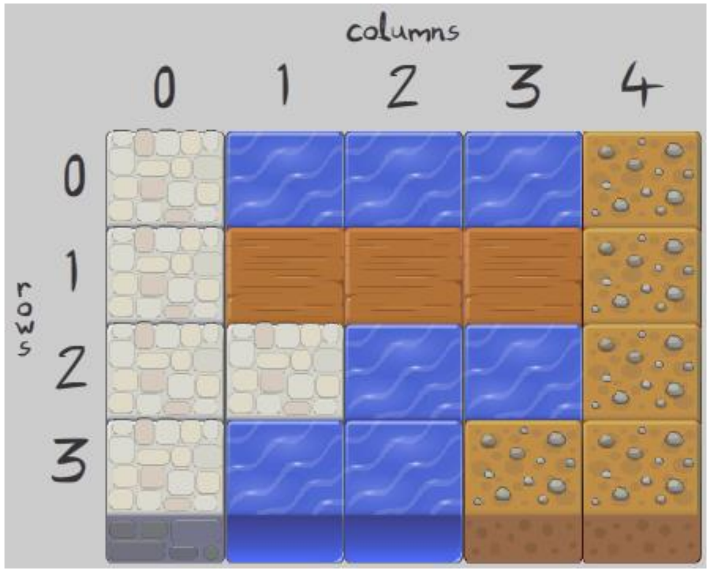
MULTI-DIMENSIONAL ARRAYS



Fundamentals of Computer Science I

Outline

- Arrays Revisited
- Two-Dimensional Arrays
- Multidimensional Arrays
- Ragged Arrays

Arrays revisited

- Arrays
  - Store a bunch of values under one name
  - Declare and create in one line:

```java
int N = 8;
int [] x = new int[10];
double [] speeds = new double[100];
String [] names = new String[N];
```

  - To get at values, use name and index between []:

```java
int sumFirst2 = x[0] + x[1];
speeds[99] = speeds[98] * 1.1;
System.out.println(names[0]);
```

  - Array indexes start at 0!

Arrays revisited

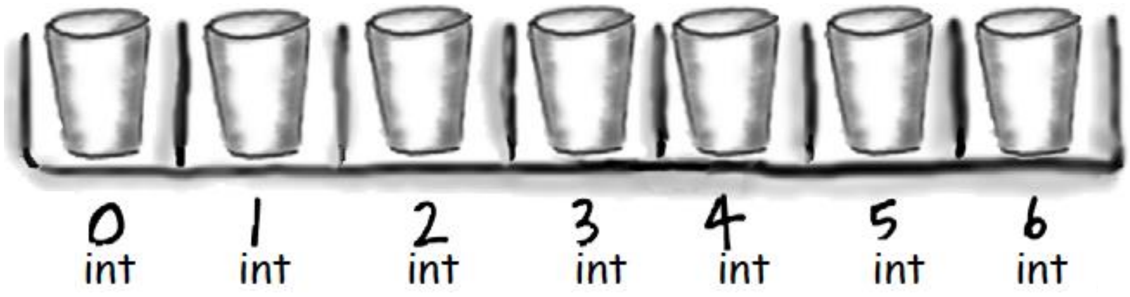
- Arrays
  - You can just declare an array:

```
int [] x;
```

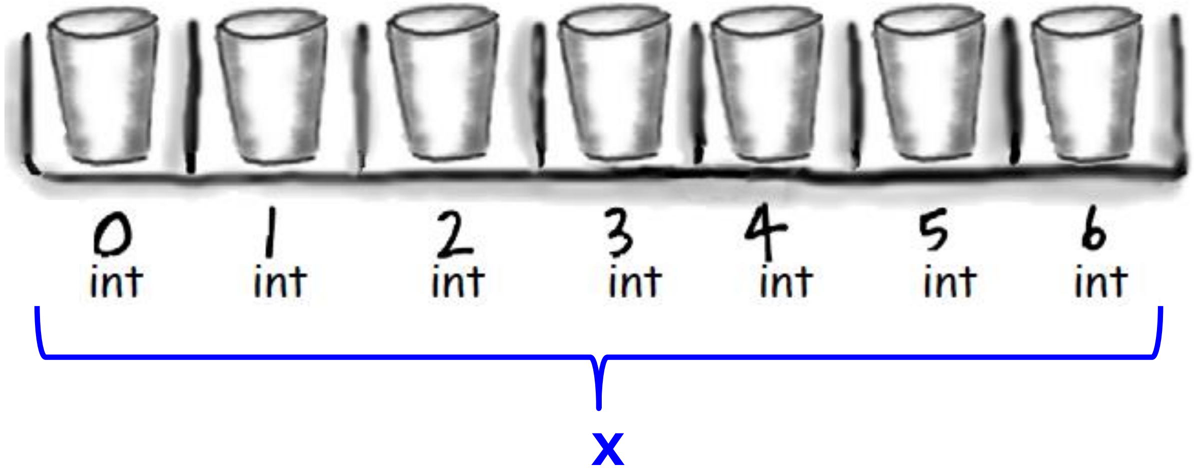
  - But x is not very useful until you "new" it:

```
int [] x;
x = new int[7];
```

