Arrays II

Lecture 22, Wed Mar 10 2010

borrowing from slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/111-10
News

- Assignment 2 out
  - due Fri Mar 26
  - start now, it's challenging!
Recap: Array Declaration and Types

- Just like ordinary variable, must
  - declare array before we use it
  - give array a type
- Since `cansSold` contains integers, make integer array:
  ```java
  int[] cansSold = new int[10]
  ```
- Looks like variable declaration, except:
  - empty brackets on the left tell Java that `cansSold` is an array...
  - the number in the brackets on the right tell Java that array should have room for 10 elements when it's created
Recap: Array Declaration and Types

```java
public class ArrayTest1 {
    public static void main(String[] args) {
        final int ARRAYSIZE = 10;
        int[] cansSold = new int[ARRAYSIZE];

        cansSold[0] = 185;
        cansSold[1] = 92;
        cansSold[2] = 370;
        cansSold[3] = 485;
        cansSold[5] = 128;
        cansSold[6] = 84;
        cansSold[7] = 151;
        cansSold[8] = 32;
        cansSold[9] = 563;

        // do useful stuff here
        System.out.println("Element 4 is " + cansSold[4]);
    }
}
```
Recap: Array Declaration and Types

```java
public class ArrayTest2 {
    public static void main(String[] args) {
        int[] cansSold = {185, 92, 370, 485, 209, 128, 84, 151, 32, 563};

        // do useful stuff here
        System.out.println("Element 4 is "+ cansSold[4]);
    }
}
```

- Can also use `initializer list`
- Right side of declaration does not include type or size
  - Java figures out size by itself
- Types of values on right must match type declared on left
- `Initializer list` may only be used when array is first declared
## Arrays With Non-Primitive Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>201.25</td>
</tr>
<tr>
<td>1</td>
<td>100.50</td>
</tr>
<tr>
<td>2</td>
<td>412.75</td>
</tr>
<tr>
<td>3</td>
<td>555.25</td>
</tr>
<tr>
<td>4</td>
<td>195.00</td>
</tr>
<tr>
<td>5</td>
<td>160.00</td>
</tr>
<tr>
<td>6</td>
<td>105.00</td>
</tr>
<tr>
<td>7</td>
<td>188.75</td>
</tr>
<tr>
<td>8</td>
<td>40.00</td>
</tr>
<tr>
<td>9</td>
<td>703.75</td>
</tr>
</tbody>
</table>

- Great if you're always storing primitives like integers or floating point numbers
- What if we want to store String types too?
- Remember that String is an object, not a primitive data type
Arrays With Non-Primitive Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>185</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>92</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>370</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>485</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>209</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>151</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>563</td>
<td>9</td>
</tr>
</tbody>
</table>

- Then we create **array of objects**
  - In this case objects will be Strings
- Array won't hold actual object
  - holds references: pointers to objects

```java
String[] location = new String[10];
```
Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
<td>0</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
<td>1</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
<td>2</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
<td>3</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
<td>4</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
<td>5</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
<td>6</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
<td>7</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
<td>8</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
<td>9</td>
</tr>
</tbody>
</table>

- Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
```
Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
<td>0 &quot;Chan Centre&quot;</td>
</tr>
</tbody>
</table>
| 1 92    | 1 100.50| 1 "Law School"
| 2 370   | 2 412.75|
| 3 485   | 3 555.25|
| 4 209   | 4 195.00|
| 5 128   | 5 160.00|
| 6 84    | 6 105.00|
| 7 151   | 7 188.75|
| 8 32    | 8 40.00  |
| 9 563   | 9 703.75 |

- Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
location[1] = "Law School";
```
# Arrays of Objects

Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
location[1] = "Law School";
location[2] = "Main Library";
```
Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

Now we can put references to Strings in our String array.

```java
    location[0] = "Chan Centre";
    location[1] = "Law School";
    location[2] = "Main Library";

    ...and so on...
```
Arrays of Objects

Or we could have done this:

```java
String[] location = {
    "Chan Centre", "Law School", "Main Library", ....
};
```
## Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

- Each individual String object in array of course has all String methods available
- For example, what would this return?

```java
location[2].length()
```
### Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

- Each individual String object in array of course has all String methods available
- For example, what would this return?

```java
location[2].length()
```

- 12
Arrays of Objects

location
cashIn
cansSold

Think about a cleaner way to do all this…
Arrays and Object Design

- Multiple array approach not very object-oriented
  - can create arrays of objects
  - can create objects of our own design...

<table>
<thead>
<tr>
<th>location</th>
<th>cashIn</th>
<th>cansSold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
<td></td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
<td></td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
<td></td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
<td></td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
<td></td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
<td></td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
<td></td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
<td></td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
<td></td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
<td></td>
</tr>
</tbody>
</table>

- "Chan Centre"
- "Law School"
- "Main Library"
- "Koerner Library"
- "Business"
- "Biology"
- "Education"
- "Applied Science"
- "Agriculture"
- "Computer Science"
Arrays and Object Design

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double
Cokematic

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double

```java
public class Cokematic {
    private int numberOfCans;
    private String location;
    private int cansSold;
    private double cashIn;

    public Cokematic(int cans, String loc, int sold, double cash) {
        numberOfCans = cans;
        location = loc;
        cansSold = sold;
        cashIn = cash;
        System.out.println("Adding machine");
    }
}
```
Cokematic

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double

