



University of British Columbia
CPSC 111, Intro to Computation
Jan-Apr 2006

Tamara Munzner

Arrays

Lecture 14, Tue Feb 28 2006

based on slides by Kurt Eiselt

<http://www.cs.ubc.ca/~tmm/courses/cpsc111-06-spr>

News

- Assignment 2
 - corrections to `ASCIIArtiste.java` posted
 - definitely read WebCT bboards!

Reading

- This week: 8.1, 8.5-8.7, topics 6.3 and 6.4

Recap: While Loop Example

```
public class WhileDemo
{
    public static void main (String[] args)
    {
        int limit = 3;
        int counter = 1;

        while (counter <= limit)
        {
            System.out.println("The square of " + counter +
                               " is " + (counter * counter));

            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

■ while version

Recap: For Loop Example

```
public class ForDemo
{
    public static void main (String[] args)
    {
        for (int counter = 1; counter <= 3; counter = counter + 1)
        {
            System.out.println("The square of " + counter +
                               " is " + (counter * counter));
        }
        System.out.println("End of demonstration");
    }
}
```

- **for** version

Recap: Do Loop Example

```
public class DoDemo
{
    public static void main (String[] args)
    {
        int limit = 3;
        int counter = 1;

        do
        {
            System.out.println("The square of " + counter +
                               " is " + (counter * counter));

            counter = counter + 1;
        } while (counter <= limit);

        System.out.println("End of demonstration");
    }
}
```

■ do version

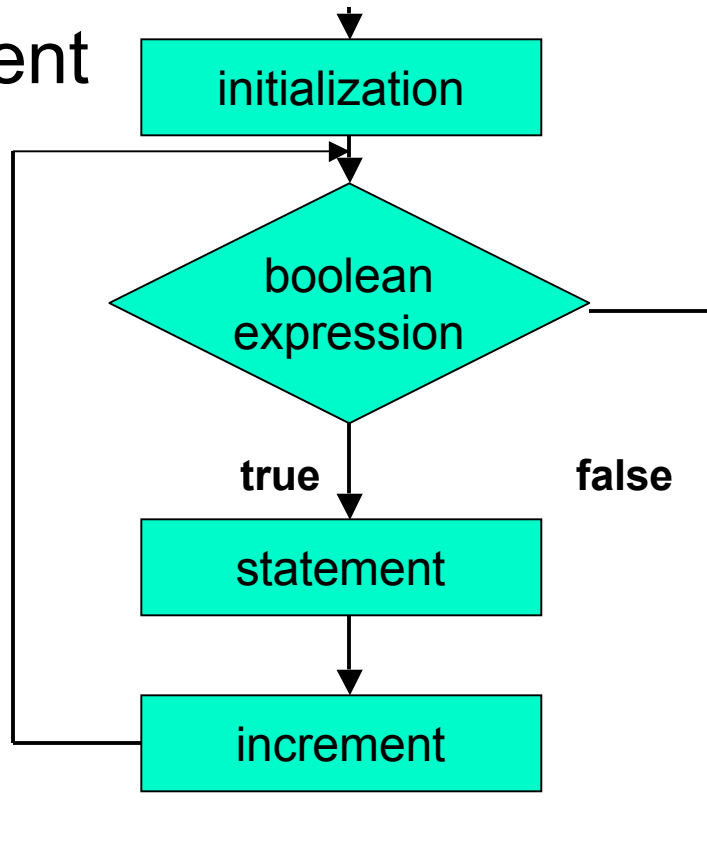
Recap: For Statement

