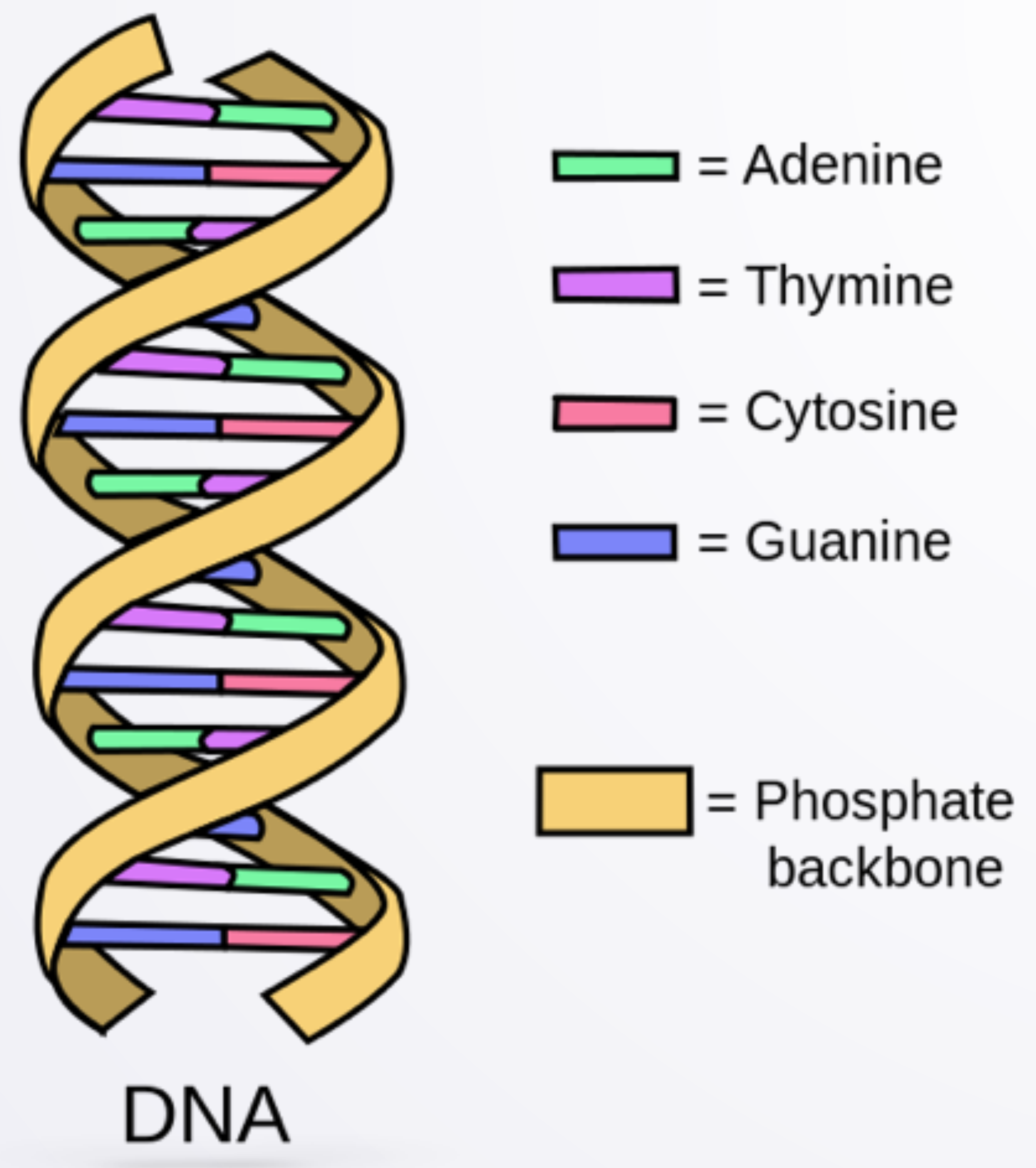
Breaking the Caesar Cipher

Arrays

Duke
UNIVERSITY

Overview of Problem

- As a genome computational scientist you:
  - Count c,g,a,t occurrences in DNA
  - Part of finding protein-coding regions