  - `new` creates the memory for the slots
    - Each slot holds an independent `int` value
    - Each slot initialized to default value for type

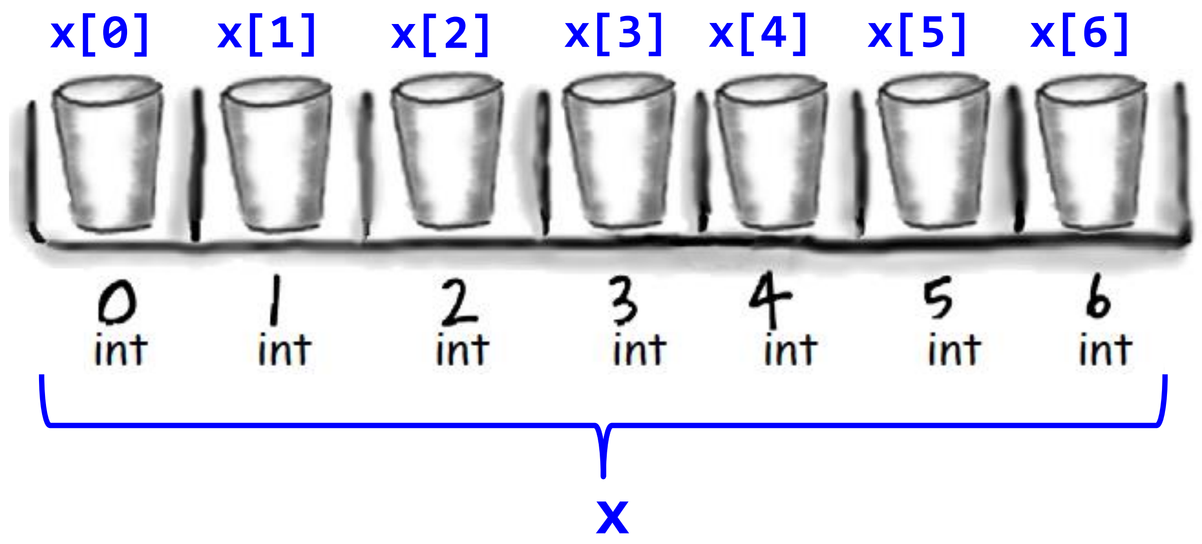


Arrays revisited



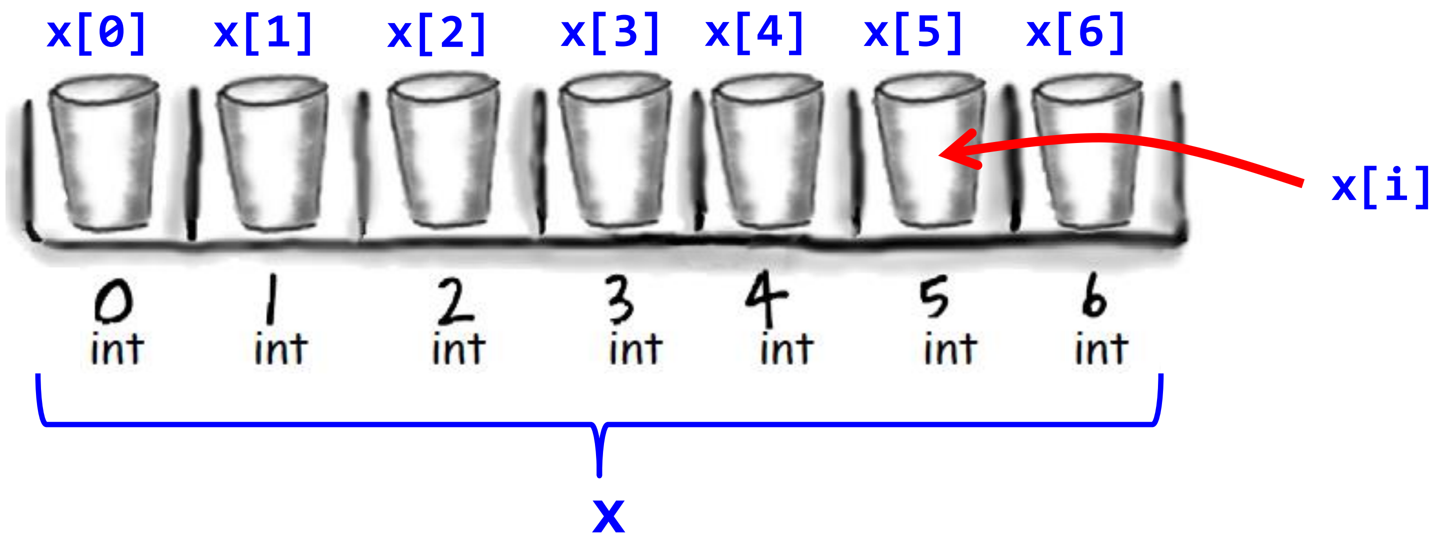
- Variable x refers to the whole set of slots
- You can't use the variable x by itself for much
- Except for finding out the number of slots: `x.length`