```
for (initialization; boolean expression; increment)
    body
```

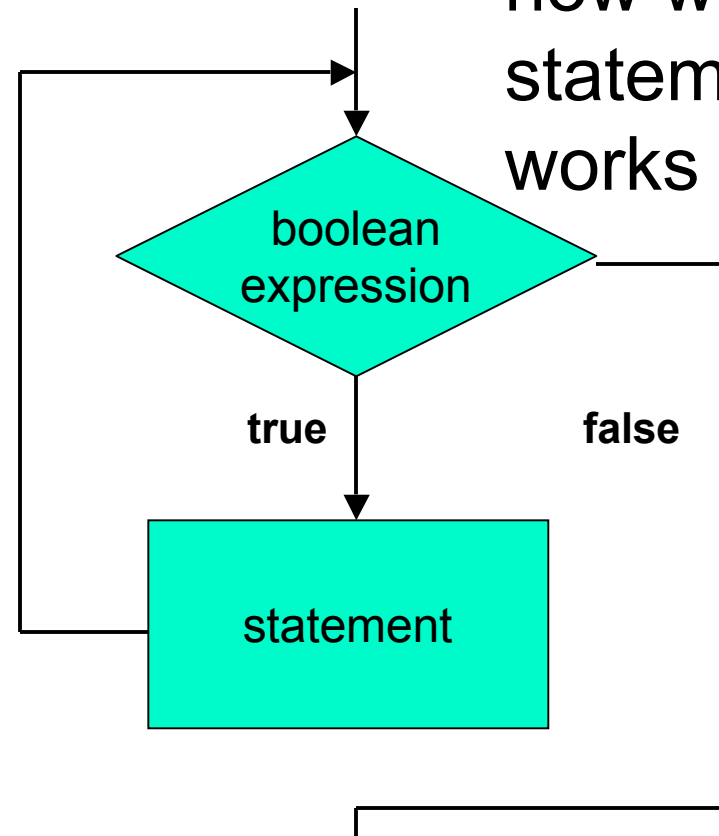
- **Body** of loop can be
 - single statement
 - whole block of many statements in curly braces
- Control flow
 - first time through: initialization
 - boolean expression evaluated
 - if expression true, body executed; if false, end
 - increment processed
 - boolean expression evaluated
 - if true, body executed; if false, end
 -

Recap: For Versus While Statement

how for
statement
works

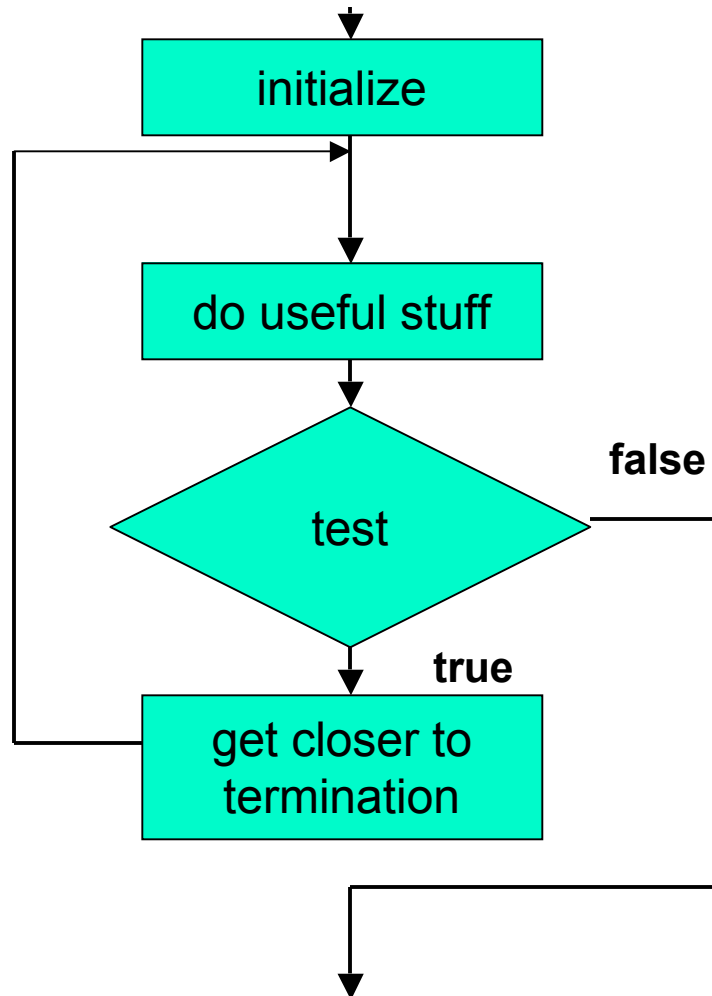


how while
statement
works



- flowcharts can be somewhat deceptive
 - need initialization and incrementing/modifying in while loop too
 - although syntax does not require it in specific spot

Recap: Do Statement



- Body always executed at least once

order of four things can change, but need them all

Objectives

- More practice with loops
- Understand when and how to use arrays
 - and loops over arrays

Flipping Coins

- Did `while` version last time
- Let's try `for` version now

Keeping Track of Things

Cans of pop sold
this month

- 185
- 92
- 370
- 485
- 209
- 128
- 84
- 151
- 32
- 563



What's the gross income?
 What's the net profit?
 Is Bubba stealing loonies?

Keeping Track of Things

Cans of pop sold
this month

- 185
- 92
- 370
- 485
- 209
- 128
- 84
- 151
- 32
- 563

In other words, how can I
organize the data above in
my computer so that I can
access it easily and do the
computations I need to do?



Answer: Arrays

Cans of pop sold
this month

185
92
370
485
209
128
84
151
32
563

- 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

cansSold

185
92
370
485
209
128
84
151
32
563

- 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

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

cansSold

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

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)
    {
        cansSold
        {
            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]);
        }
    }
}
```

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

Array Declaration and Types

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

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

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

- 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

Using Arrays and Loops

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

cansSold

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

Using Arrays and Loops

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

cansSold

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

```
public class ArrayTest3  
{
```

```
}
```

Using Arrays and Loops

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

cansSold

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

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

    }
}
```

Using Arrays and Loops

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

cansSold

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

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

    }
}
```


Using Arrays and Loops

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

cansSold

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

```
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;

}
}
```

Using Arrays and Loops

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

cansSold

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

```
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;

}
}
```