Overview of Problem

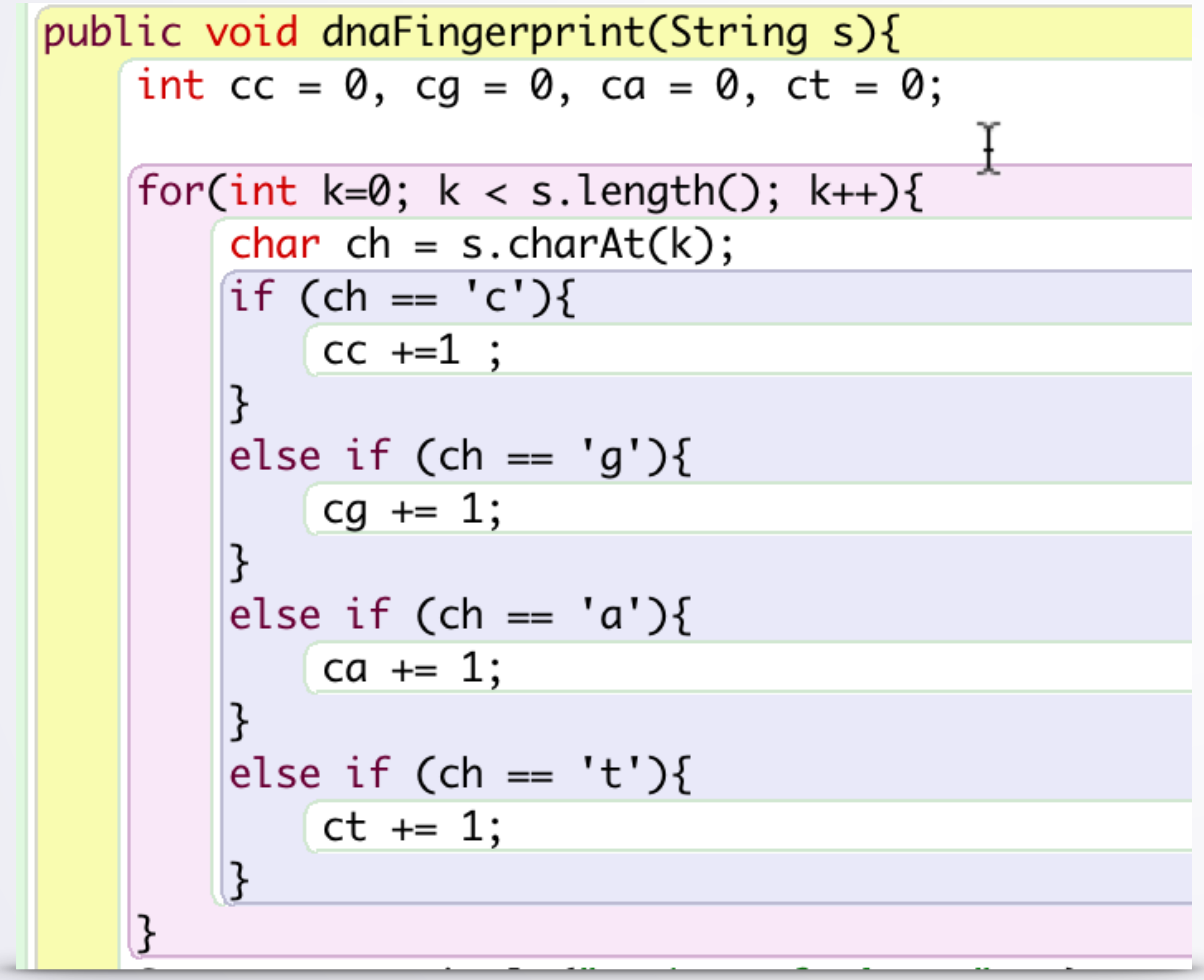
- As a genome computational scientist you:
    - Count c,g,a,t occurrences in DNA
    - Part of finding protein-coding regions
- As a programmer learning about encryption
    - Count occurrences of a,b,…,x,y,z
    - You need to break Caesar cipher

Overview of Problem

- Learn new programming concept
  - **Arrays**: homogeneous collection
  - Very important programming concept

Counting DNA Content

- Four different characters: ‘c’, ‘g’, ‘t’, ‘a’
  - Count occurrences of each one

