Arrays of Objects

GEEN163

“Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration.”

Stan Kelly-Bootle

British author,
singer-songwriter
and computer scientist

Course Evaluations

- Course evaluations are available on Blackboard
- Be sure to complete all evaluations for all classes
- Only 15% of the students in GEEN163 have completed the evaluation

Programming Project

- This week's program should be done by teams of two students
- There are three classes, each with several small methods
- The program involves a GUI with graphics

Loops

- When you are dealing with the elements of an array, you almost always use a loop

```
//  Add together all values of the array bear
double[] bear = new double[5000];
double sum= 0.0
for (int deer = 0; deer < bear.length; deer++) {
        sum += bear[ deer ];
}
```

Arrays as Objects

- An array is an object in Java
- You can call a method on the array
- Arrays have an integer class variable length that contains the length of the array

```
double[] cobra = new double[10000];
int boa = cobra.length;          // boa = 10000
```

Half Empty or Half Full

- An array might not contain as many useful values as its full capacity

```
Widget[] eagle = new Widget[100];
for (int egg = 0; egg < 47; egg++) {
        eagle[egg] = new Widget();
}
```

- The array eagle has 100 elements, but only 47 values have been used
- eagle[60].getCount() will cause a nullPointerException

Creating Objects from Files

- A program might declare an array of objects in the beginning and then create the object later
- Consider a program that reads a data file and creates an object for each line of data

Example Class

- Consider a class that holds data about classrooms

```
public class Classroom {
        private String   building;
        private int      roomNum;
        private int      capacity;
}
```

Classroom Constructor

```
public class Classroom {
        private String   building;
        private int      roomNum;
        private int      capacity;
        public Classroom( String bldg, int num, int cap ) {
                building    = bldg;
                roomNum = num;
                capacity    = cap;
        }
}
```

Getter Methods

```
public String getBuilding() {
        return building;
}
public int getRoomNum() {
        return roomNum;
}
public int getCapacity() {
        return capacity;
}
```

Building an Array of Objects

```
java.io.File frog = new java.io.File("mydata.txt");
java.util.Scanner inFile = new java.util.Scanner(frog);
Classroom[] space = new Classroom[500];
int numRooms = 0;
while (inFile.hasNext()) {
        String bldg = inFile.next();
        int room    = inFile.nextInt();
        int cap     = inFile.nextInt();
        space[numRooms] = new Classroom( bldg, room, cap);
        numRooms++;
}
```

Search Example

- Assume we have an array of Classroom objects called space with numRooms values in the array
- We want to display all rooms with capacity greater than 150

Which **if** statement will select object's whose capacity is greater than 150?

- Assume there is a for loop with i as the index

A. if (capacity[i] > 150)
B. if (space.getCapacity(i) > 150)
C. if (space[i].getCapacity() > 150)
D. if (space[i].capacity > 150)
E. if (space[150] > capacity)

What is wrong with this program?

```
for (int i = 0; i < space.length; i++) {
   if (space[i].getCapacity() > 150) {
      System.out.println(space[i].getBuilding() +
          "  " + space[i].getRoomNum());
   }
}
```

A. It should be **<=** space.length

B. space.length should be numRooms

C. You cannot call methods inside println

D. Nothing

Stringing Methods Together

- Sometimes a program will need to put many parts together

```
if (space[i].getBuilding().equals("Graham"))
```

- Java interprets this from left to right
  - Get the i^{th} element of space (a Classroom object)
  - Get the building name from the object (a String)
  - Compare the name to “Graham” (a boolean)

What is displayed?

```
String  fish = "Computer Science";
System.out.print( fish.toUpper().substring(3,6) );
```

A. put
B. mpu
C. PUT
D. PUTE
E. MPU

Histogram

- A histogram gives a count of events per unit time
- Consider reading a file and counting the number of accidents for each hour
- Assume the file contains:

  hour minute fatalities

- You want to show the percentage of accidents that occur in each hour

Counting Accidents per Hour

```
java.io.File frog = new java.io.File("accidents.txt");
java.util.Scanner inFile = new java.util.Scanner(frog);
int[] perHour = new int[24];
int total = 0;
while (inFile.hasNext()) {
        int hour= inFile.nextInt();    // read hour of accident
        inFile.next();                 // skip minutes
        inFile.next();                 // skip fatalities
        perHour[ hour ]++;             // count accidents
        total++;
}
```

Write with your Team

- Now that perHour contains the number of accidents at that hour and total contains the total number of accidents, display the percentage of the total for each hour
- The output should look like:

0:00	15.7
1:00	11.2
2:00	9.756

etc.

