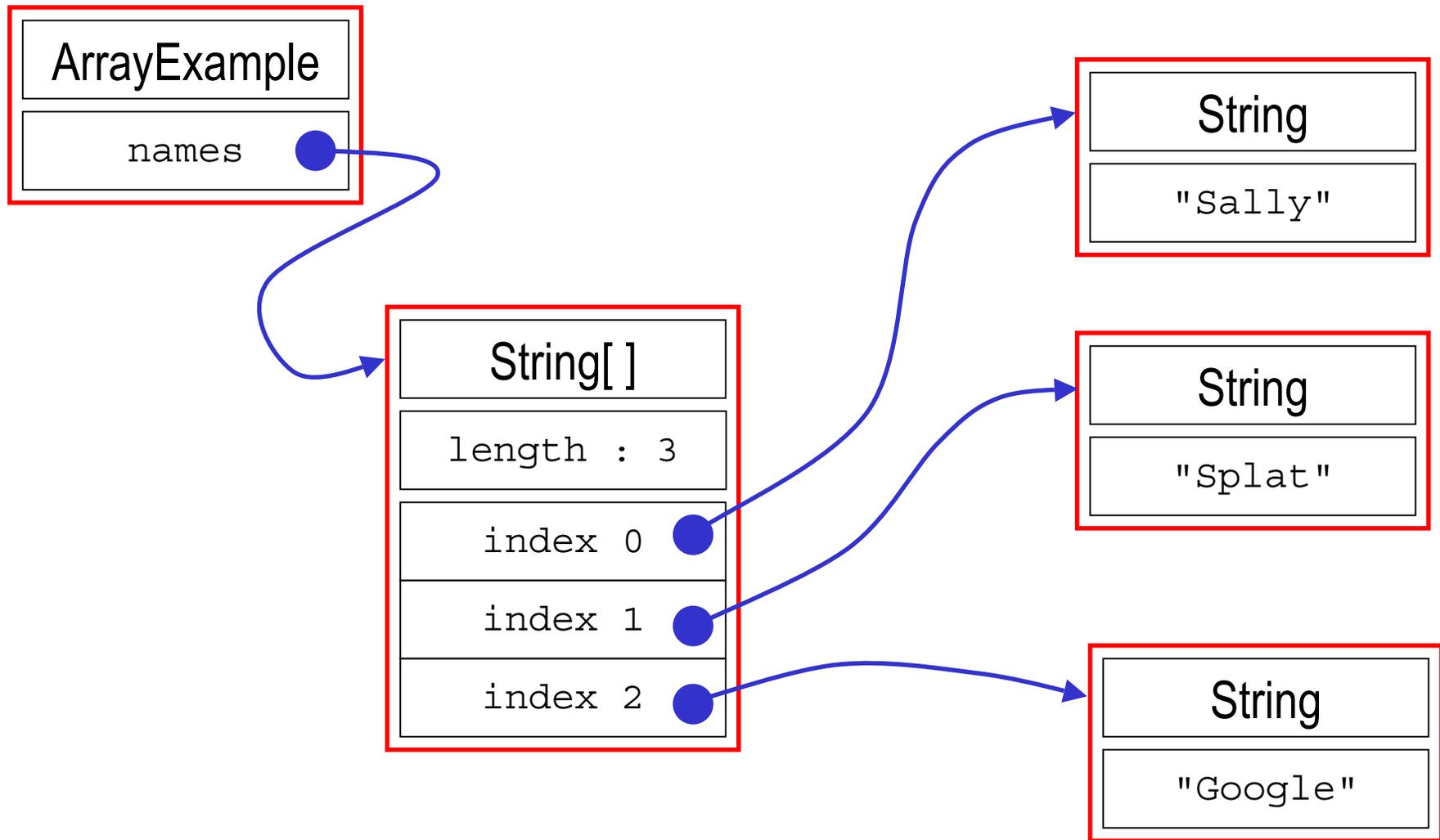
- Java (and many other languages) include *arrays* as the most basic kind of collection.
 - » Simple, ordered collections, similar to ArrayLists.
 - » Special syntax for declaring values of array type.
 - » Special syntax for accessing elements by position.
- Unlike ArrayLists:
 - » The size is fixed when the array is created.
 - » Can specify the type of the elements of arrays.