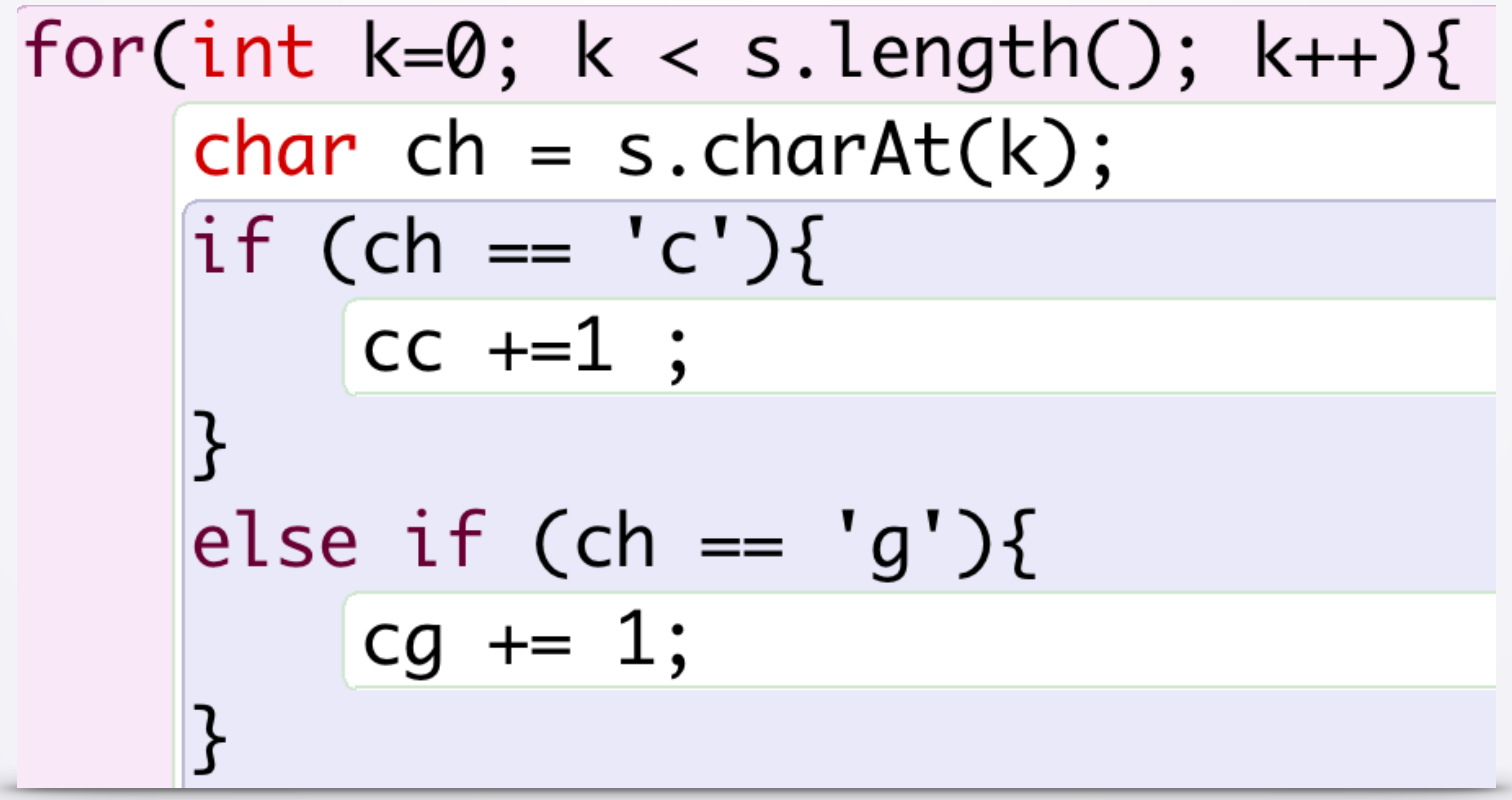


```
public void dnaFingerprint(String s){
    int cc = 0, cg = 0, ca = 0, ct = 0;

    for(int k=0; k < s.length(); k++){
        char ch = s.charAt(k);
        if (ch == 'c'){
            cc +=1 ;
        }
        else if (ch == 'g'){
            cg += 1;
        }
        else if (ch == 'a'){
            ca += 1;
        }
        else if (ch == 't'){
            ct += 1;
        }
    }
```

