Tuesday — Programming examples / Inverse Manhattan problem, using binary search; WordFrequency, using StringTokenizer. Discussed: method, constructor overloading; and signatures. Also discussed: the “sequence of objects” model of computation, as in Diffy, Opoly (sort of), and numerical palindrome examples.

```java
public class Cheese{
    private String name = "Velvetta";
    private int amount; // in grams

    public Cheese(String which, int amt){
        name = which;
        amount = amt;
    }
    public Cheese(int amt){amount = amt;}

    public String getName(){return name;}
    public int getAmount(){return amount;}
    public void setAmount(int amt){amount = amt;}
    public void setName(String name) {this.name = name;}
}
```

Write a statement that makes a 100 gram Velvetta hunk of cheese. Now make a 200 gram hunk of Limburger cheese.

Write method eatSome, which is passed an array of Cheese objects and a nibbling fraction (e.g. .15), then reduces amount of each hunk by that fraction.

NumPal example was discussed in class. Driver is below.

```java
import javax.swing.JOptionPane;
public class NumPalDriver{
    public static void main(String[] args){
        String start = JOptionPane.showInputDialog("enter positive start value");
        NumPal p = new NumPal(start);
        int ctr = 0;
        while (!p.pal() && (ctr < 10)){
            p.printNumPal();
            System.out.print("new sum: ");
            System.out.println(p.getCur() + p.getRev());
            p = p.next();
            ctr++;  }
        System.out.println("final value: "+ p.getCur());
        System.out.println("number of steps: "+ ctr);  }
```
Problems:

Write a method that’s passed an array of Strings, determines if they’re in ascending order, by length.

Write a method that’s passed an array of ints, reports the size of the largest gap between any two entries.

StringTokenizer. Lines **, ***, below, are alternatives. What’s the difference? How does each handle this phrase: “the quick, near-brown fox: it jumped!!”

```java
import java.util.StringTokenizer;
import java.util.Scanner;
public class TokenizerTest{
    public static void main(String[] args){
        StringTokenizer str;
        Scanner scan = new Scanner(System.in);
        System.out.println("enter a line of text");
        String s = scan.nextLine();
        str = new StringTokenizer(s); // ** or
        str =  new StringTokenizer(s," .,!?;:"); // ***
        while (str.hasMoreTokens()){
            System.out.println(str.nextToken());
        }
    }
}
```

Write a complete program in a single class that reads in a single string of A’s, B’s only, reports the number of blocks of B’s it sees.
Example: “AABBBABBAAAABAB” -> 4

Write a complete program (edit TokenizerTest) that reads in a single string, and then prints the number of words in the string that are strictly longer than 5 letters.

How would you use an array to examine a String, s, and determine if all letters a through z appear in s?

**Ans:** make a Boolean array, initialized to all false. Then loop through the array, setting cells to true as letters arise. All true, then, means all letters have appeared.