



Arrays

Lecture 21, Mon Mar 8 2010

borrowing from slides by Kurt Eisel

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

Reading

- This week:
 - Chap 7: 7.1, 7.5-7.7. Topics 7.3 and 7.4 (3rd ed)
 - Chap 8: 8.1, 8.5-8.7. Topics 6.3 and 6.4 (2nd ed)

News

- Midterms returned before break
 - get yours after class if you didn't already
- Departmental announcements

Department of Computer Science
Undergraduate Events

Events this week

Drop-In Resume and Cover Letter Editing (20 min. appointments)
Date: Thurs., March 11
Time: 11:30 am – 2:30 pm
Location: Rm 255, ICICS/CS

Townhall Meeting for Combined Majors /Honours, Cogs, BA, B.Comm in CS
Date: Thurs., March 11
Time: 12:30 – 2 pm
Location: DMP 310
Lunch will be provided!

CS Distinguished Lecture Series Featuring David Parkes
Title: Incentive Mechanism Engineering in the Internet Age
Date: Thurs., Mar 11
Time: 3:30 – 4:50 pm
Location: DMP 110

CSSS Movie Night: "Zombieland" & "Iron Man"
Date: Thurs., Mar 11
Time: 6 – 10 pm
Location: DMP 310
Free pop & popcorn for every attendee!

Events next week

Interview Skills Practice Session
Mon., March 15, 12 – 2 pm, Rm 202, ICICS/CS

Transport Canada Info Session
Tues., March 16, 4 – 6 pm, HENN 201

Financial Literacy 101
Wed., March 17, 12 – 1 pm, Angus 426

Townhall Meeting for CS Major/Honours Students
Thurs., March 18, 12:30 – 2 pm, DMP 310

CS Distinguished Lecture Series Featuring Jeff Hawkins
Thurs., March 18, 3:30 – 4:50 pm, DMP 110

Keeping Track of Things



Keeping Track of Things



Answer: Arrays

- use **arrays**: common programming language construct
 - grouping related data items together
 - meaningful organization such that each individual data item can be easily retrieved or updated

Answer: Arrays

- use **arrays**: common programming language construct
 - grouping related data items together
 - meaningful organization such that each individual data item can be easily retrieved or updated
- collection of variables
 - all of same type
 - share common name
- each variable holds single value

Using Arrays

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Collection of variables has single name
 - how do we access individual values?

Using Arrays

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Collection of variables has single name
 - how do we access individual values?
- Each value stored at unique numbered position
 - number called index of array element
 - aka subscript
- cansSold name of this array
 - holds 10 values

Using Arrays

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- To access individual value in array
 - use array name followed by pair of square brackets
 - inside brackets, place index of array element we want to access
- Reference to array element allowed anywhere that variables can be used
- Example:


```
System.out.println(cansSold[4]);
```

 - Prints value 209

Array Declaration and Types

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Just like ordinary variable, must
 - declare array before we use it
 - give array a type
- Since cansSold contains integers, make integer array:


```
int[] cansSold = new int[10]
```
- Looks like variable declaration, except:

Array Declaration and Types

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Just like ordinary variable, must
 - declare array before we use it
 - give array a type
- Since cansSold contains integers, make integer array:


```
int[] cansSold = new int[10]
```
- Looks like variable declaration, except:
 - empty brackets on the left tell Java that cansSold is an array...

Array Declaration and Types

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Just like ordinary variable, must
 - declare array before we use it
 - give array a type
- Since cansSold contains integers, make integer array:


```
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

Array Declaration and Types

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Just like ordinary variable, must
 - declare array before we use it
 - give array a type
- Since cansSold contains integers, make integer array:


```
int[10] 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
 - DO NOT put size of array in brackets on the left

Array Declaration and Types

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Just like ordinary variable, must
 - declare array before we use it
 - give array a type
- Since cansSold contains integers, make integer array:


```
int[10] 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
 - DO NOT put size of array in brackets on the left

Array Declaration and Types

```
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[4] = 209;
        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]);
    }
}
```

17

Array Declaration and Types

```
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

18

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

19

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
}
```

20

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
    }
}
```

21

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
    }
}
```

22

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};
    }
}
```

23

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0;

```

24

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length;

```

25

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)

```

26

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }
    }
}
```

27

Using Arrays and Loops

```
cansSold
0 185
1 92
2 370
3 485
4 209
5 128
6 84
7 151
8 32
9 563
```

- Write program to
 - create array
 - find total number of cans sold
 - print result

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
            System.out.println("We've sold " + totalCans
                + " cans of pop");
        }
    }
}
```

28

Tracing Arrays and Loops

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }
        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}
```

29

Tracing Arrays and Loops

```
public class ArrayTest3
{
    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }
        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}
```

totalCans 0

30

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        public static void main(String[] args)
        {
            int totalCans = 0;
            int[] cansSold = {185, 92, 370, 485, 209,
                128, 84, 151, 32, 563};

            for (int i = 0; i < cansSold.length; i++)
            {
                totalCans = totalCans + cansSold[i];
            }
            System.out.println("We've sold " + totalCans
                + " cans of pop");
        }
    }
}
```

totalCans 0

31

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        public static void main(String[] args)
        {
            int totalCans = 0;
            int[] cansSold = {185, 92, 370, 485, 209,
                128, 84, 151, 32, 563};

            for (int i = 0; i < cansSold.length; i++)
            {
                totalCans = totalCans + cansSold[i];
            }
            System.out.println("We've sold " + totalCans
                + " cans of pop");
        }
    }
}
```

i 0
totalCans 0

32

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 7
totalCans 1704 ■ And so on...
    