Using Arrays and Loops

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

cansSold

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

```
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++)

    }
}
```

Using Arrays and Loops

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

cansSold

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

```
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];
        }

    }
}
```

Using Arrays and Loops

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

cansSold

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

```
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");
    }
}
```

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");
    }
}
```

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

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128
6 84
7 151
8 32
9 563
totalCans 0     }
```

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

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

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
```

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

Note: In the original image, a red arrow points to index 0, a cyan box highlights '10', and a red box highlights '(int i = 0;)' in the code.

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");
        }
    }
}
```

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

i 0

totalCans 0

- Is $i < 10$?
- yes, $0 < 10$

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 185
```

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

Note: In the original image, a red arrow points to index 0, and a red box highlights the line `totalCans = totalCans + cansSold[i];` in the code.

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
                + " cans of pop");
                }
                }
i 1
totalCans 185
```

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

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");
    }
  }
}
```

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

i 1 }

totalCans 185

- Is $i < 10$?
- yes, $1 < 10$

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 1
totalCans 277
```

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

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
+ " cans of pop");
i 2            }
totalCans 277  }
```

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




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 2
totalCans 277 ■ Is i < 10?
                ■ yes, 2 < 10
```

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




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 2
totalCans 647
```

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



Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
                + " cans of pop");
                }
                }
i 3
totalCans 647
```

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

Tracing Arrays and Loops

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

```
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");
    }
}
```

i 3

totalCans 647

- Is $i < 10$?
- yes, $3 < 10$

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           { totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
                + " cans of pop");
                }
                }
                }
                }

i 3
totalCans 1132
```


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

The code snippet shows a Java class `ArrayTest3` with a `main` method. It initializes `totalCans` to 0 and creates an array `cansSold` with 10 elements. A `for` loop iterates from `i = 0` to `i < cansSold.length`. The line `totalCans = totalCans + cansSold[i];` is highlighted in red. The current state of the program is shown at the bottom: `i` is 3 and `totalCans` is 1132.

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
+ " cans of pop");
i 4            }
totalCans 1132 }
```

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



Tracing Arrays and Loops

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

```
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");
    }
}
```

i 4

totalCans 1132

- Is $i < 10$?
- yes, $4 < 10$

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1  92           {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6  84           {
7 151           { totalCans = totalCans + cansSold[i];
8  32           }
9 563           System.out.println("We've sold " + totalCans
+ " cans of pop");
}
}


i 4
totalCans 1341 ■ And so on...
```

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

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           { totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
                + " cans of pop");
                }
                }
                }
                }
                }
                }
                }
                }
                }
                }
```

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



`i` 5 }

`totalCans` 1469 ■ And so on...

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
+ " cans of pop");
i 6            }
totalCans 1553  }
```

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

→

■ And so on...

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");
        }
    }
}
```

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



i

7

totalCans

1704

■ And so on...

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128
6 84
7 151
8 32
9 563
```



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...

Tracing Arrays and Loops

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

```
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");
    }
}
```

i 9

totalCans 2299 ■ And so on...

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185            public static void main(String[] args)
1 92             {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128
6 84
7 151
8 32
9 563
→
```

```
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
```

Tracing Arrays and Loops

```
cansSold.length 10 public class ArrayTest3
cansSold        {
0 185           public static void main(String[] args)
1 92            {
2 370           int totalCans = 0;
3 485           int[] cansSold = {185, 92, 370, 485, 209,
4 209           128, 84, 151, 32, 563};
5 128           for (int i = 0; i < cansSold.length; i++)
6 84            {
7 151           totalCans = totalCans + cansSold[i];
8 32            }
9 563           System.out.println("We've sold " + totalCans
                + " cans of pop");

```



i 10 }

totalCans 2299

- Is $i < 10$?
- no, $10 \text{ not } < 10$

Tracing Arrays and Loops

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

→

```
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");
    }
}
```

i 10

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

Tracing Arrays and Loops

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

```
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");
    }
}
```

10

2299

■ What would happen if we made this little change?

Tracing Arrays and Loops

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

```
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 2299

■ What would happen if we made this little change?

java.lang.ArrayIndexOutOfBoundsException: 10

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

Initializing Array With Keyboard Input

cansSold

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

```
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]);
    }
}
```

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?

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.

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

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

Storing Different Data 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

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

Could use two arrays of same size but with different types

Storing Different Data 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

Could use two arrays of same size but with different types

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

```
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]));
        }
    }
}
```


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

Could use two arrays of same size but with different types

What happens when we run the program?