Counting DNA Content

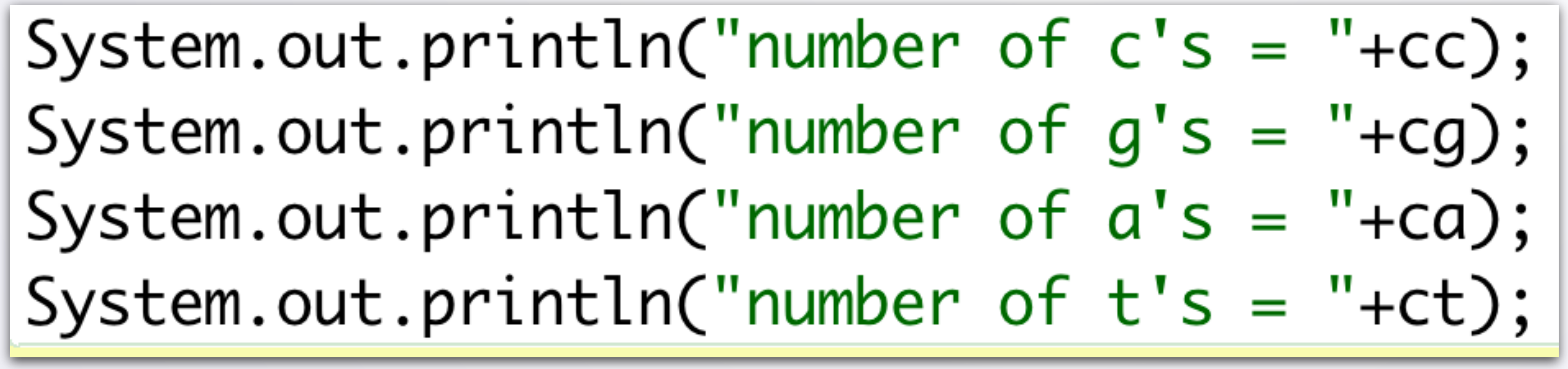
- Four different characters: ‘c’, ‘g’, ‘t’, ‘a’
  - Count occurrences of each one
- Hard to scale to ‘a’, ‘b’, ‘c’, …, ‘y’, ‘z’
  - Not conceptually hard, but brittle in the face of change



```
for(int k=0; k < s.length(); k++){
    char ch = s.charAt(k);
    if (ch == 'c'){
        cc +=1 ;
    }
    else if (ch == 'g'){
        cg += 1;
    }
```

Counting DNA Content

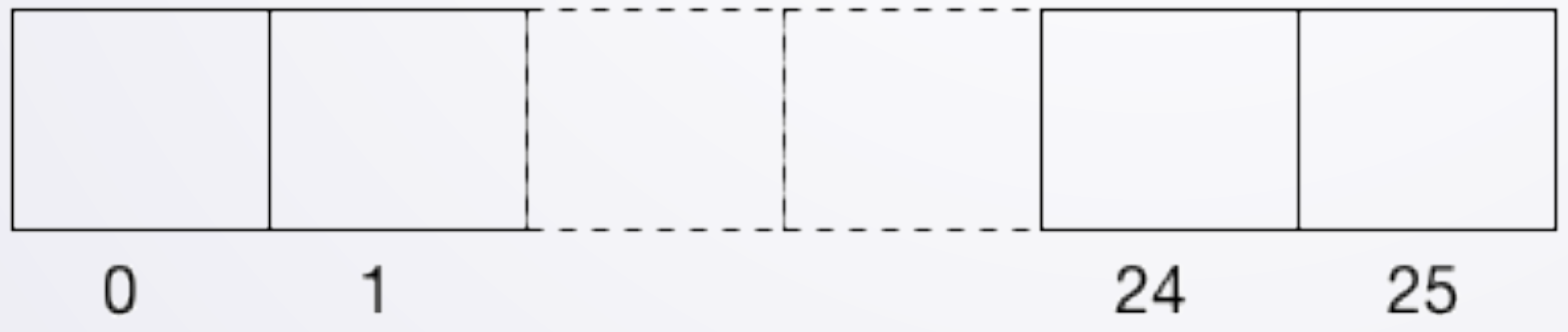
- Four different characters: ‘c’, ‘g’, ‘t’, ‘a’
  - Count occurrences of each one
- Hard to scale to ‘a’, ‘b’, ‘c’, …, ‘y’, ‘z’
  - Not conceptually hard, but brittle in the face of change
- What about printing results? Scaling?



```java
System.out.println("number of c's = "+cc);
System.out.println("number of g's = "+cg);
System.out.println("number of a's = "+ca);
System.out.println("number of t's = "+ct);
```

Array as an Indexed Collection

- To break Caesar cipher need letter counts
    - Most frequent character in English text: 'e'
    - In general counting and collecting important
- We have **StorageResource** to collect strings
    - Useful, but limited, will expand idea later
- We need an indexed collection
    - Like strings, but store anything, not just char



Concepts for Arrays

```
String s = "……";
for(int k=0; k < s.length(); k++){
    char ch = s.charAt(k);
}
```

```
int[] a = new int[256];
for(int k=0; k < a.length; k++){
    int val = a[k];
}
```

- Define array, similar to String, use [ ]

Duke
UNIVERSITY

Concepts for Arrays

```
String s = "......";
for(int k=0; k < s.length(); k++){
    char ch = s.charAt(k);
}
```

```
int[] a = new int[256];
for(int k=0; k < a.length; k++){
    int val = a[k];
}
```

- Define array, similar to String, use [ ]
  - Instead of using **`s.charAt(k)`** use **`a[k]`**

