Arrays
(Savitch, Chapter 7.1-7.2)

TOPOICS
• Array Basics
• Array Loops
• Array Programming

Arrays
• An array is a set of variables of the same type accessed by their index

```java
int[] day = new int[4];
```

- They are just accessed by their position
  - day[0], day[1], day[2], day[3]
  - Each integer has its own value
  - What happens with day[-1], day[4], day[1000]?
- Arrays can be of any type
  - int, double, char, boolean, String, class
  - Every element of an array has the same type

```java
31 28 31 30
```

day[0] day[1] day[2] day[3]

The length field (instance variable) of an array tells you how many elements it has
- day.length == 4.
Loops + arrays: challenge problem

• Task: read words from input until the word ‘quit’ appears. Then print out how many times each lowercase letter appeared.

• Question: where do you start? How do you approach this problem?

Step 1: Decomposition

• What has to be done?
• Initialize counters! – another loop
  – Read strings from terminal
  – For each string,
    • For each character,
      – Increment counter for that character
  – Print out counts per character

Steps 2-4:

• Tackle each step individually
  – Initialize counters
    • Declare the counters first
    • But there are 26 of them...
    • ... so use an array!

int[] alphaCounters = new int[26];

Step 2 (continued)

• Still initializing counters...
  – The counters count how often each letter appears
  – So we need to initialize all 26 of them to zero!
  – So we use a for loop with 26 iterations

  for(int i=0; i < 26; i++) {
    alphaCounters[i] = 0;
  }
Step 2 (yet again)

- So putting it together, we need to declare and initialize a counter for every letter in the alphabet:

```java
int[] alphaCounters = new int[26];
for(int i=0; i < 26; i++) {
    alphaCounters[i] = 0;
}
```

Does it work?

- Did we declare and initialize the counters correctly?
  - Never assume something is correct
  - Test each step before moving on
- How?
  - Method #1: Use the debugger
    - Run the code (so far) step-by-step
    - Check that the array has 26 elements
    - Check that each is set to zero
  - Method #2: Write the code to print counters
    - Test it before you start counting letters

Step 2 (one more time!)

- Actually, hard-coding '26' is a bad idea
  - What if we want to include capitals later?
  - What if we want to re-use this code?
- We want to initialize the array, and nothing more:

```java
int[] alphaCounters = new int[26];
for (int i=0; i < alphaCounters.length; i++) {
    alphaCounters[i] = 0;
}
```

Printing the Counters

- So printing the counters is another loop:
  - Why the trailing println()?

```java
for(int i=0; i < 26; i++) {
    System.out.print(alphaCounters[i]+",");
} System.out.println();
```
**Your First Test Code**

- What should this print?

```java
// Declare & Initialize Counters
int[] alphaCounters = new int[26];
for(int i=0; i < 26; i++) {
    alphaCounters[i] = 0;
}

// Print Counters
for(int i=0; i < 26; i++) {
    System.out.print(alphaCounters[i]+",");
}
System.out.println();
```

**Running Your First Test**

```java
public static void main(String[] args) {
    // Declare and initialize counters
    int[] alphaCounters = new int[26];
    for(int i=0; i < 26; i++) {
        alphaCounters[i] = 0;
    }
    // Print Counters
    for(int i=0; i < 26; i++) {
        System.out.print(alphaCounters[i]+",");
    }
    System.out.println();
}
```

**Reading Strings & Counting Letters**

- The next step has two parts:
  - Read strings (until "quit")
  - Increment letter counters

- Decompose the (sub)problem
  - Either read strings...
    - Assuming you can count letters once you have them
  - Or count letter instances
    - Given a string (assume you can get strings somehow)

*Don’t try to do two things at once!*

**One approach…**

- We’ve already written a loop to read until “quit”
  - Remember the Echo program?
- So let us start with that:

```java
Scanner in = new Scanner(System.in);
while (true) {
    String s = in.next();
    if (s.equals("quit")) break;
    System.out.println(s);  // Replace with counting
}
```
Wait! Stop! Test this First!

• How?
  – Use a print statement to make sure you are getting the right strings, or...
  – Use the debugger!

Counting Letters

• Given a string, we need to increment a counter for every letter.
  – Nested loop inside the get string loop

```
for(int i=0; i < s.length(); i++)
{
  char letter = s.charAt(i);
  OK, increment letter count here...
}
```

Counting Letters (II)

• Now increment the appropriate counter

```
for(int i=0; i < s.length(); i++)
{
  char letter = s.charAt(i);
  switch(letter) {
    case 'a': alphaCounters[0]++; break;
    case 'b': alphaCounters[1]++; break;
    case 'c': alphaCounters[2]++; break;
    // 23 more cases...
  }
```

Counting Letters (alt)

• The previous code is clean but long.
• Java supports ASCII char subtraction

```
char first_char = 'a';
char second_char = 'b';
int offset = second_char - first_char; // 1
```

```
for(int i=0; i < s.length(); i++)
{
  char letter = s.charAt(i);
  if ((offset >= 0) && (offset < 26)) {
    alphaCounters[offset]++;
  }
}
public static void main(String[] args) {
    // Declare and initialize counters
    int[] alphaCounters = new int[26];
    for(int i=0; i < 26; i++) {
        alphaCounters[i] = 0;
    }
    // Create scanner
    Scanner in = new Scanner(System.in);
    // Count letters
    while (true) {
        String s = in.next();
        if (s.equals("quit")) break;
        for(int i=0; i < s.length(); i++) {
            char letter = s.charAt(i);
            int offset = letter - 'a';
            if ((offset >= 0) && (offset < 26))
                alphaCounters[offset]++;
        }
    }
    // Close scanner
    in.close();
    // Print Counters
    for(int i=0; i < 26; i++) {
        System.out.print(alphaCounters[i]+",");
    }
    System.out.println();
}

• Test your program:
  – Make up some input
  – Count the letters by hand
  – Double check your program’s results
• Make Hard Tests
  – Include characters that aren’t letters
  – Test the case where the first input is ‘quit’
  – Test really long inputs
  – …
• Hint: it can be good to think of test cases first.
Methodology Review

1. Get a problem definition
   - Designing hard test cases can help refine the problem statements
2. Break the problem into pieces
   - Attack each piece separately
   - If a piece is big, break it up again
3. Test each piece before moving on
   - This may require temporary code
   - The debugger can help