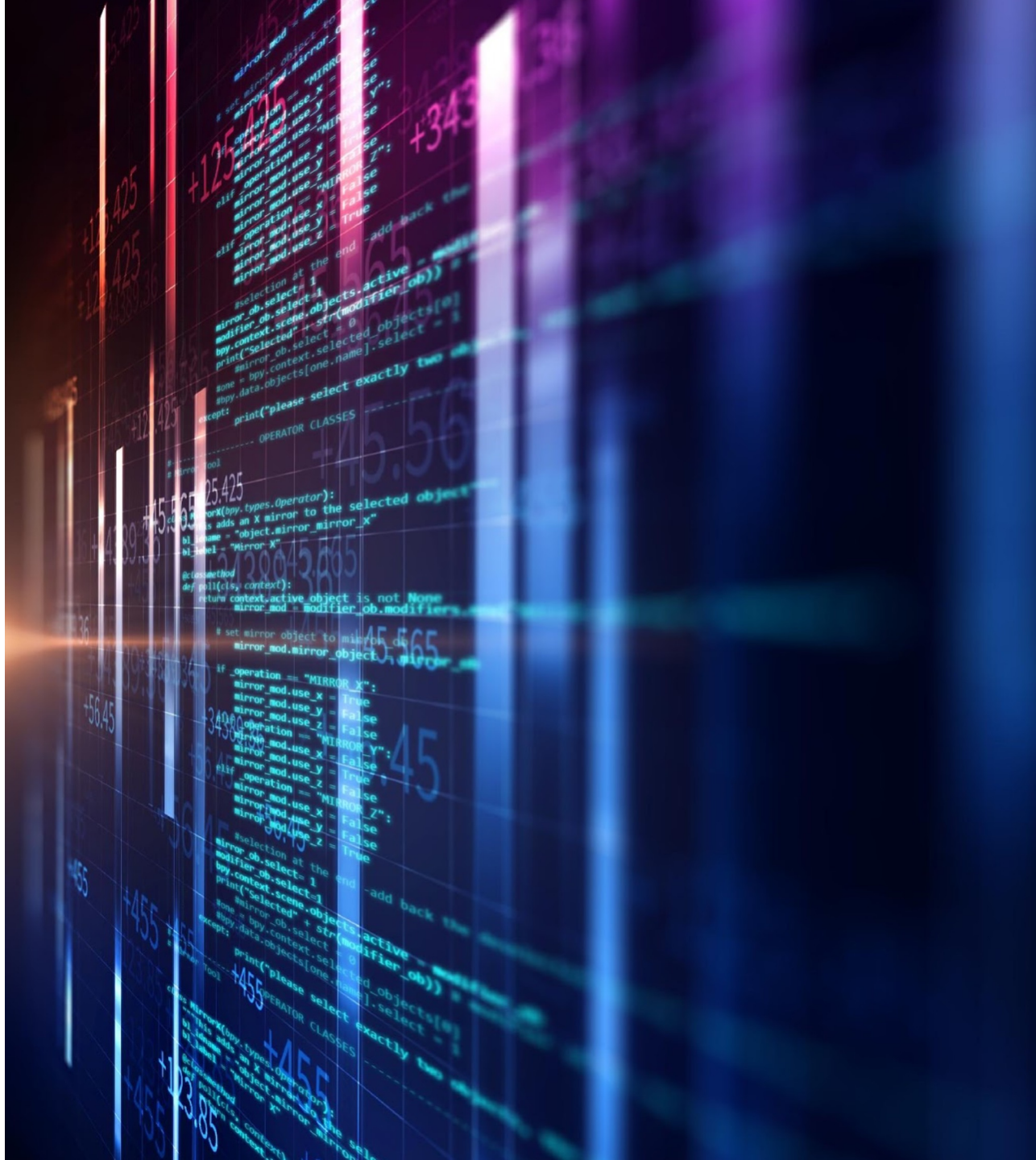
Arrays

Brandon Krakowsky

Penn Engineering



Schedule for Next Few Weeks

Schedule for Next Few Weeks

- 9/26: Arrays
- 9/28: More with Arrays & Strings
- 10/3: More with Classes, Fields & Constructors, and Access Modifiers
  - HW5 assigned (due 10/12)
- 10/5: No Class -- Open Office Hours (in Meyerson Hall B3)
- 10/6: No Recitation -- Fall Break!
- 10/10: No Class -- Open Office Hours (in Meyerson Hall B3)
- 10/12: Unit Testing & Test-Driven Development
  - Midterm assigned (due 10/17)