Arrays revisited



- x[0],x[1],…,x[6] refers to value at a particular slot
- x[-1] or x[7] = ArrayIndexOutOfBoundsException

Arrays revisited

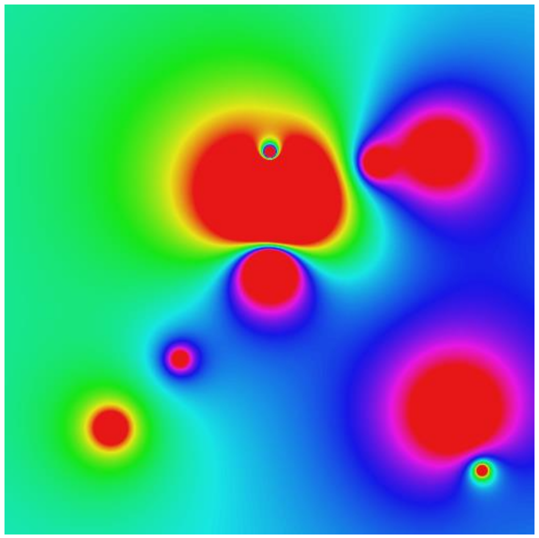


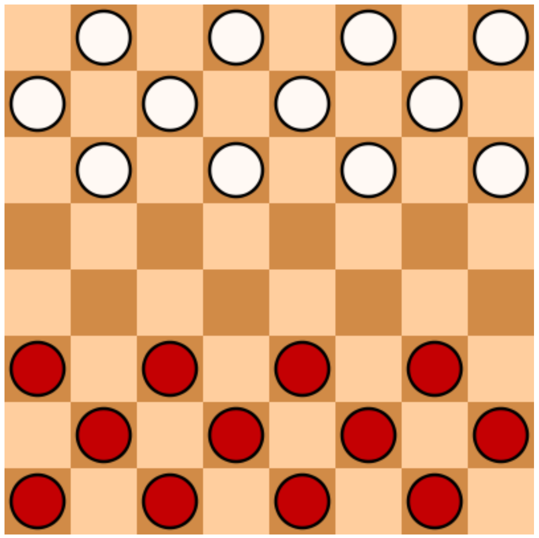
- `x[i]` refers to the value at a slot, but the slot index is determined by variable `i`
  - If i = 0 then `x[0]`, if i = 1 then `x[1]`, etc.
- Whatever inside [] must be an int
- Whatever inside [] must be in 0 to `x.length - 1` (inclusive)

Two dimensional array examples

- Two dimensional arrays
  - Tables of hourly temps for last week
  - Table of colors for each pixel of a 2D image
  - Table storing piece at each position on a checkerboard

0h	1h	…	23h
32.5	30.0		45.6
…			
59.5	62.1	…	60.0
60.7	61.8	…	70.5
62.6	62.0	…	68.0





Weather data

- Goal: Read in hourly temp data for last week
  - Each row is a day of the week
  - Each column is a particular hour of the day

01:53

20:53

10/24/11

10/29/11

45.0	48.0	48.9	48.9	48.0	46.0	45.0	46.9	45.0	48.2	[illegible]	[illegible]	[illegible]	[illegible]	59.0	57.9	57.9	57.2	54.0	50.0	48.9	46.9	44.6	45.0
44.1	43.0	43.0	43.0	39.9	37.9	37.4	39.0	39.0	39.0	39.0	37.9	39.2	41.0	41.0	41.0	39.0	37.9	36.0	35.6	33.8	32.0	32.0	30.2
30.2	28.0	27.0	23.0	23.0	23.0	19.9	19.0	19.0	23.0	30.9	33.1	34.0	37.0	35.6	36.0	32.0	32.0	32.0	27.0	27.0	25.0	21.9	23.0
21.9	21.0	21.0	21.0	19.4	17.6	17.6	17.6	19.4	19.0	21.0	26.1	34.0	37.4	39.0	41.0	41.0	39.0	37.0	37.0	37.0	34.0	35.1	34.0
33.8	32.0	37.0	30.9	32.0	34.0	33.1	30.9	32.0	35.1	39.0	41.0	39.9	42.1	43.0	43.0	42.1	39.9	36.0	33.1	27.0	25.0	23.0	19.9
19.9	19.0	18.0	16.0	16.0	15.1	14.0	14.0	15.1	21.0	[illegible]	[illegible]	[illegible]	[illegible]	52.0	50.0	51.1	50.0	46.0	48.9	44.1	44.1	39.9	39.2
46.0	46.0	45.0	44.6	44.1	44.1	44.1	44.1	42.1	42.1	42.8	44.1	45.0	46.9	46.0	44.1	44.1	42.8	39.0	37.0	35.1	35.1	30.9	30.0

Two dimensional arrays

- Declaring and creating
  - Like 1D, but another pair of brackets:

```
final int DAYS  = 7;
final int HOURS = 24;
double [][] a = new double[DAYS][HOURS];
```