```
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]));
        }
    }
}
```

Storing Different Data Types

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

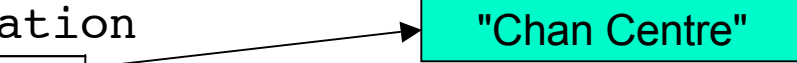
cansSold	cashIn	location
0 185	0 201.25	0
1 92	1 100.50	1
2 370	2 412.75	2
3 485	3 555.25	3
4 209	4 195.00	4
5 128	5 160.00	5
6 84	6 105.00	6
7 151	7 188.75	7
8 32	8 40.00	8
9 563	9 703.75	9

- 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

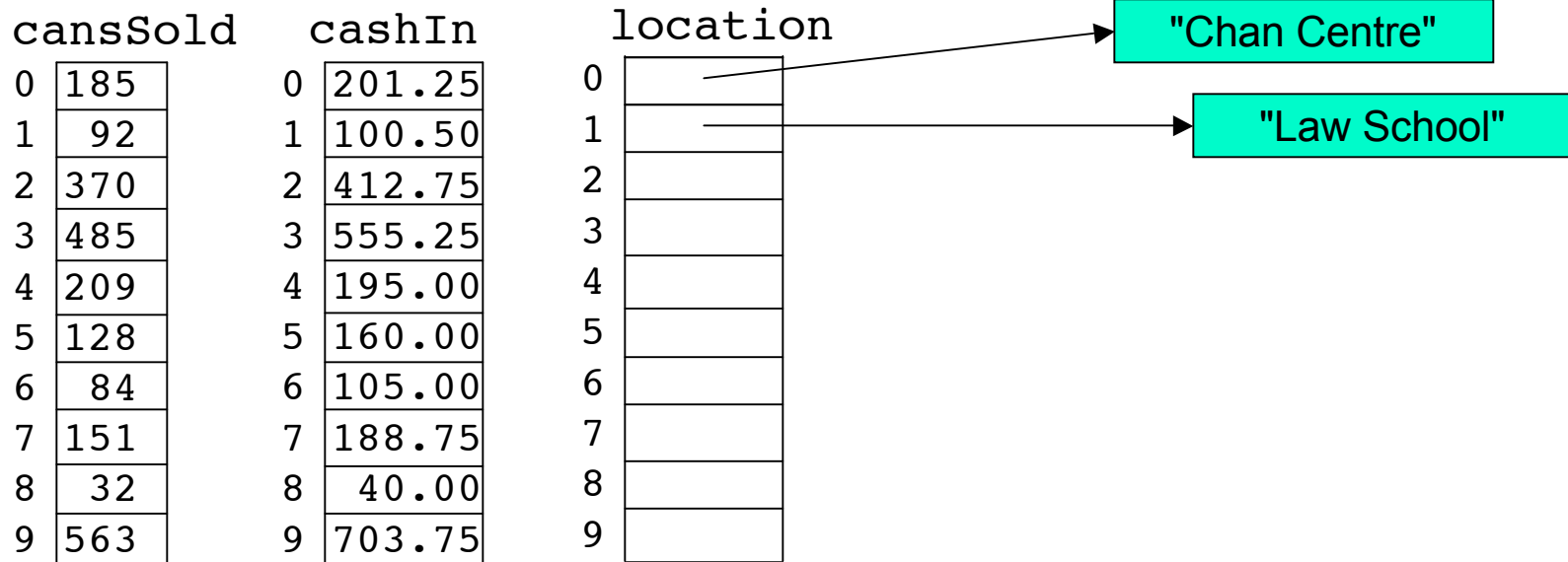
cansSold	cashIn	location
0 185	0 201.25	0
1 92	1 100.50	1
2 370	2 412.75	2
3 485	3 555.25	3
4 209	4 195.00	4
5 128	5 160.00	5
6 84	6 105.00	6
7 151	7 188.75	7
8 32	8 40.00	8
9 563	9 703.75	9