Duke
UNIVERSITY

Concepts for Arrays

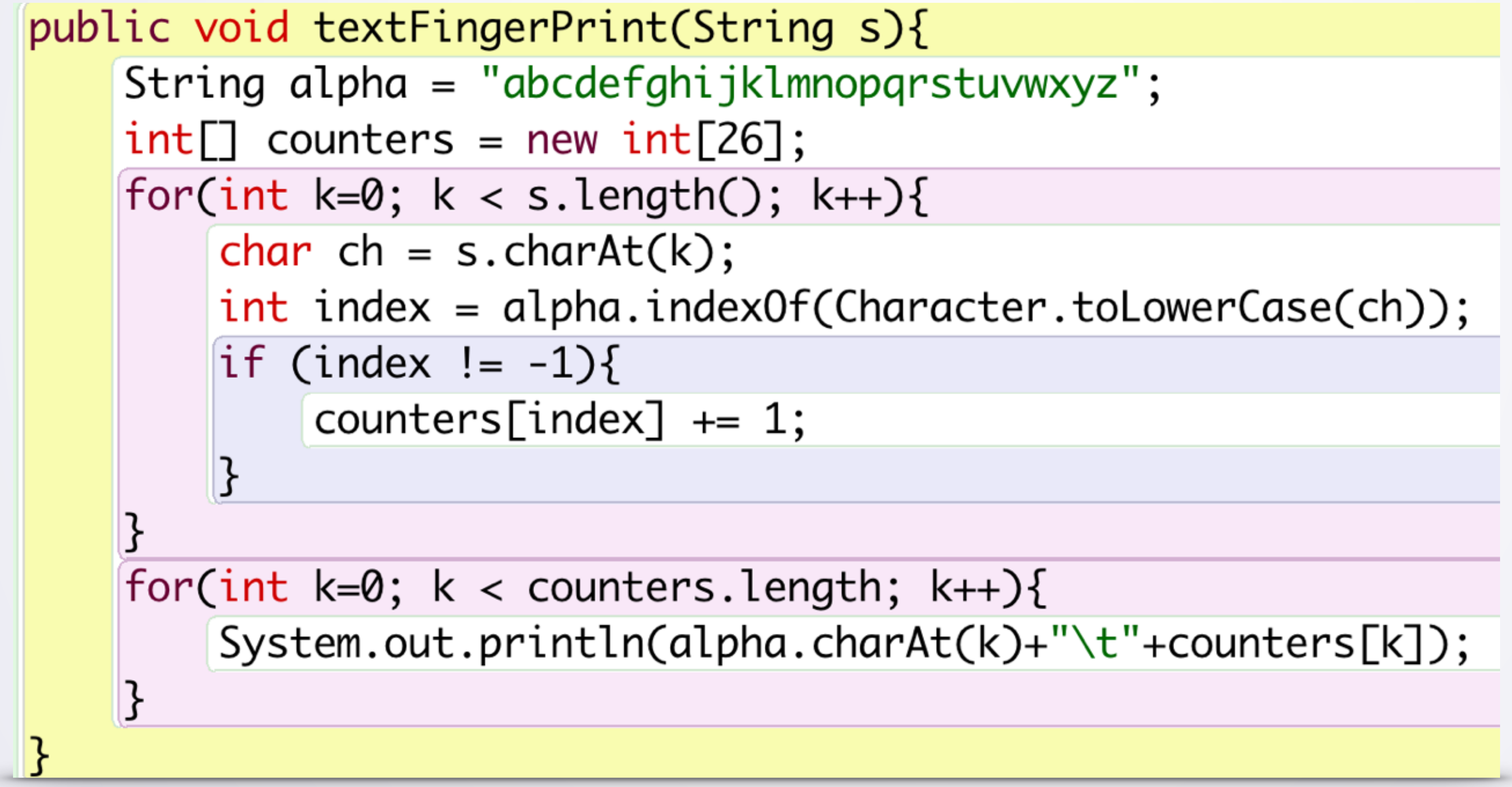
```
String s = "......";
for(int k=0; k < s.length(); k++){
    char ch = s.charAt(k);
}
```

```
int[] a = new int[256];
for(int k=0; k < a.length; k++){
    int val = a[k];
}
```

- Define array, similar to String, use [ ]
  - Instead of using **`s.charAt(k)`** use **`a[k]`**
  - Instead of using **`s.length()`** use **`a.length`**