- Accessing elements
  - To specify element at the i^{th} row and j^{th} column:

```
a[i][j]
```

a[0][0]	a[0][1]	a[0][2]	...	a[0][22]	a[0][23]
a[1][0]	a[1][1]	a[1][2]	…	a[1][22]	a[1][23]
…	…	…	…	….	…
a[6][0]	a[6][1]	a[6][2]	…	a[6][22]	a[6][23]

Temperature on second day of data, last hour of day

Reading temperature data

- Initialize all elements of our 2D array
  - Nested loop reading in each value from `keyboard`
  - Find weekly max and min temp

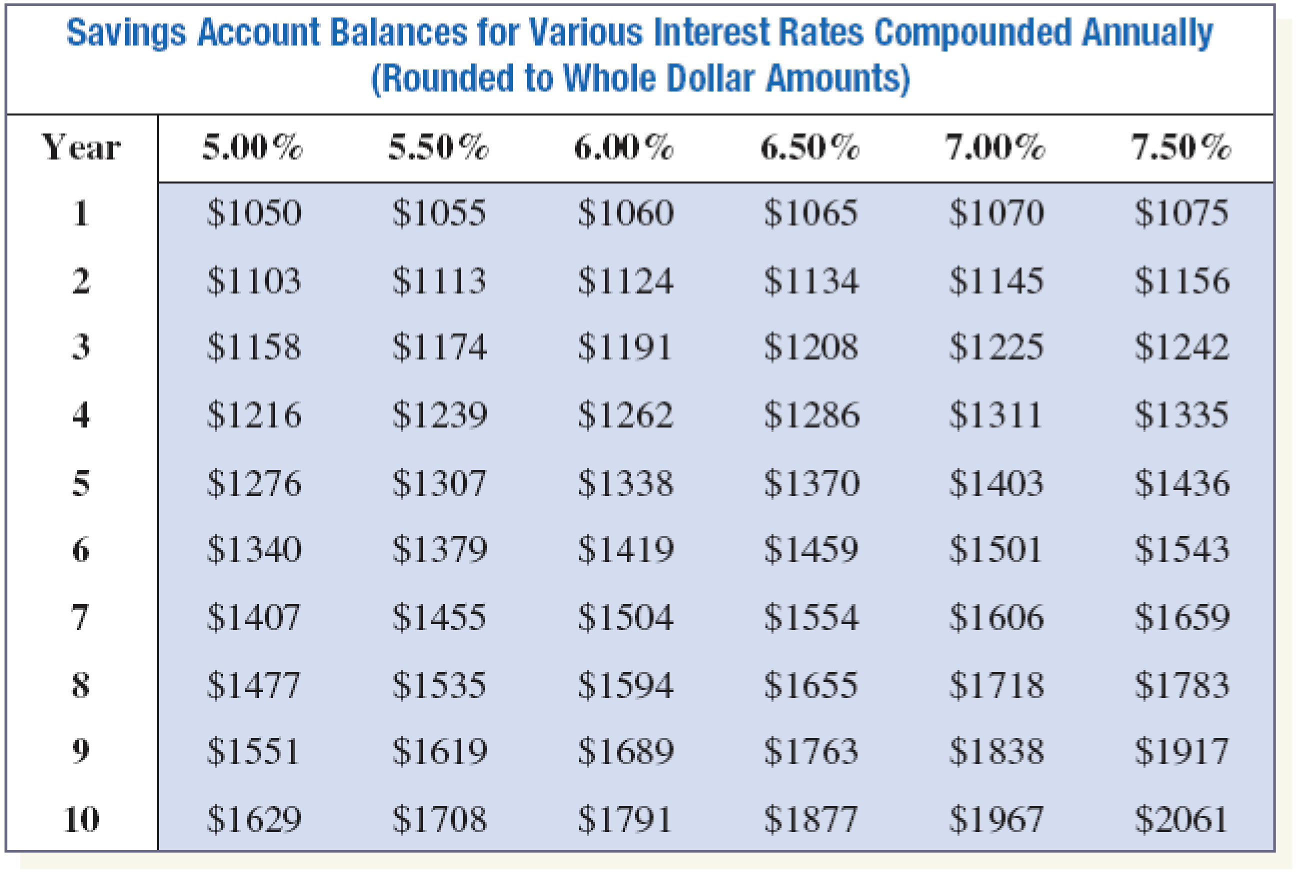
```
final int DAYS  = 7;
final int HOURS = 24;
double [][] a = new double[DAYS][HOURS];
double min = Double.POSTIVE_INFINITY;
double max = Double.NEGATIVE_INFINITY;

for (int row = 0; row < DAYS; row++)
{
   for (int col = 0; col < HOURS; col++)
   {
      a[row][col] = keyboard.nextDouble();
      min = Math.min(min, a[row][col]);
      max = Math.max(max, a[row][col]);
   }
}
System.out.println("min = " + min + ", max = " + max);
```

Start the min at a really high temp.

Start the max at a really low temp.

The new min temp is either the current min or the new data point.

