

*Introduction to Programming II*  
*Arrays in Java*

Chris Brooks

Department of Computer Science  
University of San Francisco

## 8-2: Arrays and Collections

---

- Many times in a program, you will need to have a collection of objects.
  - A list of students
  - A list of test scores
  - A set of data from multiple experiments
  - A collection of shapes to draw
  - etc
- Java has a wide variety of different collection classes.
- We'll start with arrays, and then ArrayLists.
  - Later on, Lists, then Trees and Hashtables.

## 8-3: Arrays

---

- Java provides built-in support for *arrays*
  - An array is a set of objects that are sequentially arranged in memory.
  - Size of the array is declared when it is created.

- Example:

```
int []atBats;  
int runs[]
```

- This declares the array, but not its contents.

## 8-4: Arrays

---

- We need to use 'new' to actually allocate memory for the array.

```
int x = sc.nextInt();  
atBats = new int[10];  
runs = new int[x];
```

- This means that the reference to the array will be allocated on the stack, while the array itself will be allocated on the heap.

## 8-5: A digression: Memory layout

---

- Programs typically use two types of memory:
  - Statically allocated memory
    - These are variables whose type and size can be identified at compile time.
    - Usually primitive types.
    - We say that this memory is allocated “on the stack”
  - Dynamically allocated memory
    - These are variables whose size is determined at run time.
    - Created with `new`
    - We say that this memory is allocated “on the heap.”

## 8-6: Indexing an Array

---

- One nice feature of arrays is that they can be indexed using the `[]` operator.
  - For example, `myArray[4]`.
  - Indices start at 0.
  - This is sometimes called *random access*; we can jump anywhere in the array we want.

## 8-7: Indexing an Array

---

- One of the most common things to do with an array is loop over it and do something with all the elements.

```
public double computeAverage(int scores[]) {
    double ave = 0;
    for (int i = 0; i < scores.length; i++) {
        ave = ave + scores[i];
    }
    ave = ave / scores.length;
    return ave;
}
```

- Notice that arrays have a length data member - this is different from length() with strings.

## 8-8: Initializing Arrays

---

- If you know the elements in your array ahead of time, you can initialize them like this:

```
String daysOfWeek[] =  
{ ``Monday`` , ``Tuesday`` , ``Wednesday`` ,  
  ``Thursday`` , ``Friday`` , ``Saturday`` , ``Sunday`` };
```

- This is convenient when you have a sequence of values to use as constants. (days, months, colors, grades, etc)

## 8-9: Arrays of Objects

---

- We can also make arrays of objects.

```
Student studentArray[] = new Student[10];  
for (int i = 0; i < 10; i++) {  
    studentArray[i] = new Student();  
}
```

## 8-10: Method calls with arrays

---

- The method signature contains the array name and brackets - this indicates that an array is being passed in.
  - `double average(int scores[])`
- When calling a method, just pass the name of the array, as if it was a regular variable.
  - `result = average(myscores)`
- You can also send an element of an array into a method, as long as the parameter is of the appropriate type.
  - `int square(int value)`
  - `square(myscores[5])`

## 8-11: In-class exercise, pt 1

---

- Let's build a Student class
  - Give it three instance variables: name, ID, GPA and an appropriate constructor.
- Create a program that:
  - Prompts the user for a number of students:
  - Allocates an array of Students.
  - For each student, asks the user their name, ID, GPA.
  - Uses the Student constructor to fill in that element of the array.

## 8-12: In-class exercise, pt 2

---

- Now modify your program to:
  - Compute and display the average GPA for all students.
  - Print out a list of all students and their IDs.

## 8-13: In-class exercise, pt 3

---

- Now, modify your program so that the user can input a minimum and maximum GPA.
- Print out the name and ID of all students whose GPA is between the minimum and maximum.

## 8-14: Advantages and disadvantages of arrays

---

- Advantages:
  - All memory is contiguous
  - Can 'jump' directly to any element of the array.
- Disadvantages:
  - Hard to resize or add elements.

## 8-15: Summary

---

- Arrays let you create a sequential list of objects.
- Need to declare array, and then contents with new.
- Can iterate over array or access any element using the index.