Arrays

Arrays

- An array is a way to store a collection of items, as a single unit
  - For reference, it's like a list in Python, but not exactly …

Arrays

- An array is a way to store a collection of items, as a single unit
  - For reference, it's like a list in Python, but not exactly …
- Arrays are defined with a *fixed number of slots*, each of which holds an individual item
  - You can add and delete items from those slots as needed
  - Arrays are *mutable*, which means, once defined, the individual items can be changed

Arrays

- An array is a way to store a collection of items, as a single unit
  - For reference, it's like a list in Python, but not exactly …
- Arrays are defined with a *fixed number of slots*, each of which holds an individual item
  - You can add and delete items from those slots as needed
  - Arrays are *mutable*, which means, once defined, the individual items can be changed
- Each item in an array is distinguished by a numerical *index* between 0 and the array size minus 1
  - For reference, this works exactly the same as list indexing in Python

Arrays

- An array is a way to store a collection of items, as a single unit
  - For reference, it's like a list in Python, but not exactly …
- Arrays are defined with a *fixed number of slots*, each of which holds an individual item
  - You can add and delete items from those slots as needed
  - Arrays are *mutable*, which means, once defined, the individual items can be changed
- Each item in an array is distinguished by a numerical *index* between 0 and the array size minus 1
  - For reference, this works exactly the same as list indexing in Python
- Arrays *can* contain any type of element value (primitive types or Objects), but you *can't* store different types in a single array
  - You can have an array of ints, an array of Strings, or even an array of arrays, but you can't have an array that contains, for example, both Strings and ints

Arrays

- An array is a way to store a collection of items, as a single unit
  - For reference, it's like a list in Python, but not exactly …
- Arrays are defined with a *fixed number of slots*, each of which holds an individual item
  - You can add and delete items from those slots as needed
  - Arrays are *mutable*, which means, once defined, the individual items can be changed
- Each item in an array is distinguished by a numerical *index* between 0 and the array size minus 1
  - For reference, this works exactly the same as list indexing in Python
- Arrays *can* contain any type of element value (primitive types or Objects), but you *can't* store different types in a single array
  - You can have an array of ints, an array of Strings, or even an array of arrays, but you can't have an array that contains, for example, both Strings and ints
- An array itself, is an Object

Creating Arrays – 1st Step

To create an array in Java, you first declare a variable to hold the array

- Array variables indicate the type of object the array will hold, followed by empty brackets [], and the name of the array
- For example, this declares an array of ints:
  ```
  int[] myArrayOfInts;
  ```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 1st Step

To create an array in Java, you first declare a variable to hold the array

- Array variables indicate the type of object the array will hold, followed by empty brackets [], and the name of the array
- For example, this declares an array of ints:
  ```
  int[] myArrayOfInts;
  ```
- This declares an array of Strings:
  ```
  String[] myArrayofStrings;
  ```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 1st Step

To create an array in Java, you first declare a variable to hold the array

- Array variables indicate the type of object the array will hold, followed by empty brackets [], and the name of the array
- For example, this declares an array of ints:
  ```
  int[] myArrayOfInts;
  ```
- This declares an array of Strings:
  ```
  String[] myArrayofStrings;
  ```
- Imagine we have a Customer class. This declares an array of Customers:
  ```
  Customer[] myArrayOfCustomers;
  ```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 2nd Step