Another Example



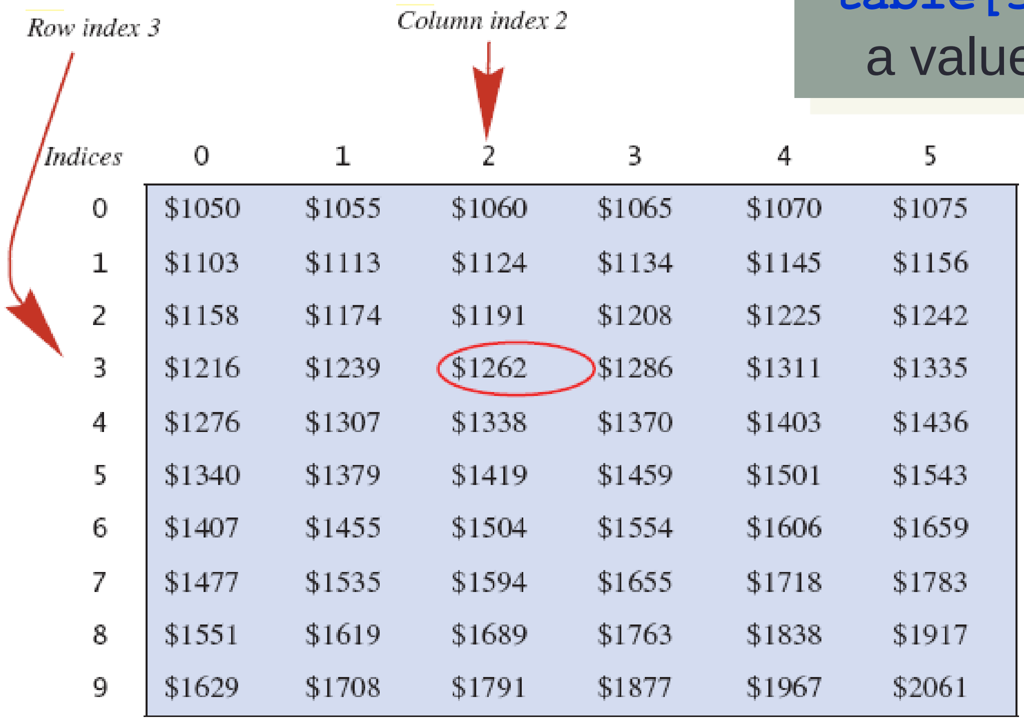
Savings Account Balances for Various Interest Rates Compounded Annually (Rounded to Whole Dollar Amounts)

Year	5.00%	5.50%	6.00%	6.50%	7.00%	7.50%
1	$1050	$1055	$1060	$1065	$1070	$1075
2	$1103	$1113	$1124	$1134	$1145	$1156
3	$1158	$1174	$1191	$1208	$1225	$1242
4	$1216	$1239	$1262	$1286	$1311	$1335
5	$1276	$1307	$1338	$1370	$1403	$1436
6	$1340	$1379	$1419	$1459	$1501	$1543
7	$1407	$1455	$1504	$1554	$1606	$1659
8	$1477	$1535	$1594	$1655	$1718	$1783
9	$1551	$1619	$1689	$1763	$1838	$1917
10	$1629	$1708	$1791	$1877	$1967	$2061

Multidimensional-Array Basics

- Figure 7.7 Row and column indices for an array named **`table`**

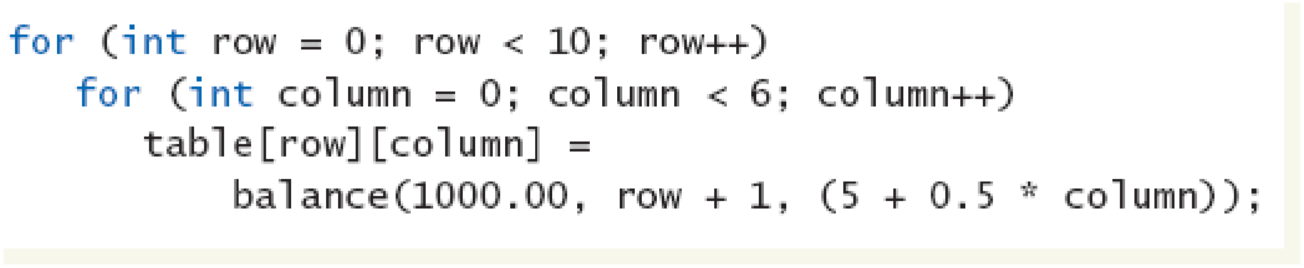
`table[3][2]` has a value of 1262


Row index 3
Column index 2

Indices	0	1	2	3	4	5
0	$1050	$1055	$1060	$1065	$1070	$1075
1	$1103	$1113	$1124	$1134	$1145	$1156
2	$1158	$1174	$1191	$1208	$1225	$1242
3	$1216	$1239	$1262	$1286	$1311	$1335
4	$1276	$1307	$1338	$1370	$1403	$1436
5	$1340	$1379	$1419	$1459	$1501	$1543
6	$1407	$1455	$1504	$1554	$1606	$1659
7	$1477	$1535	$1594	$1655	$1718	$1783
8	$1551	$1619	$1689	$1763	$1838	$1917
9	$1629	$1708	$1791	$1877	$1967	$2061

