CSC 207
Algorithms and Object-Oriented Design

Weiss, from the beginning of chapter 3 through section 3.2 and sections 3.4 through 3.7 (pages 69–73 and pages 76–90)
References

• The type of data is either primitive or reference.

• A reference variable stores a memory address. In that memory address, an object is stored.
  ➢ A reference stores null when it is not referencing any object.

• Memory contains
  – Single values for primitive types.
  – Multiple values for reference types (composite types)
References (cont.)

Reference variables can be set to null.
Reference variables can be manipulated by the assignment operator.
point3 = point2;

Reference variables can be compared to each other by (in)equality operator.
point2 == point3;
point1 != point2;

The result of point3=point2: point3 now references the same object as point2.
Object

- Object is an instance of a class (blueprint).

```java
Point originOne = new Point(23, 94);

Point originOne: Declaration (it does not allocate memory)
new: Instantiation (creates an object in memory and returns and address
Initialization: Point(23, 94);
```

```java
public class Point {
    private int x;
    private int y;

    public Point(int a, int b) {
        x = a; y = b;
    }

    public void move(int c, int d) {
        x = x + c;
        y = y + d;
    }
}
```
The dot Operator

• Objects are manipulated when calling a method.

    originOne.move(23, 94);

• Fields can also be accessed by the dot operator, but it is not recommended to directly access fields.
The Garbage Collector

• Objects can be destroyed when they are not needed.

• An object is not needed when it is not referenced by a reference variable.

• For example, a reference variable goes out of scope, or the variable is set to null.

• Java runtime environment finds out unreferenced objects. It destroys those objects with garbage collector.
Objects Comparison

• `==` operator compares the memory addresses of referenced objects.

```java
Point point1 = new Point(0,0);
Point point2 = piont1;
if (point1 == point2) {
    ...
}
```

• `equals()` compares the contents of objects. For two instantiated objects with identical contents, `equals()` need to be re-defined.

```java
Point point3 = new Point(0,0);
if (point1.equals(point3)) {
    ...
}
```