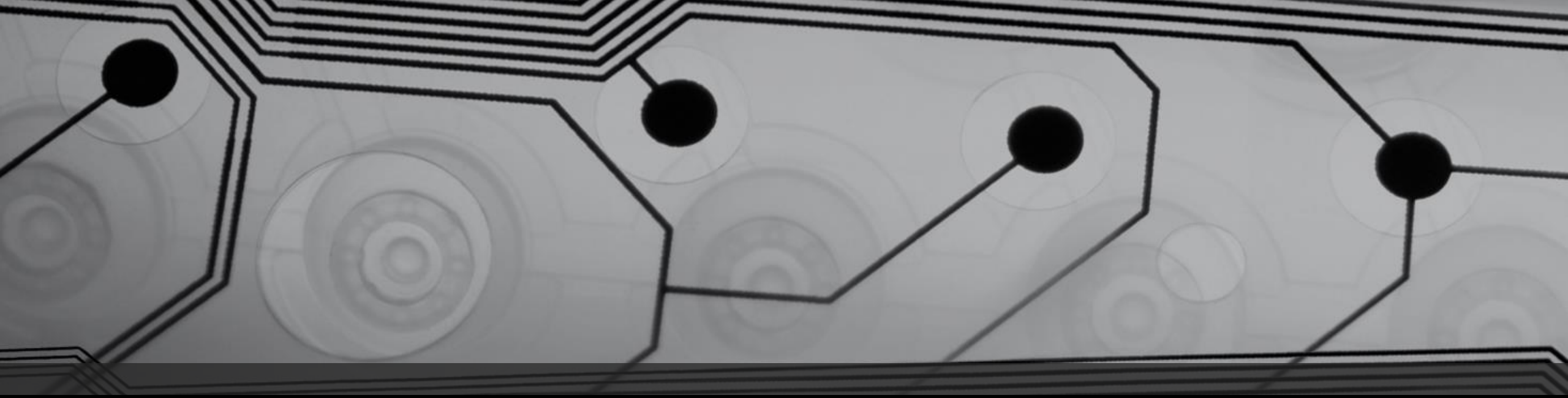

CMSC 131

Fall 2018

Announcements

- Median exam score was 75
- ADS exams are with me (see me after class)
- Project #6 is due Thursday
  - Regarding equals for the Entree class… Do it the old (wrong) way:

    public boolean equals(Entree x) {…}

For-Each Loops

Assume x is a collection of Cat references
(Could be an array, ArrayList, any Java Collection…)

Instead of these:

```
for (i = 0; i < a.length; i++) {
   … a[i] …
}
```

```
for (i = 0; i < x.size(); i++) {
   … c.get(i)…
}
```

Use this:

```
for (Cat c : x) {
    … c …
}
```

Example: ForEachExample.java, ForEachWithTwoDimensionalArray.java

Limitations of For-Each Loops

- Can’t Loop through part of the list
- No index number
- Example: DoesNotWorkWithForEachLoop.java
- You cannot add or remove elements from the current list during the loop
  Example: DoNotTryRemoveInsideForEachLoop

Choosing Algorithms

- What is an algorithm?

- What factors do we consider when choosing?
  - Familiar design
  - Ease of coding
  - Efficiency
    - Runtime
    - Memory

Algorithmic Complexity

- Fact: The more data we process the longer it takes
- Question: Is the processing time always *proportional* to the size of the data set?
- How can we classify algorithms with regard to this?
- Let’s draw some graphs!

Linear Search

- Demonstration: Linear Search
- What shape will the runtime graph be?
- Roughly speaking, what happens to the runtime as the size of the dataset doubles?
- We classify *all* algorithms with this shape as “linear” algorithms.