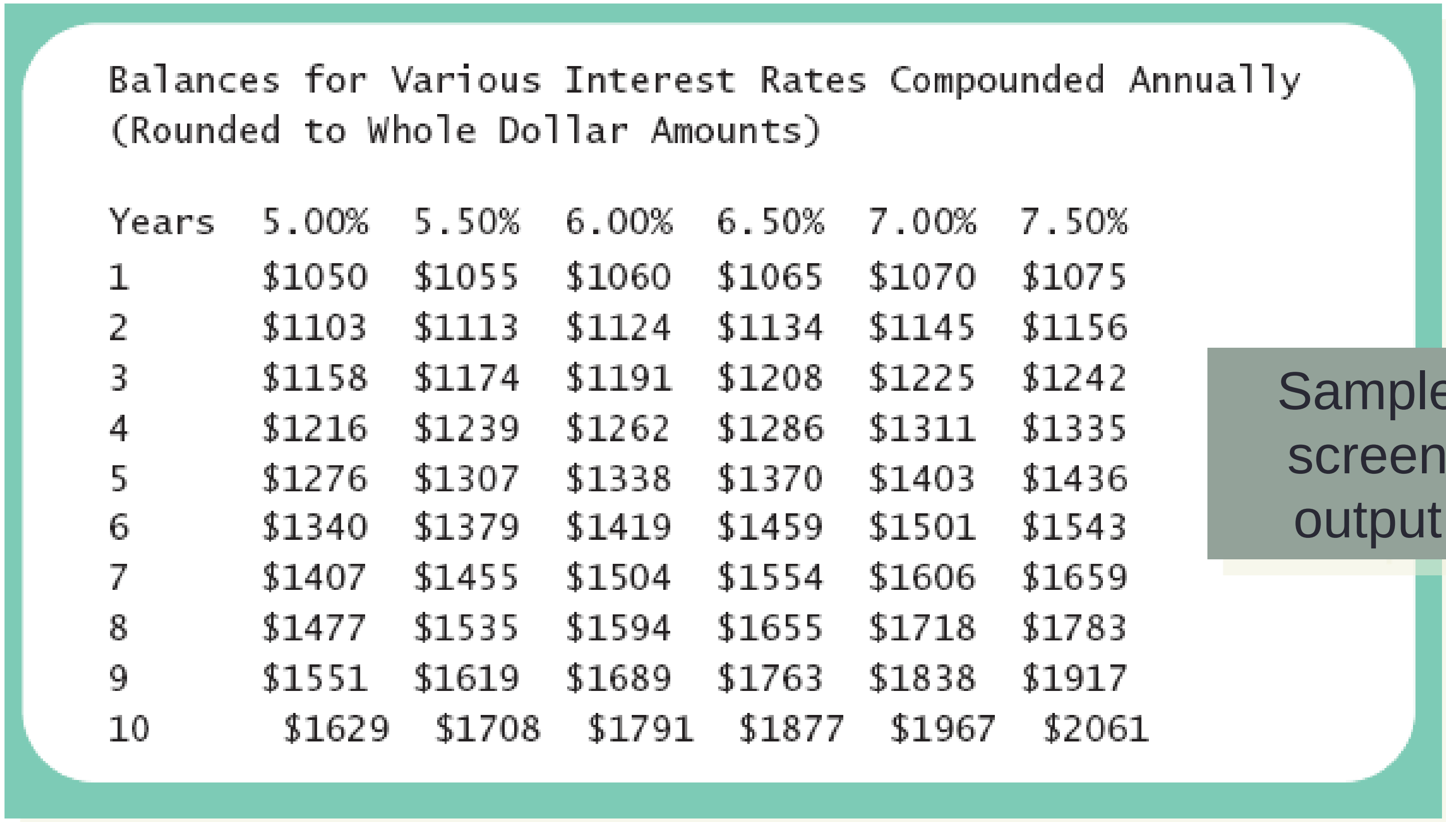
Multidimensional-Array Basics

- We can access elements of the table with a nested for loop
- Example:



```
for (int row = 0; row < 10; row++)
    for (int column = 0; column < 6; column++)
        table[row][column] =
             balance(1000.00, row + 1, (5 + 0.5 * column));
```

Multidimensional-Array Basics



```
Balances for Various Interest Rates Compounded Annually
(Rounded to Whole Dollar Amounts)

Years   5.00%  5.50%  6.00%  6.50%  7.00%  7.50%
1       $1050  $1055  $1060  $1065  $1070  $1075
2       $1103  $1113  $1124  $1134  $1145  $1156
3       $1158  $1174  $1191  $1208  $1225  $1242
4       $1216  $1239  $1262  $1286  $1311  $1335
5       $1276  $1307  $1338  $1370  $1403  $1436
6       $1340  $1379  $1419  $1459  $1501  $1543
7       $1407  $1455  $1504  $1554  $1606  $1659
8       $1477  $1535  $1594  $1655  $1718  $1783
9       $1551  $1619  $1689  $1763  $1838  $1917
10       $1629  $1708  $1791  $1877  $1967  $2061
```

