More on Arrays and Loops

- Reading for this Lecture:
  - Section 5.4, 6.3-6.4, 8.1-8.2
- Break and Continue in Loops
- Arrays and For-each Loops
- Arrays and Loops - Examples

Break and Continue in Loops

- The `break` statement causes execution to "break" out of the repetitive loop execution (goes to just outside the loop's closing "}")
- The `continue` statement causes execution to "continue" at the end of this pass of a loop (goes to just inside the loop's closing "}")
- Both are discouraged because an alternative way of coding the logic is usually available

Break and Continue in Loops

- Bad practice to use an infinite loop with only break statements to exit the loop:

```
while (true)
{
    if (normal exit condition)
        break;
    // body of loop
}
```

Break and Continue in Loops

- Accepted practice for a loop with a normal exit condition to use break statements for exiting the loop on error condition(s):

```
while (normal exit condition)
{
    if (some error condition) {
        // print an error message e.g.
        break;
    }
    // rest of body of loop
}
```

Break and Continue in Loops

- Not a good practice to use continue at all:

```
while (normal exit condition)
{
    if (condition1)
        continue;
    // rest of body of loop
}
```

- Use an if statement without continue as on the next slide

Break and Continue in Loops

- Use if alone rather than continue:

```
while (normal exit condition)
{
    if (condition2)
    {
        // rest of body of loop
    }
}
```

- Note: `condition2 == !condition1`

“for-each” with Arrays

- We can use “for-each” loops to access the elements in an array:
- Example Code:

```
boolean [] array = {true, false, true};
// for-each loop – note difference with for
for (boolean entry : array)
  System.out.println(entry);
```

- Example Run:

```
true
false
true
```

“for-each” with Arrays

- Note limitation of “for-each” versus “for”
- We can not initialize or update the element values in the array using a “for-each” loop

```
for(int num : nums)
   num = 5;  // doesn't update element
```

- We must use a regular “for” loop for that

```
for (int i = 0; i < nums.length; i++)
   nums[i] = 5;
```

Arrays and Loops - Examples

```
public class BasicArray
{
  public static void main (String[] args)
   {
      final int LIMIT = 15, MULTIPLE = 10;
      int[] list = new int[LIMIT];

      //  Initialize the array values
      for (int index = 0; index < LIMIT; index++)
         list[index] = index * MULTIPLE;

      list[5] = 999;  // change one array value

      //  Print the array values
      for (int value : list)
         System.out.print (value + "  ");
   }
}
> run BasicArray
0  10  20  30  40  999  60  70  80  90  100  110  120  130  140  >
```

Arrays and Loops - Examples

```
public class ArrayExample
{
  public static void main(String [] args)
  {
    char [] vowels = {'a', 'e', 'i', 'o', 'u'};
    int [] counts = new int[vowels.length];
    String s = "Now is the time for all good men to come to the aid of their country.";

    for (int i = 0; i < vowels.length; i++) {
      for (int j = 0; j < s.length(); j++)
        if (vowels[i] == s.charAt(j))
          counts[i]++;
    }
    for (int i = 0; i < vowels.length; i ++)
      System.out.println(vowels[i] + "\'s = " + counts[i]);
  }
}
> run ArrayExample
a's = 2
e's = 6
i's = 4
o's = 9
u's = 1
>
```