Topic 05: Arrays (advanced)

- Initializing arrays 6.1.1
- Filling arrays 6.1.3
- Other common algorithms 6.3.*
Initializing Arrays

- What does an array contain when you allocate it?

  Zeroes!

- Numeric Types
  - boolean
  - char
- References

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>(char)0</td>
<td>(char)0</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>
Two Different Syntaxes

```
int[] array = new int[10];
```

- Most common
- Fills with zeroes

```
int[] array = {1,2,3,4};
```

- Rare but useful
- Fills the array with the values given
- Length == number of items
- Only works for variable declarations
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2,3,5,7,11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2,3,5,7,11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

This line declares a variable, but does not allocate any array, or initialize the variable.
Array Initialization

```java
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2,3,5,7,11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}
```

This line allocates an array of length 10.

All of the elements are initialized to zero.
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2, 3, 5, 7, 11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

This line allocates an array of length 5.
The elements are initialized to the first 5 prime numbers.
Array Initialization

public static void dumpArray(int[] array) {
    System.out.print("{");
    for (int i=0; i<array.length; i++) {
        if (i != 0) {
            System.out.print(",");
        }
        System.out.print(array[i]);
    }
    System.out.println("}");
}

This prints out the contents of an array.
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2,3,5,7,11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

This code won't compile.
But why is that?
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2, 3, 5, 7, 11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

**Compile Output:**
$ javac ArrayInit1.java
ArrayInit1.java:9: error: variable foo might not have been initialized
dumpArray(foo);
  ^
1 error
Side Note: Uninitialized Variables

- Java checks for uninitialized variables
  - Won't allow you to use a variable that you might not have initialized

- Arrays are different!
  - Every element in the array is initialized, every time!

- Objects are different
  - Initialized the same way as arrays
public static void main(String[] args) {
    int[] foo;
    int[] bar = new int[10];
    int[] baz = {2,3,5,7,11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

This array reference is **uninitialized**. The value stored in the reference has never been set to anything.

Java will not allow you to use any uninitialized variable.
This array reference is **initialized**. It points at a particular object (an array).

The **elements** of the array are also initialized (to zero). This happens every time that an array is allocated.
public static void main(String[] args)
{
    int[] foo = null;
    int[] bar = new int[10];
    int[] baz = {2, 3, 5, 7, 11};

    dumpArray(foo);
    dumpArray(bar);
    dumpArray(baz);
}

We've fixed the code by initializing the array reference.

It is set to null, but null is a valid (and specific) value!
Topic 05: Arrays

- Initializing arrays 6.1.1
- Filling arrays 6.1.3
- Other common algorithms 6.3.*
Filling an Array

- To fill an array over time:
  - Allocate a big array
  - Keep a count of how many items are in use

- Useful for reading input (and lots of other things!)
import java.util.Scanner;

public class HasNext1
{
    public static void main(String[] args)
    {
        System.out.println("STARTING...");

        Scanner in = new Scanner(System.in);
        while (in.hasNext())
        {
            String word = in.next();
            System.out.println(word);
        }

        System.out.println("ENDED");
    }
}
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

This version of the code saves the values into an array, and prints them later.

Like PrimeFinder, it saves into an array; but the array may not be full.
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING... ");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

Constant:
MAX_WORDS can be changed to build a new program which will store more words.
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

Init:
We allocate an array. Each entry in the array will store a single String.
```java
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }

    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
```

**Counter:**

`wordCount` keeps track of how many words are stored in the array so far.
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;
System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }

    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

Pattern:
This pattern (index array by count, then increment count) is very common in programming!
Study it … and remember it for the future!
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");

Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");

Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

Error Handling:

Break out of the loop when an error occurs.

Note that we don't read all of the input...some is left behind.
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()) {
    if (wordCount == MAX_WORDS) {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()){
    if (wordCount == MAX_WORDS){
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()) {
    if (wordCount == MAX_WORDS) {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
```java
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;
System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()) {
    if (wordCount == MAX_WORDS) {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
```

**InputWords:**

- [0] = “asdf”
- [1] =
- [2] =
- [3] =

**wordCount:** 1
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;
System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

InputWords:
[0] = “asdf"
[1] =
[2] =
[3] =

wordCount: 1
final int MAX_WORDS = 1000;

String[] inputWords = new String[MAX_WORDS];

int wordCount = 0;

System.out.println("STARTING...");

Scanner in = new Scanner(System.in);

while (in.hasNext())
{
    if (wordCount == MAX_WORDS)
    {
        System.out.println("ERROR: Too many words!");
        break;
    }

    inputWords[wordCount] = in.next();
    wordCount++;
}

System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'" + inputWords[i] + "'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;
System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()) {
    if (wordCount == MAX_WORDS) {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");
for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");
final int MAX_WORDS = 1000;
String[] inputWords = new String[MAX_WORDS];
int wordCount = 0;

System.out.println("STARTING...");
Scanner in = new Scanner(System.in);
while (in.hasNext()) {
    if (wordCount == MAX_WORDS) {
        System.out.println("ERROR: Too many words!");
        break;
    }
    inputWords[wordCount] = in.next();
    wordCount++;
}
System.out.println("ENDED");

for (int i=0; i<wordCount; i++)
    System.out.println("'"+inputWords[i]+"'");