Array Example

```
public class ArraySample {
    public ArraySample() {
        names = new String[3];
        names[0] = "Sally";
        names[1] = "Splat";
        names[2] = "Google";
        for (int i=0; i<names.length; i++) {
            System.out.println("Name "+i+" is "+names[i]);
        }
    }

    String[] names;
}
```

Array Example



Java Array Object

- Arrays are objects! They...
 - » Must be instantiated with **new** unless immediately initialized
 - » Can contain **Object** references or primitive types
 - » Have class members (length, clone(),...)
 - » Have zero-based indexes
 - » Throw an exception if bounds are exceeded

Array Declaration and Creation

- Array have special type and syntax:

<element type>[] *<array name>* = **new** *<element type>* [*<length>*];

- Arrays can only hold elements of the specified type.
 - » Unlike ArrayList, element type can be int, double, etc.
 - » type can be Object, in which case very similar to ArrayList
- *<length>* is any positive integer expression
- Elements of newly created arrays are initialized
 - » but generally you should provide explicit initialization
- Arrays have an instance variable that stores the length
<array name>.length

Declaring and Allocating Arrays

- Declare an Array of ten **String** references

```
String[] myArray = new String[10];
```

- Declare an array and initialize elements

» the compiler counts the number of elements in this case

```
String[] myArray = { "Java", "is", "cool" };
```

- Declare, initialize, and use an array

» this is an "anonymous" array

```
boolean okay = doLimitCheck(x, new int[] {1, 100});
```

Array Element Access

- Access an array element using the array name and position: *<array name>* [*<position>*]
- Details:
 - » *<position>* is an integer expression.
 - » Positions count from zero, as with ArrayLists.
 - » Type of result is the element type of the array
- Can update an array element by assigning to it:
<array name> [*<position>*] = *<new element value>* ;
 - » Like ArrayList's set method

Looping Over Array Contents

- The length attribute makes looping over Array objects easy:

```
for (index=0; index<myArray.length; index++) {  
    System.out.println(myArray[index]);  
}
```

- The length attribute is a read-only value
 - » You can't change the size of the array after it has been created

Passing Array Objects to Methods

- You must declare that a method parameter is an Array:

```
public static void main(String[] args)
```

- Arrays are objects and so you are passing a reference when you call a method with an array
 - » This means array contents can be changed by methods
 - » This may be what you want, but if not, you need to make sure that other methods only get a copy of your array and the elements in it

Array Summary

- Arrays are the fundamental low-level collection type built in to the Java language.
 - » Also found in essentially all programming languages
- Size fixed when created
- Indexed access to elements
- Used to implement higher-level, richer container types
 - » ArrayList for example
 - » More convenient, less error-prone for users

The Arrays Class

- There is also a class called `java.util.Arrays`
 - » Note the capital A, this is a class name
 - » part of package `java.util`
 - » utility functions for using arrays
 - search
 - sort
 - initialize
 - » These are **static** methods so they exist and can be used without creating an object first

The Collections Class

- There is also a class called `java.util.Collections`
 - » utility functions for using classes that implement the `Collection` interface
 - » This class consists exclusively of static methods that operate on or return collections. It contains polymorphic algorithms that operate on collections, "wrappers", which return a new collection backed by a specified collection, and a few other odds and ends.
 - » These are **static** methods so they exist and can be used without creating an object first

Useful methods in Collections class

- static void sort(List list)
 - » Sorts the specified list into ascending order, according to the natural ordering of its elements.
 - » "natural order" is defined when you implement the interface Comparable
- static void sort(List list, Comparator c)
 - » Sorts the specified list according to the order induced by the specified comparator
 - » Comparator lets you define several different orders