- The second step is to create an array object and assign it to that variable. There are two ways to do this.
- One way is to use the *new* operator to create a new instance of an array

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 2nd Step

- The second step is to create an array object and assign it to that variable. There are two ways to do this.
- One way is to use the *new* operator to create a new instance of an array
- This creates a new array of Strings with 10 slots (sometimes called elements)

```
String[] names = new String[10]; //declare and create instance of array of 10 Strings
```

  - When you create an array object using *new*, you must indicate how many slots that array will hold, inside the brackets []
  - This does not put actual String values in the slots -- you'll have to do that later

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 2nd Step

- The second step is to create an array object and assign it to that variable. There are two ways to do this.
- One way is to use the *new* operator to create a new instance of an array
- This creates a new array of Strings with 10 slots (sometimes called elements)

```
String[] names = new String[10]; //declare and create instance of array of 10 Strings
```

  - When you create an array object using *new*, you must indicate how many slots that array will hold, inside the brackets []
  - This does not put actual String values in the slots -- you'll have to do that later

- This creates a new array of ints with 99 slots

```
int[] temps; //declare array
temps = new int[99]; //create instance of array with 99 slots
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 3rd Step

The third step is to store things in the array

- You can access the value in any slot of an array by specifying the index number inside brackets []
  - Again, this works exactly the same as list indexing in Python
  - Remember, indexing starts at 0

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 3rd Step

The third step is to store things in the array

- You can access the value in any slot of an array by specifying the index number inside brackets []
  - Again, this works exactly the same as list indexing in Python
  - Remember, indexing starts at 0
- This creates an array of 3 doubles and sets the values

```
double[] myDoubleArray = new double[3];
myDoubleArray[0] = 5.0; //sets 1st value to 5.0
myDoubleArray[1] = 4.1; //sets 2nd value to 4.1
myDoubleArray[2] = 3.9; //sets 3rd  value to 3.9
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 3rd Step

- This creates an array of 2 booleans and sets the values

```
boolean[] myBoolArray = new boolean[2];
myBoolArray[1] = true; //sets 2nd value to true
myBoolArray[0] = false; //sets 1st value to false
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Creating Arrays – 3rd Step

- This creates an array of 2 booleans and sets the values

```
boolean[] myBoolArray = new boolean[2];
myBoolArray[1] = true; //sets 2nd value to true
myBoolArray[0] = false; //sets 1st value to false
```

- Note, if you use an index outside of 0 up to myArray.length - 1, you'll get an *ArrayIndexOutOfBoundsException*

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Initializing Arrays in One Line

Another way to create an array is to enclose the elements of the array inside curly braces {}, separated by commas

- This initializes the contents (values) of the array in the array declaration
- For example, this creates an array of ints with actual prime numbers

```
int[] primes = {2, 3, 5, 7, 11, 13, 19};
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Initializing Arrays in One Line

Another way to create an array is to enclose the elements of the array inside curly braces {}, separated by commas

- This initializes the contents (values) of the array in the array declaration
- For example, this creates an array of ints with actual prime numbers
  ```
  int[] primes = {2, 3, 5, 7, 11, 13, 19};
  ```
- This creates an array of Strings, with actual programming languages
  ```
  String[] languages = {"Java", "C", "C++”};
  ```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Initializing Arrays in One Line

Another way to create an array is to enclose the elements of the array inside curly braces {}, separated by commas