**InputWords:**
[0] = “asdf”
[1] = “jkl”
[2] =
[3] =

**wordCount:** 2
Topic 05: Arrays

- Initializing arrays 6.1.1
- Filling arrays 6.1.3
- Other common algorithms 6.3.*
public static int sum(int[] array) {
    int sum = 0;
    for (int val : array) {
        sum += val;
    }
    return sum;
}
public static int min(int[] array) {
    int min = array[0];

    for (int i=1; i<array.length; i++)
        if (array[i] < min)
            min = array[i];

    return min;
}
public static void printWithSpacer(int[] array, String spacer)
{
    System.out.print(array[0]);

    for (int i=1; i<array.length; i++)
        System.out.print(spacer+array[i]);

    System.out.println();
}
Linear Search (6.3.5)

```java
public static int find(int[] array, int val) {
    for (int i=0; i<array.length; i++)
        if (array[i] == val)
            return i;
    return -1;
}
```
```java
int[] array = ... ;

// we want to swap elements [x] and [y]
int temp = array[x];
array[x] = array[y];
array[y] = temp;
```
Swapping Two Elements (6.3.8)

```
int[] array = ... ;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
```
Swapping Two Elements (6.3.8)

```java
int[] array = ... ;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
```
Swapping Two Elements (6.3.8)

```java
int[] array = ...;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
```
Swapping Two Elements (6.3.8)

int[] array = ... ;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
Swapping Two Elements (6.3.8)

```java
int[] array = ...;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
```
Swapping Two Elements (6.3.8)

```java
int[] array = ...;

// we want to swap elements [x] and [y]

int temp = array[x];
array[x] = array[y];
array[y] = temp;
```

[x] 17 [y] 23
public static int[] delElement(int[] array, int delIndex) {
    int[] retval = new int[array.length-1];
    // this set of copies lines up perfectly
    for (int i=0; i<delIndex; i++)
        retval[i] = array[i];
    // this set is offset by one
    for (int i=delIndex+1; i<array.length; i++)
        retval[i-1] = array[i];
    return retval;
}

How would you write insert?
Deleting in Place

```c
int[] array = ... ;
int curLen = ... ;

// we want to delete element [x]
for (int i=x+1; i<curLen; i++)
    array[i-1] = array[i];

curLen--;```

Deleting in Place

int[] array = ... ;
int curLen = ... ;

// we want to delete element [x]

for (int i=x+1; i<curLen; i++)
    array[i-1] = array[i];

curLen--;
Deleting in Place

```c
int[] array = ...;
int curLen = ...;

// we want to delete element [x]
for (int i=x+1; i<curLen; i++)
    array[i-1] = array[i];

curLen--;```

```
12  2  -3   0  0  100  0  0
curLen: 6

[0]  [1]  [2]  [3]  [4]  [5]  [6]  [7]
i=4
```
Deleting in Place

```java
int[] array = ...;
int curLen = ...;

// we want to delete element [x]
for (int i=x+1; i<curLen; i++)
    array[i-1] = array[i];

curLen--;```

```
12  2  -3  0  100  100  0  0
[0]  [1]  [2]  [3]  [4]  [5]  [6]  [7]

i=5
curLen: 6
```
Deleting in Place

```java
int[] array = ...;
int curLen = ...;

// we want to delete element [x]
for (int i=x+1; i<curLen; i++)
    array[i-1] = array[i];

curLen--;```

```
12  2  -3  0  100 100  0  0
[0] [1] [2] [3] [4] [5] [6] [7]
```

curLen: 5
```java
int[] array = ...;
int curLen = ...

// we want to insert a hole at slot [x]
for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;```

We want to add a hole at [2]
Inserting in Place

```java
int[] array = ...;
int curLen = ...;

// we want to insert a hole at slot [x]
for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;
```

```
12  2  -3  0  100  100  0  0
[0]  [1]  [2]  [3]  [4]  [5]  [6]  [7]

curLen: 5
```
Inserting in Place

```java
int[] array = ...;
int curLen = ...;

// we want to insert a hole at slot [x]
for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;
```

```
12  2  -3  0  0  100  0  0   curLen: 5
[0] [1] [2] [3] [4] [5] [6] [7]
```
Inserting in Place

```java
int[] array = ...;
int curLen = ...;

// we want to insert a hole at slot [x]
for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;
```

![Diagram showing array elements before and after insertion of a hole.](55)
Inserting in Place

int[] array = ...;
int curLen = ...;

// we want to insert a hole at slot [x]

for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;

Update curLen last of all.
Inserting in Place

int[] array = ... ;
int curLen = ... ;

// we want to insert a hole at slot [x]
for (int i=curLen-1; i>=x; i--)
    array[i+1] = array[i];

curLen++;

12  2  -3  -3  0  100  0  0

We haven't (yet) filled this slot.
Topic 05: Arrays

- Initializing arrays 6.1.1
- Filling arrays 6.1.3
- Other common algorithms 6.3.*

Summary