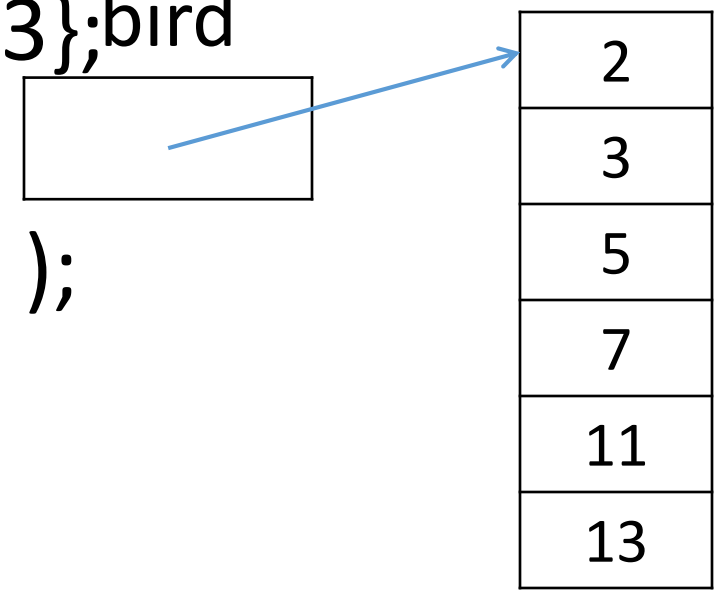
Possible Solution

```
for (int i = 0; i < 24; i++ ) {
        double percent = 100.0 * (double)perHour[ i ] / total;
        System.out.println( i +":00  " + percent )
}
```

Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

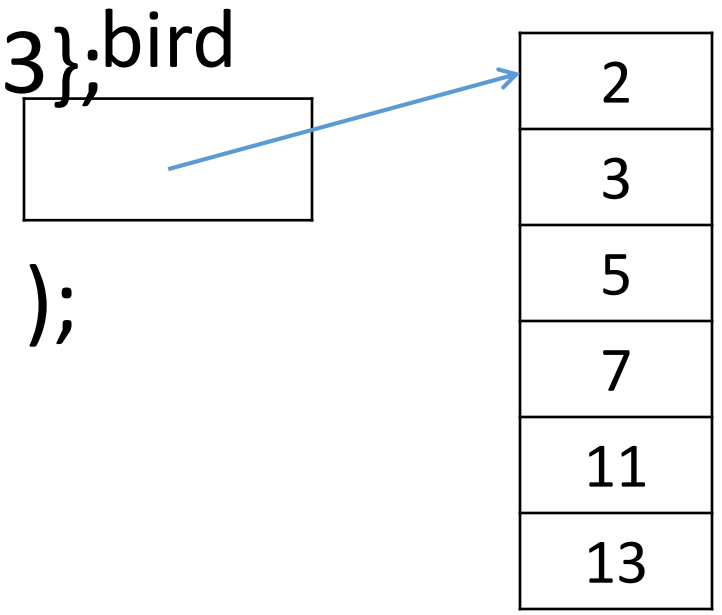
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
        . . .
void aMethod(int[] fish) {
   fish[2] = 47;
}
```



Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

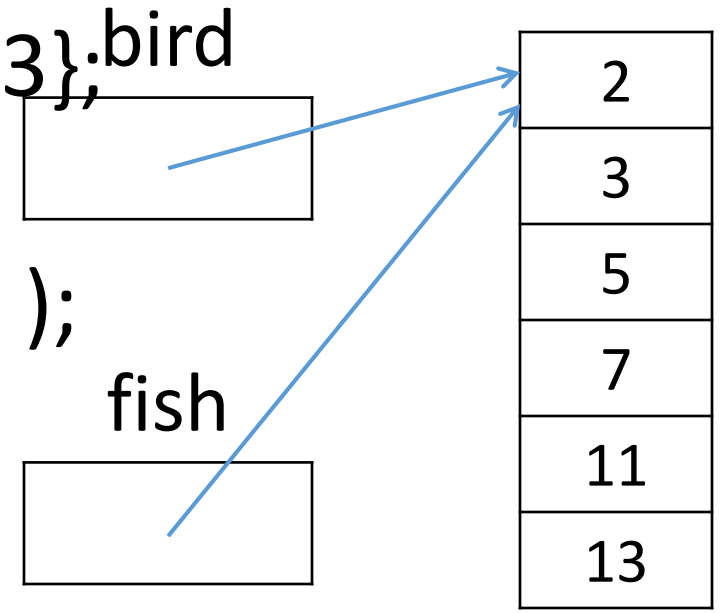
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

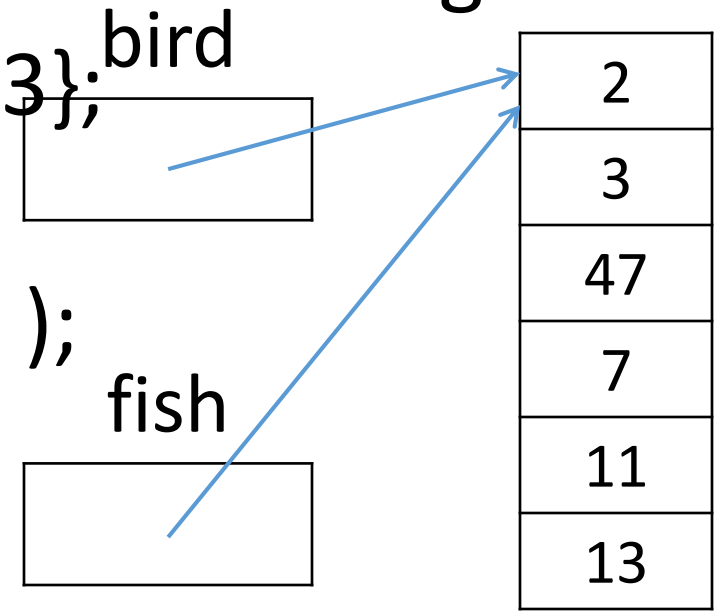
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

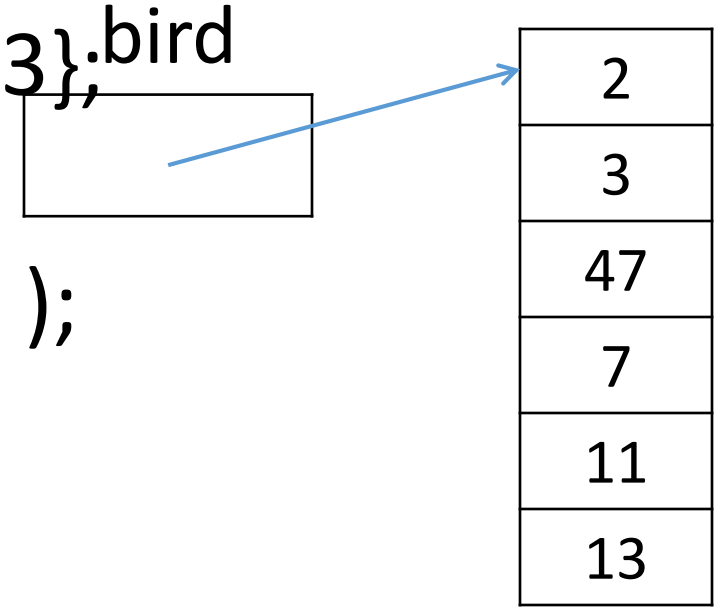
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Arrays in Methods

- Consider the problem to count the number of negative numbers in an array of doubles
- The header of the methods would be

```
int  countNegative( double[] dog )
```

- The method would require a loop, probably a for loop
- Inside the loop there would be an if statement to check if the number is less than zero
- If less than zero, the method would increment a counter

Write with your Team

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog )
```

Possible Solution

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog ){
    int count = 0;
    for (double cat : dog) {
        if (cat < 0) {
            count++;
        }
    }
    return count;
}
```

Another Possible Solution

- Write a method to count the number of negative values in an array of doubles

```
int  countNegative( double[] dog ){
    int count = 0;
    for (int cat=0; cat <dog; cat++) {
        if (dog[cat] < 0) {
            count++;
        }
    }
    return count;
}
```

Putting a Positive Spin on It

- Consider a modification to our method that makes all of the negative number a positive number of the same magnitude

Will this work?

- Make all the numbers positive

```
int  makePositive( double[] dog ){
    int count = 0;
    for (int cat=0; cat <dog; cat++) {
        if (dog[cat] < 0) {
            dog[cat] = -dog[cat];
        }
    }
    return count;
}
```

A. No – you cannot change the array
B. No – you can't put a dash before dog
C. All the above
D. Yes

Course Evaluations

- Course evaluations are available on Blackboard
- Be sure to complete all evaluations for all classes
- Only 15% of the students in GEEN163 have completed the evaluation

Programming Project

- This week's program should be done by teams of two students
- There are three classes, each with several small methods
- The program involves a GUI with graphics