Objectives: StringBuilder; algorithm analysis;
Up next: Quiz 10 Now!; MP7; Movie Rating App bonus points;

2. **StringBuilder vs. String**

Advantages:

Disadvantages:

3. **StringBuilder vs. String: appending** - the race is on…

Write a procedural program (remember those?) to compare the speed of appending a character to a String (s = s + "!") to a StringBuilder object (s.append("!")).

`System.currentTimeMillis()` might be handy!
From Java Documentation:

```
public static long currentTimeMillis()
```

Returns the current time in milliseconds represented as the difference (measured in milliseconds) between the current time and midnight, January 1, 1970 UTC.

1. You have an array of doubles. You want to search between indices 'lo' and 'hi'. Write a recursive method to find the largest product of two neighboring values. e.g. `findPair({ 1.0 , 1.0 , 7.5 , 4.0, 4.1 , 3.5 },0,5)` returns 30.0 (7.5 * 4.0), which is largest product of two neighboring values.

```
public static double findPair(double[] array, int lo, int hi) {
```

Write a FORWARD recursive method to find the first index of the largest product of two neighboring values. e.g. `findPair({ 1.0 , 1.0 , 7.5 , 4.0, 4.1 , 3.5 },0,5)` returns 2 because 7.5x4.0=30.0 is largest product of two neighboring values.

```
public static int findPair(double[] array, int lo, int hi) {
```

4. ***Work with a neighbor*:** How many seconds will each problem take?
It takes 5s to ask each person a question and get a response, 10s for two people to swap positions. All participants are sitting in one row.

a. Find the first person carrying one $20, two $10 and three $5 bills.
Variables used: currentindex (perhaps: array, max index)
Dataset size 10 20 40
Best case :
Worst case :

b. Find the smallest amount of change carried by one person.
Variables used: ?

Dataset size 10 20 40
Best case :
Worst case :

c. Find the largest total amount of change carried by two neighboring people in a row.
Variables used: ?

Dataset size 10 20 40
Best case :
Worst case :

d. Partition (divide) the group of people into two subgroups: "more obnoxious than Kardashian" group and less-virtuous-than-Kardashian group.
Variables used: ?

Dataset size 10 20 40
Best case :
Worst case :

Moore's Law: a prediction made in 1965 by Intel co-founder Gordon Moore that the density of transistors in integrated circuits would continue to double every 1 to 2 years. Even more remarkable – and even less widely understood – is that in many areas, performance gains due to improvements in algorithms have vastly exceeded even the dramatic performance gains due to increased processor speed.

5. How long will the following algorithm take to run as N get's large?
Best case? Worst Case? Assume data.length > N

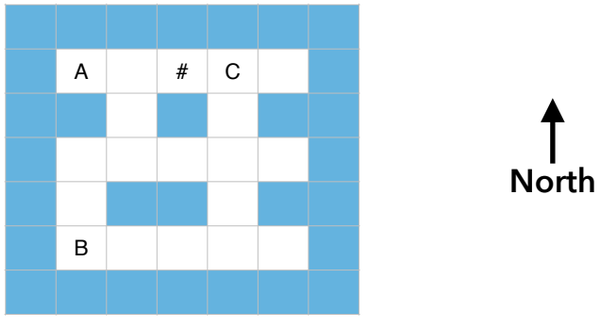
```
public static void bar(double[] data, int N) {
  int i= 2000;

  while( i <= N ){
    if( data[i]  == 42 )
      return;

    i += 2;
  }
}
```

6. Write an expression for the ***worst-case*** running time of each algorithm. t(N) = ….
Define any constants you need.

```
public static boolean foo1(int N) {    t(N)= ?
  int i = N * 2;

  return (i*i + N*N )>1000;

}


public static void foo2(int N) {
  int i= 2000;

  while( i <= N ){

    i += 2;

  }
}
```

6. Searching Mazes: recursive algorithm

From starting points A,B,C, which (x,y) positions will be checked by the code below?



(0,0) is bottom-left of the maze.

Pseudo-code

```
String explore(x,y,flagX,flagY,wall,…          ) {
    if(wall[x][y]) return null; // No path here
    if(x == flagX && y == flagY) return  ""; // Found


     String goNorth = explore(    ,     , …)   ??
     String goEast  = explore(    ,     , …)   ??
     String goSouth = explore(    ,     , …)   ??
     String goWest  = explore(    ,     , …)   ??
// There are more elegant implementations

    if(goNorth != null) goNorth = "N" + goNorth;
    if(goEast != null)  goEast  = "E" + goEast ;
    if(goSouth != null) goSouth = "S" + goSouth;
    if(goWest != null)  goWest  = "W" + goWest ;


if(goNorth == null && goEast  == null
   && goSouth == null && goWest==null ) return null;
// No path ☹

    String shortestViablePath = … pick shortest non-null path
    return shortestViablePath;
}
```

7. You have an array of doubles. You want to search between indices 'lo' and 'hi'. Write a recursive method to find the largest product of two neighboring values. e.g. `findPair({ 1.0 , 1.0 , 7.5 , 4.0, 4.1 , 3.5 },0,5)` returns 30.0 (7.5 * 4.0), which is largest product of two neighboring values.

```
public static double findPair(double[] array, int lo,
                                                int hi) {
```

Write a FORWARD recursive method to find the first index of the largest product of two neighboring values. e.g. `findPair({ 1.0 , 1.0 , 7.5 , 4.0, 4.1 , 3.5 },0,5)` returns 2 because 7.5x4.0=30.0 is largest product of two neighboring values.

```
public static int findPair(double[] array, int lo, int hi)
{
```