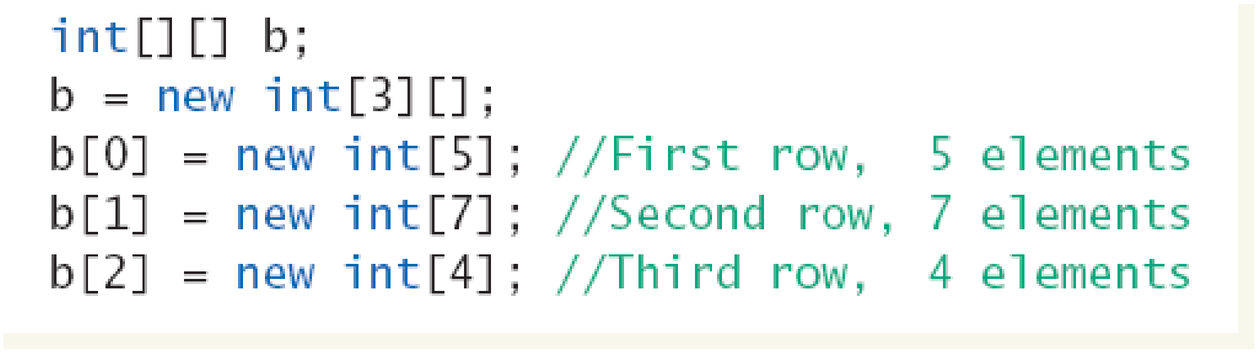
Sample screen output

Java's Representation of Multidimensional Arrays

- Multidimensional array represented as several one-dimensional arrays
- Given
  `int [][] table = new int [10][6];`
- Array table is actually 1 dimensional of type `int[]`
  - It is an array of arrays
- Important when sequencing through multidimensional array

Ragged Arrays

- Not necessary for all rows to be of the same length
- Example:

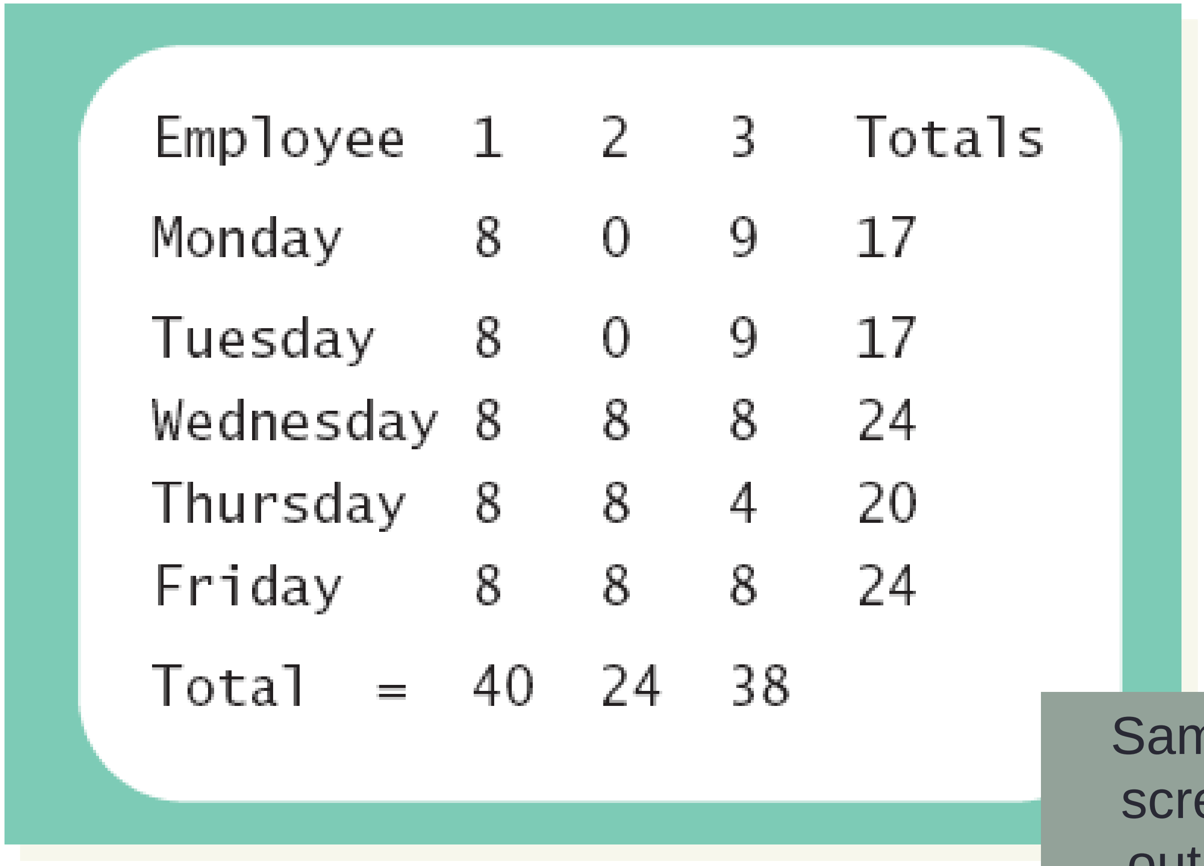


```
int[][] b;
b = new int[3][];
b[0] = new int[5]; //First row,  5 elements
b[1] = new int[7]; //Second row, 7 elements
b[2] = new int[4]; //Third row,  4 elements
```