```java
public void buyCoke()
{
  if (numberOfCans > 0)
  {
    numberOfCans = numberOfCans - 1;
    cansSold = cansSold + 1;
    cashIn = cashIn + 1.25;
    System.out.println("Have a Coke");
    System.out.println(numberOfCans + " remaining");
  }
  else
  {
    System.out.println("Sold out.");
  }
}
```
public String getLocation()
{
    return location;
}

public int getCansSold()
{
    return cansSold;
}

public double getCashIn()
{
    return cashIn;
}

public void reloadMachine(int newCans)
{
    numberOfCans = numberOfCans + newCans;
    System.out.println("reloading machine");
}

public int getNumberOfCans()
{
    return numberOfCans;
}

public String toString()
{
    return (location + " sold: " + cansSold + " left: " + numberOfCans + " made: " + cashIn);
}

Cokematic

- In driver, executing

```java
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```
Cokematic

- In driver, executing

```java
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```

- Results in

```
Cokematic

numberOfCans: 100
location: "Chan Centre"
cansSold: 185
cashIn: 201.25

buyCoke()
getLocation()
getCansSold()
getCashIn()
reloadMachine()
toString()
```
Cokematic

- In driver, executing

```java
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```

- Results in

![UML Diagram]

- Note: leaving out methods in UML diagrams from now on to fit on page
Cokematic

- In driver, executing

Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);

- Results in

Note: leaving out methods in UML diagrams from now on to fit on page
CokeEmpire

Contains array of Cokematic objects

```java
public class CokeEmpire {
    private Cokematic[] collection; // what does this do?

    public CokeEmpire() {
        collection = new Cokematic[10]; // what does this do?
    }

    public void addCokematic(int index, int cans, String loc, int sold, double cash) {
        collection[index] = new Cokematic(cans, loc, sold, cash);
    }

    public Cokematic getCokematic(int index) {
        return collection[index];
    }
}
```
CokeEmpire

- In driver, executing:

```java
CokeEmpire myMachines = new CokeEmpire();
```
CokeEmpire

- In driver, executing

```java
CokeEmpire myMachines = new CokeEmpire();
```

- results in

![Diagram showing a list of 10 elements labeled 0 to 9]
CokeEmpire

- Populate array with Cokematic objects
Populate array with Cokematic objects

```java
myMachines.addCokematic(0, 100, "Chan Centre", 185, 201.25);
```
CokeEmpire

- Populate array with Cokematic objects

```java
myMachines.addCokematic(1, 150, "Law School", 92, 100.50);
```

Cokematic
- `numberOfCans`: 100
- `location`: "Chan Centre"
- `cansSold`: 185
- `cashIn`: 201.25

Cokematic
- `numberOfCans`: 150
- `location`: "Law School"
- `cansSold`: 92
- `cashIn`: 100.50
CokeEmpire

 Populate array with Cokematic objects

myMachines.addCokematic(2, 200, "Main Library", 370, 412.75);
CokeEmpire

What does this return?

myMachines.getCokematic(1).getCansSold()
What does this return?

```java
myMachines.getCokematic(1).getCansSold()
```
CokeEmpire

What does this return?

```java
myMachines[getCokematic(1)].getCansSold()
```

```
Cokematic
numberOfCans: 100
location: "Chan Centre"
cansSold: 185
cashIn: 201.25

Cokematic
numberOfCans: 150
location: "Law School"
cansSold: 92
cashIn: 100.50

Cokematic
numberOfCans: 200
location: "Main Library"
cansSold: 370
cashIn: 412.75
```
CokeEmpire

What does this return?

```
myMachines.getCokematic(1).getCansSold()
```
Arrays of Arrays
Arrays of Arrays

- In any given array, all data must be of same type
Arrays of Arrays

- In any given array, all data must be of same type
- All arrays in array of arrays must be of same type
Arrays of Arrays

- In any given array, all data must be of same type
- All arrays in array of arrays must be of same type
- So easier to use a two-dimensional array!
## Two-Dimensional Arrays

- In Java, 2D array implemented internally as array of arrays
- but externally syntax of 2D array may seem easier to use

<table>
<thead>
<tr>
<th>rows</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

**columns**

0     1     2

0     0     0
0     1     2
0     2     4
0     3     6
Two-Dimensional Arrays

- In Java, 2D array implemented internally as array of arrays
  - but externally syntax of 2D array may seem easier to use
- Typical control structure for computing with 2D array is nested loop
  - loop within another loop
- Let's write program to
  - load array with values shown
  - print contents of array

```
0     1     2
0     0     0
0     1     2
0     2     4
0     3     6
```
Two-Dimensional Arrays

```
public class ArrayTest5 {
    public static void main(String[] args) {
        // Code goes here
    }
}
```
Two-Dimensional Arrays

public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];
        // Table values
        multTable[0][0] = 0; multTable[0][1] = 0; multTable[0][2] = 0;
        multTable[1][0] = 0; multTable[1][1] = 1; multTable[1][2] = 2;
        multTable[2][0] = 0; multTable[2][1] = 2; multTable[2][2] = 4;
        multTable[3][0] = 0; multTable[3][1] = 3; multTable[3][2] = 6;
    }
}

public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }
    }
}
Two-Dimensional Arrays

```
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }
    }
}
```
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                System.out.print(multTable[row][col] + " ");
            }
            System.out.println();
        }
    }
}
Two-Dimensional Arrays

public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                System.out.print(multTable[row][col] + " ");
            }
        }
    }
}

{ | 0 | 1 | 2 |
  0 | 0 | 0 | 0 |
  1 | 0 | 1 | 2 |
  2 | 0 | 2 | 4 |
  3 | 0 | 3 | 6 |
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                System.out.print(multTable[row][col] + " ");
            }
            System.out.println();
        }
    }
}
```