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

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

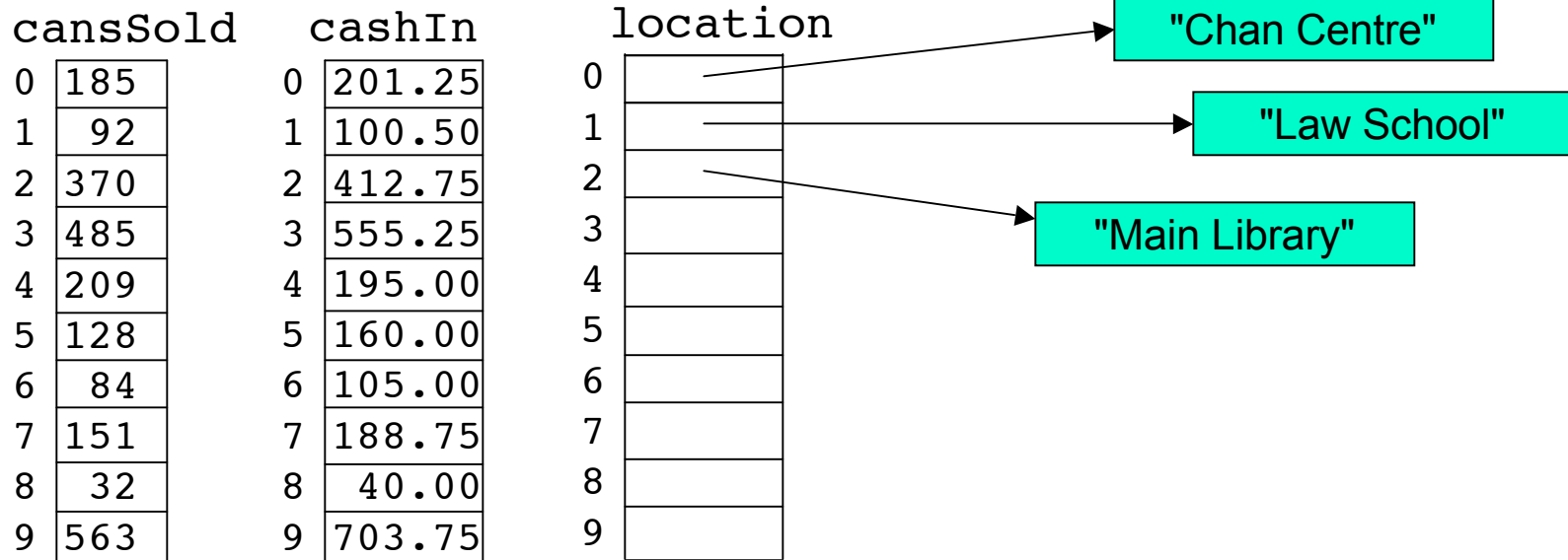
Arrays of Objects



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

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

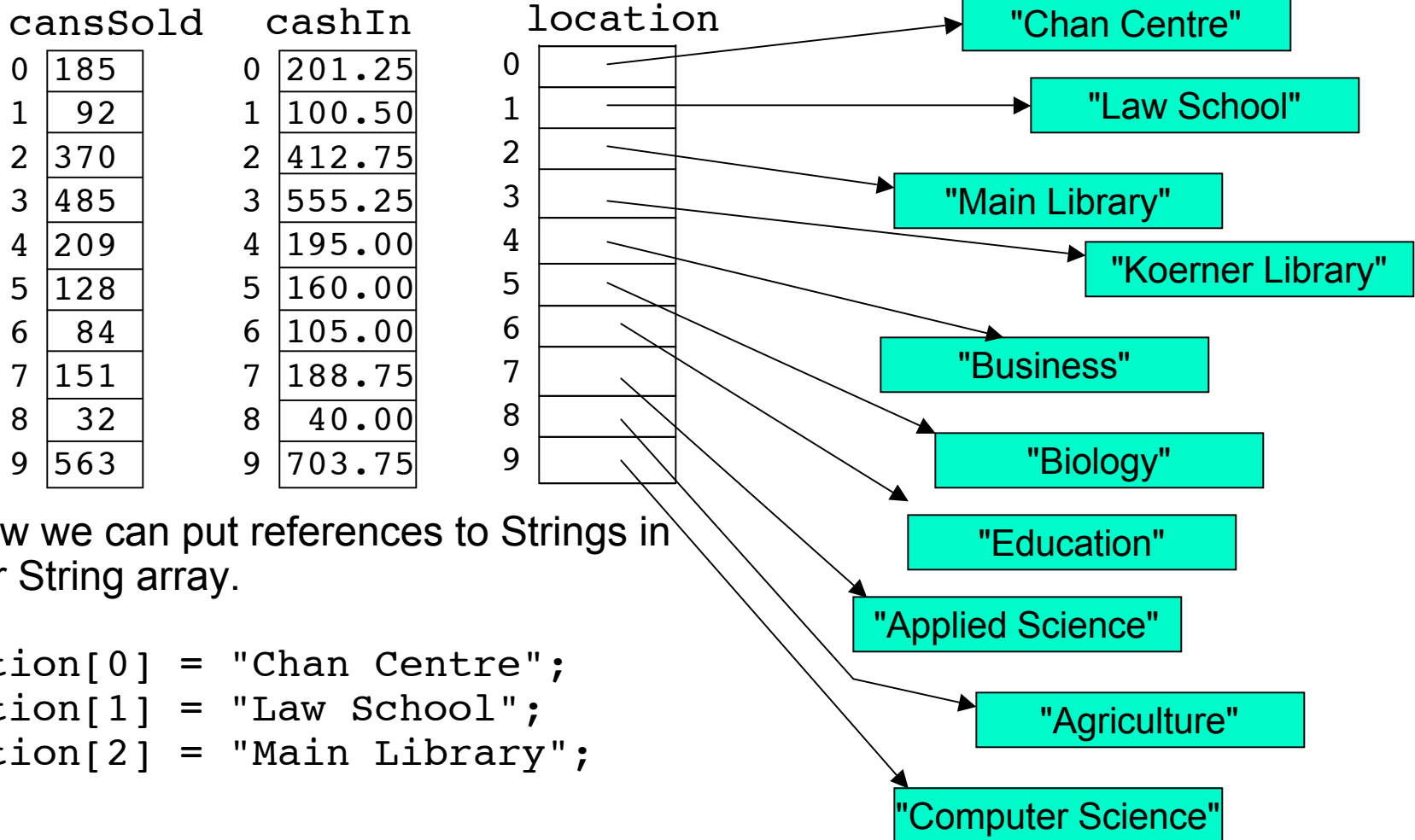
Arrays of Objects



- 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