- This initializes the contents (values) of the array in the array declaration
- For example, this creates an array of ints with actual prime numbers
```
int[] primes = {2, 3, 5, 7, 11, 13, 19};
```
- This creates an array of Strings, with actual programming languages
```
String[] languages = {"Java", "C", "C++”};
```
- Imagine we have a Customer class. This creates an array of Customers, with actual customers
```
Customer[] customers = {new Customer(“Brandon”), new
Customer(“Betsy”)};
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Initializing Arrays in One Line

Another way to create an array is to enclose the elements of the array inside curly braces {}, separated by commas

- This initializes the contents (values) of the array in the array declaration
- For example, this creates an array of ints with actual prime numbers
```
int[] primes = {2, 3, 5, 7, 11, 13, 19};
```
- This creates an array of Strings, with actual programming languages
```
String[] languages = {"Java", "C", "C++”};
```
- Imagine we have a Customer class. This creates an array of Customers, with actual customers
```
Customer[] customers = {new Customer(“Brandon”), new
Customer(“Betsy”)};
```
- The syntax above can *only* be used in the array declaration. You can’t do this:
```
int[] composites;
composites = {4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20}; //illegal
```

Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html

Length of An Array

Every array has a *length* variable, that tells how large the array is

- This array of ints has a length of 10

```
int[] scores = new int[10];
System.out.println(scores.length); //length of 10
```

Length of An Array

Every array has a *length* variable, that tells how large the array is

- This array of ints has a length of 10

```
int[] scores = new int[10];
System.out.println(scores.length); //length of 10
```

- And this array of Customers has a length of 2

```
Customer[] customers = {new Customer("Brandon"), new
Customer("Betsy")};
System.out.println(customers.length); //length of 2
```

Length of An Array

Every array has a *length* variable, that tells how large the array is

- This array of ints has a length of 10

```
int[] scores = new int[10];
System.out.println(scores.length); //length of 10
```

- And this array of Customers has a length of 2

```
Customer[] customers = {new Customer("Brandon"), new
Customer("Betsy")};
System.out.println(customers.length); //length of 2
```

- *length* is a variable, not a method
  - On the other hand, Strings have a *length()* method

Length of An Array

Every array has a *length* variable, that tells how large the array is

- This array of ints has a length of 10

```
int[] scores = new int[10];
System.out.println(scores.length); //length of 10
```

- And this array of Customers has a length of 2

```
Customer[] customers = {new Customer("Brandon"), new
Customer("Betsy")};
System.out.println(customers.length); //length of 2
```

- *length* is a variable, not a method
  - On the other hand, Strings have a *length()* method
- Arrays cannot be easily resized
  - You'd have to create a new array, copy everything from the old array, and add the new elements to the new array

Stepping Through an Array – *for* Loop

You can use a *for* loop to visit (and/or set) every element in an array, using its index

- Here we iterate over an array using its *length*, where i represents the index of each item

```
for (int i = 0; i < scores.length; i++) {
    System.out.println(scores[i]);
}
```

Stepping Through an Array – *for* Loop

You can use a *for* loop to visit (and/or set) every element in an array, using its index

- Here we iterate over an array using its *length*, where i represents the index of each item

```
for (int i = 0; i < scores.length; i++) {
    System.out.println(scores[i]);
}
```

- The name i is traditional for loops
  - i is instantly recognizable as the index of an enclosing for loop
  - Note, inner (nested) loops typically use j, then k

Stepping Through an Array – *for* Loop

You can use a *for* loop to visit (and/or set) every element in an array, using its index

- Here we iterate over an array using its *length*, where i represents the index of each item

```
for (int i = 0; i < scores.length; i++) {
    System.out.println(scores[i]);
}
```

- The name i is traditional for loops
  - i is instantly recognizable as the index of an enclosing for loop
  - Note, inner (nested) loops typically use j, then k
- Use of *length* is always preferred over using a constant (hard-coded) value (such as 10)
  - Try not to do this: for (int i = 0; i < 10; i++) { … }

Stepping Through an Array – *Enhanced for* Loop

You can also use an *enhanced for* loop to visit every element in an array

- For reference, this is like the *for x in list* syntax in Python
- Here we iterate over an array of Strings, where `n` represents each item in the array

```
for (String n : names) {
    System.out.println("Name: " + n);
}
```

Stepping Through an Array – *Enhanced for* Loop

You can also use an *enhanced for* loop to visit every element in an array

- For reference, this is like the *for x in list* syntax in Python
- Here we iterate over an array of Strings, where `n` represents each item in the array

