Garbage Collection
CSCI 201L

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu
HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

- Garbage Collection
Garbage Collection

- Java doesn’t require much memory management
  - C++ has `malloc`, `free`, `delete`, etc.
- Since Java does not have these key words, another mechanism was needed to clean up memory after it was used
  - When a variable goes out of scope, the Garbage Collector will come around and delete that reference in memory
- Garbage collection looks at the heap and identifies which objects are in use and which are not
  - An object is a candidate for garbage collection when it no longer is referenced by any part of your program
When are the following variables candidates for garbage collection?

```java
public class Garbage {
    private int num;
    private static int val;
    public Garbage(int num) {
        num++;
        this.num = num;
        val = 10;
    }
    public static void meth() {
        Garbage g = new Garbage(number);
        g.num = 10;
        Garbage.val = 11;
    }
    public static void main(String [] args) {
        int number = 3;
        for (int i=0; i < number; i++) {
            number++;
        }
        meth();
        System.out.println(number);
    }
}
```
Garbage Collection Steps

- Step 1 – Marking
  - The garbage collector determines which pieces of memory are in use and which are not

Garbage Collection Steps

- **Step 2 – Normal Deletion**
  - Garbage collector removes unreferenced objects leaving referenced objects and pointers to free space

Garbage Collection Steps

- Step 2a – Deletion with Compacting
  - Garbage collector deletes unreferenced objects and compacts the remaining referenced objects

Invoking Garbage Collection

- Programmers are not able to explicitly invoke the garbage collector
- We can make a suggestion to the JVM to run the garbage collector with
  ```java
  System.gc();
  ```
- This does not mean that the garbage collector will be run immediately though
- More garbage collection information can be found at