Arrays in Action: Count 26 Characters

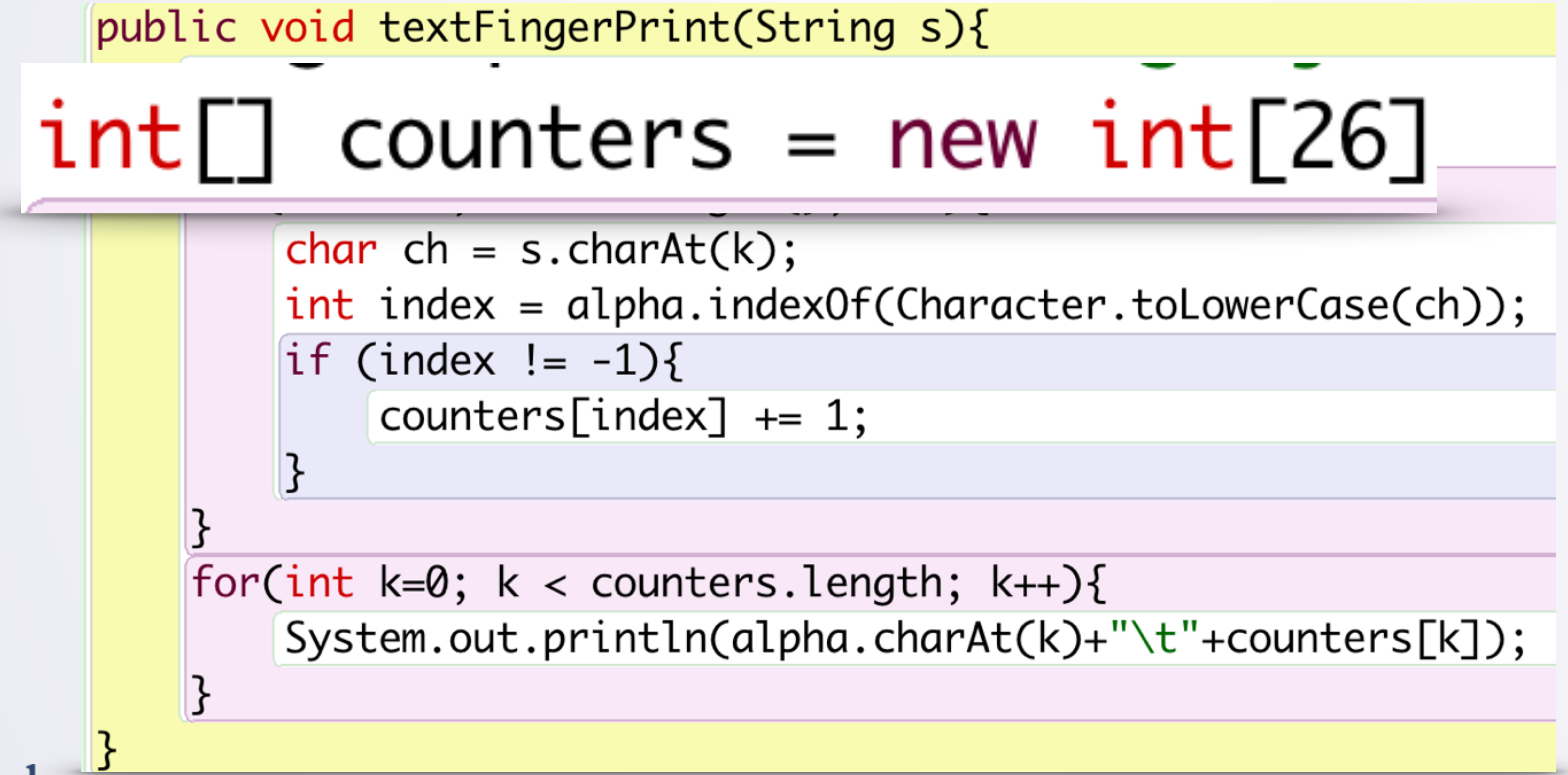
- Count: counters[0] is # of 'a' occurrences
    - counters[k] is # occurrences of kth letter (Z=25)



```java
public void textFingerPrint(String s){
    String alpha = "abcdefghijklmnopqrstuvwxyz";
    int[] counters = new int[26];
    for(int k=0; k < s.length(); k++){
        char ch = s.charAt(k);
        int index = alpha.indexOf(Character.toLowerCase(ch));
        if (index != -1){
            counters[index] += 1;
        }
    }
    for(int k=0; k < counters.length; k++){
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);
    }
}
```

Arrays in Action: Count 26 Characters

- Count: counters[0] is # of 'a' occurrences
    - counters[k] is # occurrences of kth letter (Z=25)

