Arrays

CSE 142, Summer 2003

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/03su/

Readings and References

- Reading
  - » Section 22.1, *Intro to Programming and Object-Oriented Design Using Java*, Niño and Hosch
- Other References
  - » "Arrays", Java tutorial
    - http://java.sun.com/docs/books/tutorial/java/data/arrays.html
  - » Popular Baby Names from Social Security Administration
    - http://www.ssa.gov/cgi-bin/babynames.cgi

Collections in the Real World

- Think about:
  - » words in a dictionary
  - » list of pets in your household
  - » deck of cards
  - » books in a library
  - » songs on a CD
- These things are all *collections*.
- Some collections are *ordered*, others are *unordered*

How can we manage lists of objects?

- We need a class that will let us ...
  - » add things to the list
  - » look at the elements of the list one by one
  - » find out how many things have been put in the list
  - » remove things from the list
  - » … among other things

PetSet example

- Think about PetSet in homework 2
  - » There were two animal objects in the distributed version of PetSet
  - » You designed a new type of animal, and then created at least one new object of this new type
  - » In order to manage the activities of the new animal you had to change the source code in PetSet
- Changing source code in order to implement variations in the data set is costly and inflexible

PetSet example

```
private Cat cat;
private Dog dog;

public void dine() {
    cat.eat(cat.getMealSize()*2);
    dog.eat(dog.getMealSize()*2);
}
```

- It would be nice if we could somehow keep track of the objects in a more general way

Arrays

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

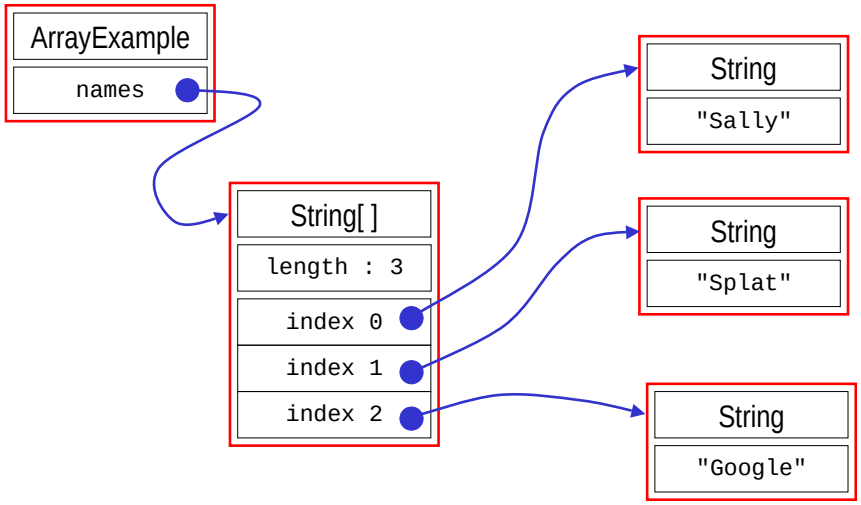
Array Example

```
public class ArraySample {

    String[] names;

    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]);
        }
    }
}
```

Array Example



Java Array Object

- Arrays are objects! They...
  - » Must be instantiated with **`new`** unless immediately initialized
  - » Can contain primitive types or references to objects
  - » 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
  - » element type can be a reference type (ie, objects)
  - » element type can be int, double, etc.
- *<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
  - » 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>* ;

```
names[1] = "Splat";
```

```
return names[idx];
```


NameList example

```
public class NameList {

private String[] theBook = {
    "Jacob","Michael","Joshua","Matthew","Emily",
    "Ethan","Joseph","Andrew","Christopher","Madison",
    "Daniel","Nicholas","William","Anthony","Hannah",
    "David","Tyler","Alexander","Ryan","John"};

private String[] names;

public NameList(int count) {
    names = new String[count];
    int span = theBook.length;
    for (int i=0; i<names.length; i++) {
        names[i] = theBook[(int)(Math.random()*span)];
    }
  }
...
```


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

- Method parameters can be Arrays:

  `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

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