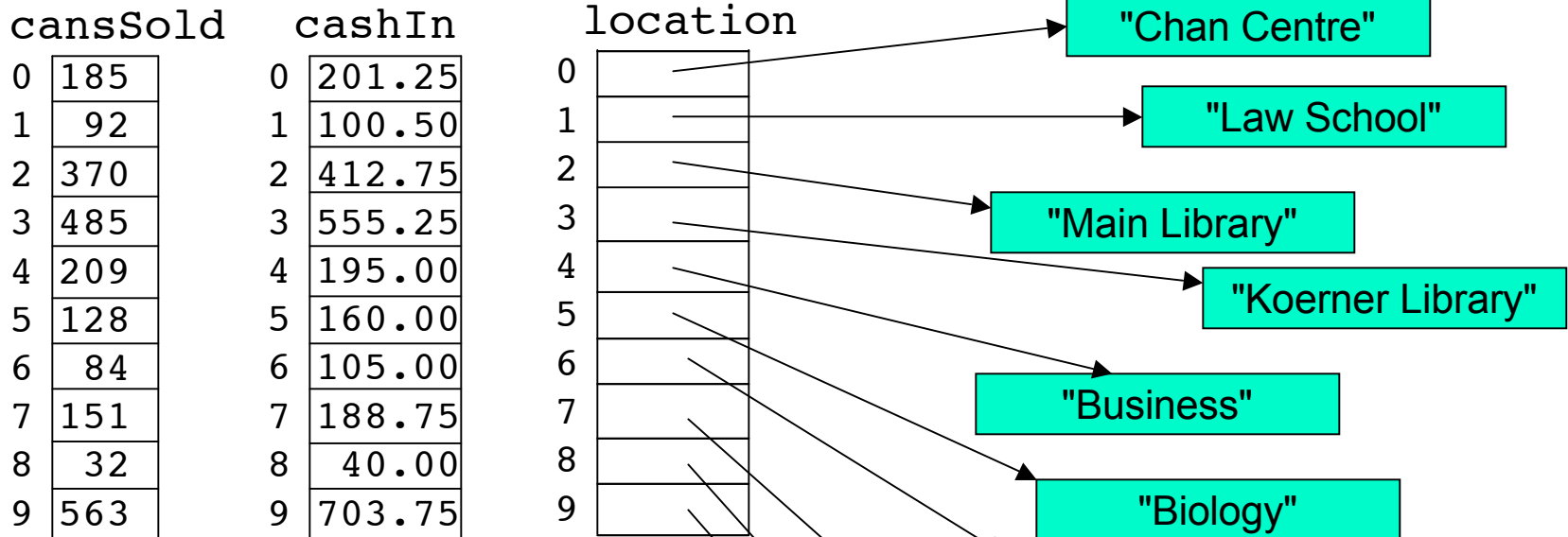


- Now we can put references to Strings in our String array.

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

...and so on...

Arrays of Objects

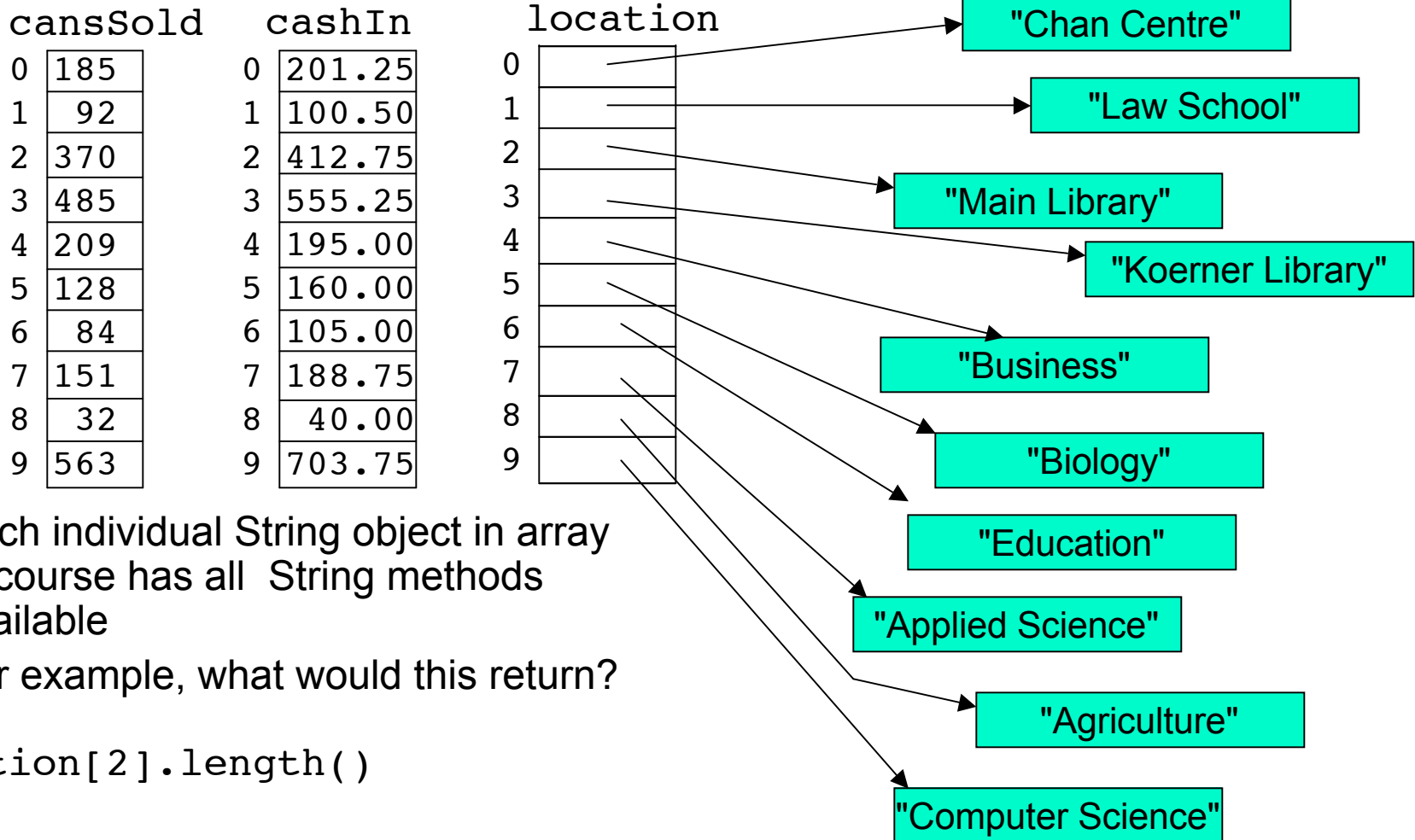


- Or we could have done this:

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

"Computer Science"