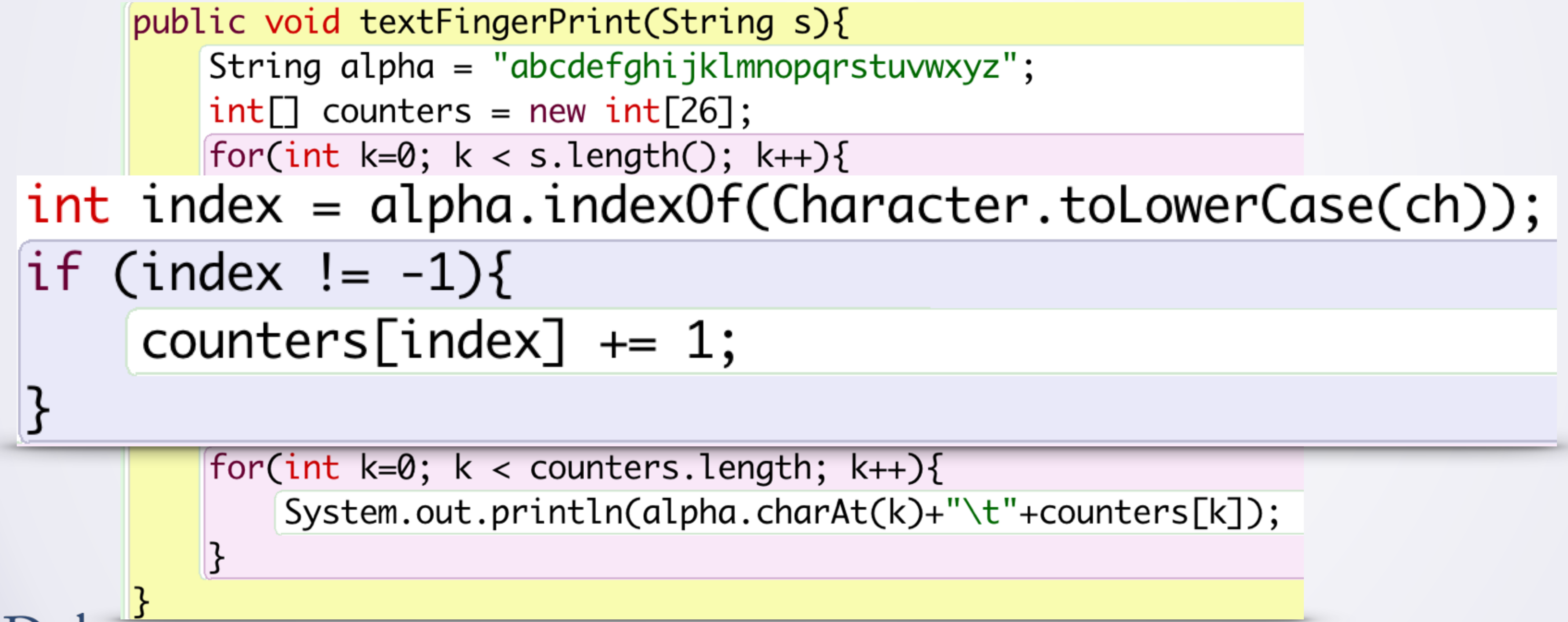


```
public void textFingerPrint(String s){
int[] counters = new int[26]
        char ch = s.charAt(k);
        int index = alpha.indexOf(Character.toLowerCase(ch));
        if (index != -1){
            counters[index] += 1;
        }
    }
    for(int k=0; k < counters.length; k++){
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);
    }
}
```

Duke UNIVERSITY

Arrays in Action: Count 26 Characters

- Count: counters[0] is # of 'a' occurrences
  - counters[k] is # occurrences of kth letter (Z=25)