```

49

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 8
totalCans 1736 ■ And so on...
    
```

50

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 9
totalCans 2299 ■ And so on...
    
```

51

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 10
totalCans 2299
    
```

52

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 10
totalCans 2299 ■ Is i < 10?
    ■ no, 10 not < 10
    
```

53

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i < cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 10
totalCans 2299 ■ "We've sold 2299 cans of pop" printed out
    
```

54

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i <= cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 10
totalCans 2299 ■ What would happen if we made this little change?
    
```

55

Tracing Arrays and Loops

```

cansSold.length 10 public class ArrayTest3
{
    cansSold
    {
        0 185
        1 92
        2 370
        3 485
        4 209
        5 128
        6 84
        7 151
        8 32
        9 563
    }

    public static void main(String[] args)
    {
        int totalCans = 0;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};

        for (int i = 0; i <= cansSold.length; i++)
        {
            totalCans = totalCans + cansSold[i];
        }

        System.out.println("We've sold " + totalCans
            + " cans of pop");
    }
}

i 10
totalCans 2299 ■ What would happen if we made this little change?

java.lang.ArrayIndexOutOfBoundsException: 10
    
```

56

Something To Remember

```

cansSold.length 10
cansSold
{
    0 185
    1 92
    2 370
    3 485
    4 209
    5 128
    6 84
    7 151
    8 32
    9 563
}
    
```

- Array `cansSold` created with 10 elements
 - Indices (plural of index) are 0 through 9
- In general, array of size `n` will have indices ranging from 0 through `n-1`
- When you number things, you're used to beginning with 1
 - Computer folks begin with 0
 - leads to "off by one" errors, even among computer veterans

57

Initializing Array With Keyboard Input

```

import java.util.Scanner;
public class ArrayTest3b
{
    public static void main(String[] args)
    {
        final int ARRAYSIZE = 10;
        int[] cansSold = new int[ARRAYSIZE];
        Scanner scan = new Scanner(System.in);

        for (int i = 0; i < cansSold.length; i++)
        {
            System.out.print("Enter machine " +
                (i+1));
            cansSold[i] = scan.nextInt();
        }

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

58

Averaging Loop Example

```

numbers
{
    0 6
    1 8
    2 11
    3 18
    4 20
    5 17
    6 14
    7 10
    8 5
    9 2
}
    
```

- Let's say we want to write a program that prints average of values in some arbitrarily large array
 - like the one to the left called numbers
- Will require loop
- Simple task for looping in the context of an array
 - how will we make this happen?

59

PrintMax Loop Example

```

numbers
{
    0 6
    1 8
    2 11
    3 18
    4 20
    5 17
    6 14
    7 10
    8 5
    9 2
}
    
```

- Now instead of average, we want to find and print maximum value from some arbitrarily large array
 - Similar loop, but with some extra tweaks.

60

Histogram Loop Example

```

numbers
{
    0 6 *****
    1 8 *****
    2 11 *****
    3 18 *****
    4 20 *****
    5 17 *****
    6 14 *****
    7 10 *****
    8 5 *****
    9 2 **
}
    
```

- Now use same data as basis for histogram
 - Write one loop to look at value associated with each row of array
 - for each value print a line with that many asterisks
 - For example, if program reads value 6 from the array, should print line of 6 asterisks
 - Program then reads the value 8, prints a line of 8 asterisks, and so on.
 - Need outer loop to read individual values in the array
 - Need inner loop to print asterisks for each value

61

Storing Different Data Types

```

cansSold
{
    0 185
    1 92
    2 370
    3 485
    4 209
    5 128
    6 84
    7 151
    8 32
    9 563
}
    
```

62

Storing Different Data Types

```

cansSold  cashIn
{
    0 185 0 201.20
    1 92 1 100.50
    2 370 2 112.75
    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
}
    
```

Could use two arrays of same size but with different types

63

Storing Different Data Types

```

cansSold  cashIn
{
    0 185 0 201.20
    1 92 1 100.50
    2 370 2 112.75
    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
}
    
```

- Write program to compare what's been collected from each machine vs. how much should have been collected?