```
for (String n : names) {
    System.out.println("Name: " + n);
}
```

- This simple structure allows you to visit each element of an array without explicitly expressing how to go from element to element using an index

Homework Scores

- Let's keep track of some homework scores in an *array*

HomeworkScores.java

```
1  /**
2   * Keep track of homework scores in an array.
3   * @author lbrandon
4   *
5   */
6  public class HomeworkScores {
7
8      public static void main(String[] args) {
9
10         //declare and initialize an array of doubles with 4 slots
11         double[] homeworkScores = new double[4];
12
13         //set each value in the array
14         homeworkScores[0] = 89.3;
15         homeworkScores[1] = 99.3;
16         homeworkScores[2] = 77.8;
17         homeworkScores[3] = 82.84;
18
```

Homework Scores

- Let's keep track of some homework scores in an *array*

```
18
19          //access/print the length of the array
20          System.out.println(homeworkScores.length);
21
22          //access/print the 2nd element in the array
23          System.out.println(homeworkScores[1]);
24
25          //access/print the 1st element in the array
26          System.out.println(homeworkScores[0]);
27
28          //update the first element in the array
29          homeworkScores[0] = 99.3;
30
31          //access/print the 1st element in the array again
32          System.out.println(homeworkScores[0]);
33
```

Homework Scores

- Let's keep track of some homework scores in an *array*

```
30
31          //use a for loop to visit (and/or set) every element in an array, using its index
32          for (int i = 0; i < homeworkScores.length; i++) {
33              System.out.println(homeworkScores[i]);
34          }
35
```

Homework Scores

- Let's keep track of some homework scores in an *array*

```
36          //another way
37          //use an enhanced for loop to visit each element of an array
38          //without explicitly expressing how to go from element to element using an index
39          for (double score : homeworkScores) {
40              System.out.println(score);
41          }
42      }
43  }
```

Find Even & Odd Numbers

- Let's iterate over an *array* and identify the even and odd numbers
- Also, let's get a count of the even and odd numbers by *incrementing* counts of each

EvenOrOdd.java

```
1  /**
2   * Identify even and odd numbers in an array.
3   * @author lbrandon
4   *
5   */
6  public class EvenOrOdd {
7
8      public static void main(String[] args) {
9
10         //declare an array of ints with 8 ints
11         int[] numbers = {1, 343, 98, 68574, 6382, 3948, 23456, 185};
12
13         //let's first update the first item in the array
14         numbers[0] = 0;
```

Find Even & Odd Numbers

- Let's iterate over an *array* and identify the even and odd numbers
- Also, let's get a count of the even and odd numbers by *incrementing* counts of each

```
15
16          //iterate over the array and identify the even and odd numbers
17          //and get a count of the even and odd numbers by incrementing counts of each
18          int evenCount = 0;
19          int oddCount = 0;
20
21          //enhanced for loop to visit each int in array
22          for (int number : numbers) {
23
24              //if it's even, print and increment evenCount
25              if (number % 2 == 0) {
26                  System.out.println(number + " is even");
27                  evenCount++;
28              //else (if it's odd), print and increment oddCount
29              } else {
30                  System.out.println(number + " is odd");
31                  oddCount++;
32              }
33          }
34
```

Find Even & Odd Numbers

- Let's iterate over an *array* and identify the even and odd numbers
- Also, let's get a count of the even and odd numbers by *incrementing* counts of each

```
34
35          System.out.print("There are " + evenCount + " even numbers, ");
36          System.out.println("and " + oddCount + " odd numbers in the array");
37      }
38  }
39
```

Find Minimum Value - Exercise

- Write code that finds the minimum value of an array of numbers 5, 3, 8, -1, -2, 0
  - Instantiate a 'numbers' array variable containing the proper values (above)
  - Iterate over that array and find the min value
  - Print the minimum value in the format: "... is the min number"

Find Minimum Value - Exercise

- Write code that finds the minimum value of an array of numbers 5, 3, 8, -1, -2, 0
  - Instantiate a 'numbers' array variable containing the proper values (above)
  - Iterate over that array and find the min value
  - Print the minimum value in the format: "... is the min number"

FindMinNumber.java