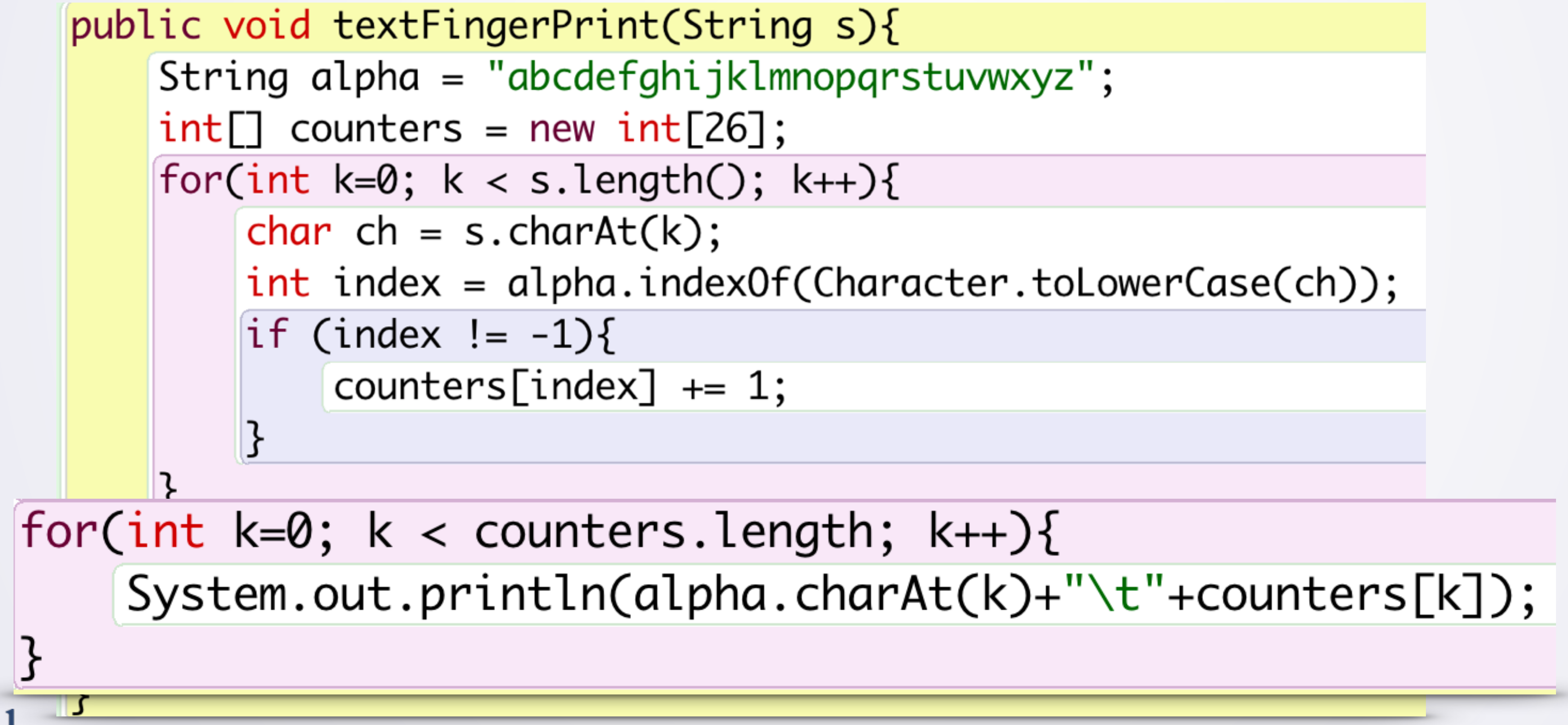
```
public void textFingerPrint(String s){
    String alpha = "abcdefghijklmnopqrstuvwxyz";
    int[] counters = new int[26];
    for(int k=0; k < s.length(); k++){
```

```
int index = alpha.indexOf(Character.toLowerCase(ch));
if (index != -1){
    counters[index] += 1;
}
```

```
    for(int k=0; k < counters.length; k++){
        System.out.println(alpha.charAt(k)+"\t"+counters[k]);
    }
}
```

Arrays in Action: Count 26 Characters

- Count: counters[0] is # of 'a' occurrences
  - counters[k] is # occurrences of kth letter (Z=25)



```
public void textFingerPrint(String s){
    String alpha = "abcdefghijklmnopqrstuvwxyz";
    int[] counters = new int[26];
    for(int k=0; k < s.length(); k++){
        char ch = s.charAt(k);
        int index = alpha.indexOf(Character.toLowerCase(ch));
        if (index != -1){
            counters[index] += 1;
        }
    }
```

```
for(int k=0; k < counters.length; k++){
    System.out.println(alpha.charAt(k)+"\t"+counters[k]);
}
```

```
}
```

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];         // zero
String[] s = new String[12]; // null
```

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];          // zero
String[] s = new String[12]; // null
```

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];          // zero
String[] s = new String[12];  // null
```

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];          // zero
String[] s = new String[12]; // null
```

- Read and write via indexes:
  - `s[3] = "Hello";` `x[2] = x[3] + 4;`

Duke UNIVERSITY

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];          // zero
String[] s = new String[12]; // null
```

- Read and write via indexes:
    - `s[3] = "Hello";` `x[2] = x[3] + 4;`

Duke UNIVERSITY

Array Summary

- Indexed collection of elements or values

```
int[] x; // no storage, just type
int[] x = new int[12];          // zero
String[] s = new String[12]; // null
```

- Read and write via indexes:
  - `s[3] = "Hello";  x[2] = x[3] + 4;`
- Storage allocated, then doesn't change
  - Why **`.length`** is a value, not a method
  - Can modify elements via method calls