Could use two arrays of same size but with different types

64

Storing Different Data Types

- Write program to compare what's been collected from each machine vs. how much should have been collected?

```

cansSold  cashIn
0 185 0 201.25
1 92 1 100.50
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

public class ArrayTest4
{
    public static void main(String[] args)
    {
        double expected;
        int[] cansSold = {185, 92, 370, 485, 209,
            128, 84, 151, 32, 563};
        double[] cashIn = {201.25, 100.50, 412.75,
            555.25, 195.00, 160.00,
            105.00, 188.75, 40.00,
            703.75};

        for (int i = 0; i < cansSold.length; i++)
        {
            expected = cansSold[i] * 1.25;
            System.out.println("Machine " + (i + 1) +
                " off by $" +
                (expected - cashIn[i]));
        }
    }
}

```

Could use two arrays of same size but with different types

Storing Different Data Types

- Write program to compare what's been collected from each machine vs. how much should have been collected?

```

cansSold  cashIn
0 185 0 201.25
1 92 1 100.50
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

public class ArrayTest4
{
    double expected;
    int[] cansSold = {185, 92, 370, 485, 209,
        128, 84, 151, 32, 563};
    double[] cashIn = {201.25, 100.50, 412.75,
        555.25, 195.00, 160.00,
        105.00, 188.75, 40.00,
        703.75};

    for (int i = 0; i < cansSold.length; i++)
    {
        expected = cansSold[i] * 1.25;
        System.out.println("Machine " + (i + 1) +
            " off by $" +
            (expected - cashIn[i]));
    }
}

```

Could use two arrays of same size but with different types

What happens when we run the program?

Storing Different Data Types

```

cansSold  cashIn  Machine  off by $30.0
0 185 0 201.25  Machine 1 off by $14.5
1 92 1 100.50  Machine 2 off by $49.75
2 370 2 412.75  Machine 3 off by $51.0
3 485 3 555.25  Machine 4 off by $66.25
4 209 4 195.00  Machine 5 off by $0.0
5 128 5 160.00  Machine 6 off by $0.0
6 84 6 105.00  Machine 7 off by $0.0
7 151 7 188.75  Machine 8 off by $0.0
8 32 8 40.00  Machine 9 off by $0.0
9 563 9 703.75  Machine 9 off by $0.0

```

Somebody has been stealing from the machines after all! We need an anti-theft plan...

Arrays With Non-Primitive Types

```

cansSold  cashIn
0 185 0 201.25
1 92 1 100.50
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

```

- 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

```

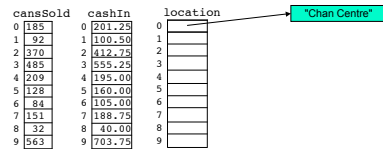
cansSold  cashIn  location
0 185 0 201.25 0
1 92 1 100.50 1
2 370 2 412.75 2
3 485 3 555.25 3
4 209 4 195.00 4
5 128 5 160.00 5
6 84 6 105.00 6
7 151 7 188.75 7
8 32 8 40.00 8
9 563 9 703.75 9

```

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

```
String[] location = new String[10];
```

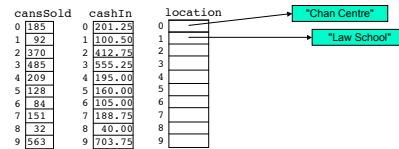
Arrays of Objects



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

```
location[0] = "Chan Centre";
```

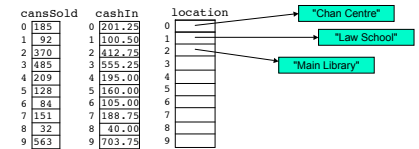
Arrays of Objects



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

```
location[0] = "Chan Centre";
location[1] = "Law School";
```

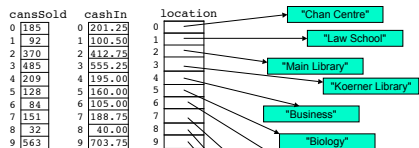
Arrays of Objects



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

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

Arrays of Objects

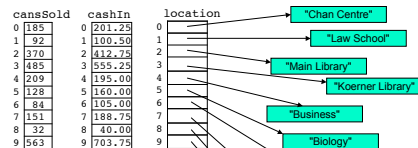


- Now we can put references to Strings in our String array.

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

...and so on...

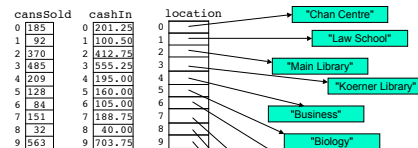
Arrays of Objects



- Or we could have done this:

```
String[] location =
{"Chan Centre", "Law School",
"Main Library", ... };
```

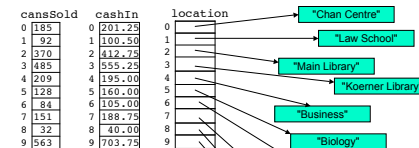
Arrays of Objects



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

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

Arrays of Objects

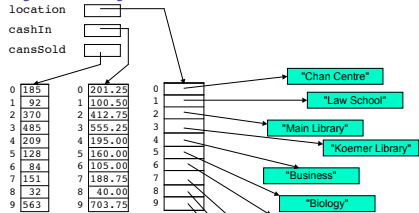


- Each individual String object in array of course has all String methods available
- For example, what would this return?

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

12

Arrays of Objects



- Think about a cleaner way to do all this...