```
/**
 * Find the minimum value of an array of numbers.
 * @author lbrandon
 *
 */
public class FindMinNumber {

    public static void main(String[] args) {

        //declare and initialize int array with 6 ints
        int[] numbers = {5, 3, 8, -1, -2, 0};

        //set min to first int in array
        int minNumber = numbers[0];
```

Find Minimum Value - Exercise

- Write code that finds the minimum value of an array of numbers 5, 3, 8, -1, -2, 0

```
15
16          //iterate over array
17          for (int i = 0; i < numbers.length; i++) {
18
19              //if int is less than current min
20              if (numbers[i] < minNumber) {
21
22                  //reset min
23                  minNumber = numbers[i];
24              }
25          }
26
27          System.out.println(minNumber + " is the min number");
28      }
29  }
30
```

Iterating Over Array of Strings

- You can iterate over *arrays* of strings

IterateOverArrayOfStrings.java

```
1  /**
2   * Keep track of planets (as Strings) in an array.
3   * @author lbrandon
4   *
5   */
6  public class IterateOverArrayOfStrings {
7
8      public static void main(String[] args) {
9
10         //declare an array of Strings with 5 Strings
11         String[] planets = {"Sun", "Mercury", "Venus", "Earth", "Mars"};
12
```

Iterating Over Array of Strings

- You can iterate over *arrays* of strings

```
13          //iterate over the array
14          for (int i = 0; i < planets.length; i++) {
15
16              //get planet at index i
17              String planet = planets[i];
18
19              if (planet.equals("Sun")) {
20                  System.out.println(planet + " is not a planet");
21              } else {
22                  System.out.println(planet + " is a planet");
23
24                  if (planet.equals("Mercury")) {
25                      System.out.println(planet + " is closest to the Sun");
26                  }
27              }
28          }
29      }
30  }
31
```

Iterating Over a String

- A *string* is kind of like an *array* of characters
- You can *iterate* over a string using a *for* loop

IterateOverAString.java

```java
import java.util.Scanner;

/**
 * Demonstrates iterating over a string using a for loop.
 * @author lbrandon
 *
 */
public class IterateOverAString {

    public static void main(String[] args) {

        //declare and initialize a string
        String month = "February";
        System.out.println(month + " is spelled: ");

        //iterate over the string
        //for each index, from 0 to the length of the string (minus 1),
        //get/print each char (character)
        for (int u = 0; u < month.length(); u++) {
            System.out.println(month.charAt(u));
        }
        System.out.println();

```

Iterating Over a String - Exercise

- Prompt the user for their first name
- Using a *for* loop
  - Print each letter of the name (on the same line)
  - Count each letter in the name
- Print the count of letters in the name

Iterating Over a String - Exercise

- Prompt the user for their first name
- Using a *for* loop
  - Print each letter of the name (on the same line)
  - Count each letter in the name
- Print the count of letters in the name

```
23
24          //create a scanner
25          Scanner scan = new Scanner(System.in);
26
27          //prompt for first name
28          System.out.println("What is your first name? ");
29
30          //get input as a string
31          String name = scan.next();
32
```

Iterating Over a String - Exercise

```
32
33          //initialize count of letters
34          int letterCount = 0;
35
36          System.out.print(name + " is spelled: ");
37
38          //iterate over the name
39          //for each index 0 - length of the name (minus 1), get/print each char (character)
40          //and increment count of letters
41          for (int i = 0; i < name.length(); i++) {
42              System.out.print(name.charAt(i) + " ");
43              letterCount += 1;
44          }
45          System.out.println();
46
```

Iterating Over a String - Exercise

```
46
47          //print count of letters
48          System.out.println("There are " + letterCount + " letters in " + name);
49
50          //close the scanner
51          scan.close();
52
53      }
54  }
55
```