Programming Example

- Employee Time Records
  - Two-dimensional array stores hours worked
    - For each employee
    - For each of 5 days of work week

Programming Example

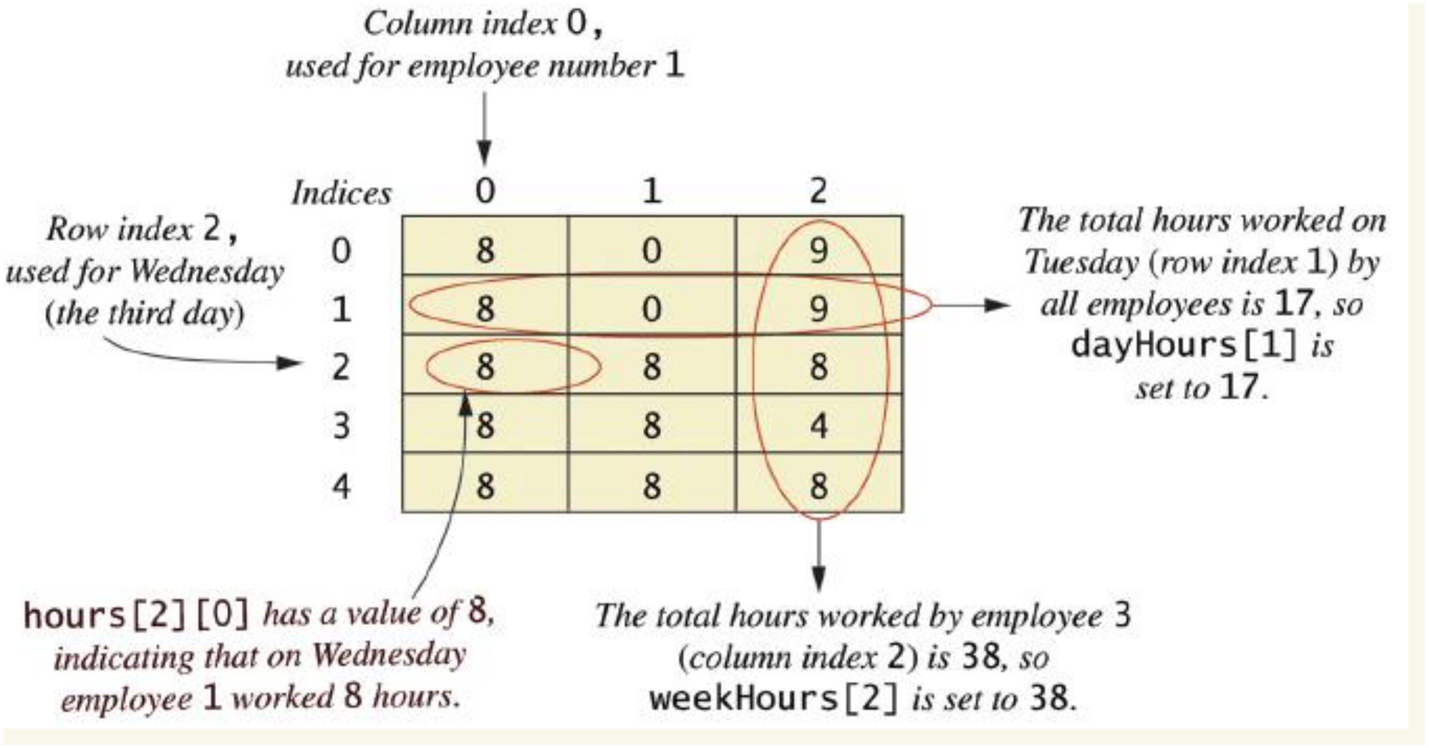


```
Employee   1    2    3    Totals
Monday     8    0    9    17
Tuesday    8    0    9    17
Wednesday  8    8    8    24
Thursday   8    8    4    20
Friday     8    8    8    24
Total    = 40   24   38
```

Sample screen output

Programming Example

- Figure 7.8 Arrays for the class **TimeBook**


Column index 0, used for employee number 1
Indices
Row index 2, used for Wednesday (the third day)
The total hours worked on Tuesday (row index 1) by all employees is 17, so dayHours[1] is set to 17.
hours[2][0] has a value of 8, indicating that on Wednesday employee 1 worked 8 hours.
The total hours worked by employee 3 (column index 2) is 38, so weekHours[2] is set to 38.

	0	1	2
0	8	0	9
1	8	0	9
2	8	8	8
3	8	8	4
4	8	8	8

Summary

- Arrays Revisited
- Two-Dimensional Arrays
- Multidimensional Arrays
- Ragged Arrays