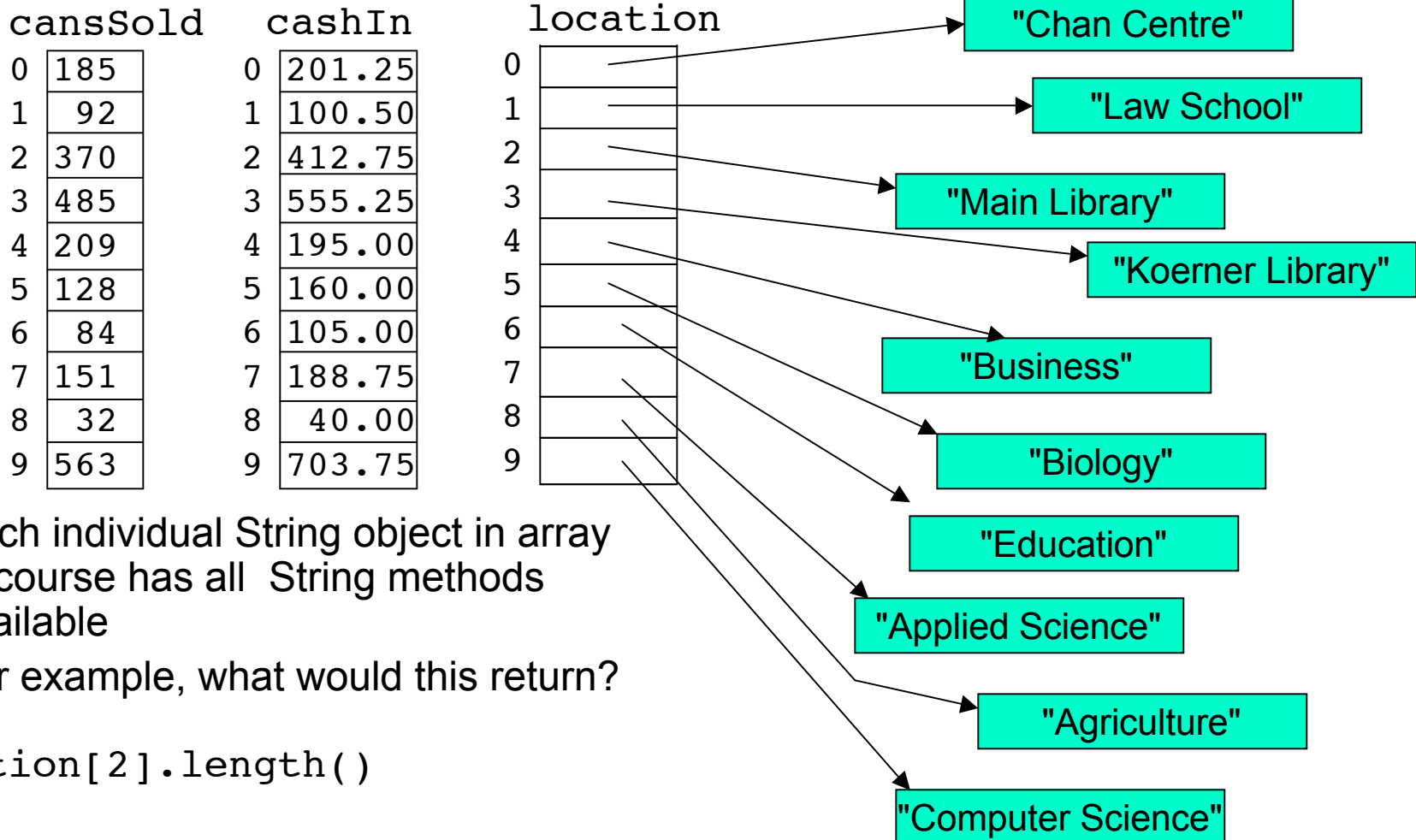
Arrays of Objects



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

`location[2].length()`

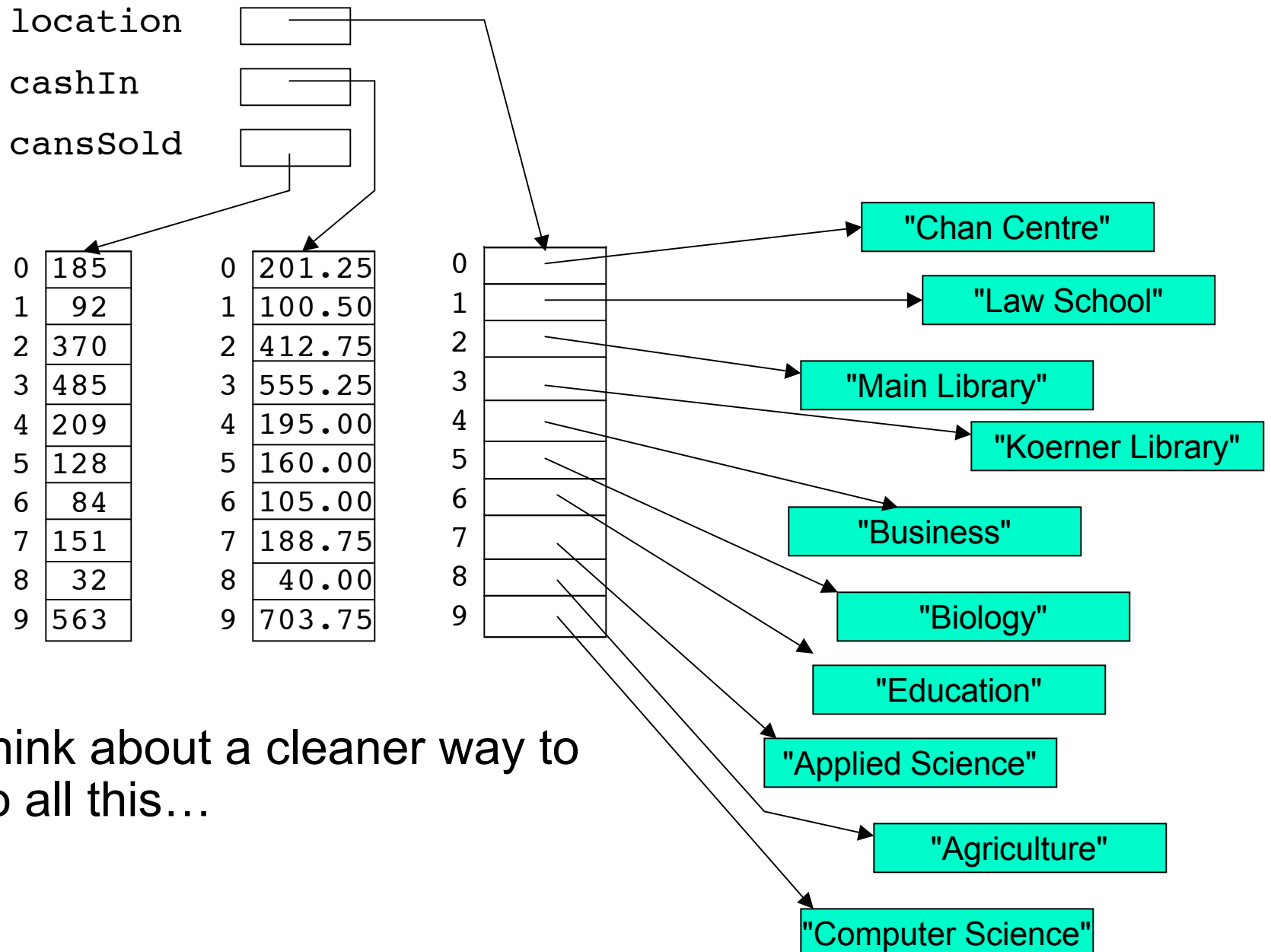
Arrays of Objects



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

`location[2].length()`

Arrays of Objects



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