Suppose we want to copy the elements of an array to another array. We could try this code:

```java
double[] billsBackup;
billsBackup = cellBills; // incorrect!
```

Although this code compiles, it is logically incorrect! We are copying the `cellBills` object reference to the `billsBackup` object reference. We are not copying the array data.

The result of this code is shown on the next slide.

Example: this code copies the values of all elements in an array named `cellBills` to an array named `billsBackup`, both of which have previously been instantiated with the same length:

```java
double[] billsBackup = new double[cellBills.length];
for (int i = 0; i < cellBills.length; i++)
{
billsBackup[i] = cellBills[i];
}
```

To compare whether the elements of two arrays are equal:
1. Determine if both arrays have the same length.
2. Compare each element in the first array with the corresponding element in the second array.

To do this, we'll use a flag variable and a for loop.
Comparing `cellBills1` to `cellBills2`

```java
boolean isEqual = true;
if (cellBills1.length != cellBills2.length)
    isEqual = false; // sizes are different
else {
    for (int i = 0; i < cellBills1.length && isEqual; i++)
        if (Math.abs(cellBills1[i] - cellBills2[i]) > 0.001)
            isEqual = false; // elements are not equal
}
```

Using Arrays in Classes

- In a user-defined class, an array can be
  - an instance variable
  - a parameter to a method
  - a return value from a method
  - a local variable in a method

Methods with Array Parameters

- To define a method that takes an array as a parameter, use this syntax:
  ```java
  accessModifier returnType methodName(
      dataType [] arrayName)
  ```
- To pass an array as an argument when calling a method, use the array name without brackets:
  ```java
  = methodName(otherArrayName);
  ```
- To define a method that returns an array, use this syntax:
  ```java
  accessModifier dataType [] methodName(
      parameterList)
  ```

Arrays as Instance Variables

- Because arrays are objects, the name of an array is an object reference.
- Methods must be careful not to share references to instance variables with the client. Otherwise, the client could directly change the array elements using the reference to the array instance variable.

Accessors for Arrays

- Similarly, an accessor method for the array instance variable should return a copy of the array.
  ```java
  public double [] getCellBills( )
  {
      // instantiate temporary array
double [] temp = new double [cellBills.length];

      // copy instance variable values to temp
      for (int i = 0; i < cellBills.length; i++)
          temp[i] = cellBills[i];

      // return copy of array
      return temp;
  }
  ```

Software Engineering Tip

- Sharing array references with the client violates encapsulation.
- To accept values for an instance variable array as a parameter to a method, instantiate a new array and copy the elements of the parameter array to the new array.
- Similarly, to return an instance variable array, a method should copy the elements of the instance variable array to a temporary array and return a reference to the temporary array.