CS 213 : Software Methodology

Spring 2017

Sesh Venugopal

Lecture 6: Feb 2
Inheritance: Object Class/equals method – Part 1

Object Class

- Root of java class hierarchy
  - Every class ultimately is a subclass of `java.lang.Object`
- Methods in `Object` you have seen – all of these are inherited by ANY class (since every class is implicitly a subclass of `Object`):
  - `equals`: compares address of objects
  - `toString`: returns address of object
  - `hashCode`: returns hash code value for object
- Must generally override `equals` and `toString`

Writing code banking on equals being there

```
public class Searcher {
   public static <T> boolean
   sequentialSearch(T[] list, T target) {
      for (int i=0; i < list.length; i++) {
          if (target.equals(list[i])) {
             return true;
          }
          return false;
      }
   }
}
```

Don't know what T will be at runtime, but it is guaranteed to have the `equals` method

- Because the `Object` class defines equals, you—as an algorithm designer—can *independently* write code to compare two objects using the `equals` method, and the code will compile (And when an application sends in, say, `Point` objects, the overridden equals will be called)

Overriding `equals`

Boiler-plate way to override equals (e.g. `Point`):

```
public class Point {
   int x,y;
   ...
   public boolean equals(Object o) {
      if (o == null || !(o instanceof Point)) {
         return false;
      }
      Point other = (Point)o;

      return x == other.x && y == other.y;
   }
   ...
}
```

1 Header must be same as in `Object` class

2 Check if actual object (runtime) is of type `Point`, or a subclass of `Point`

3 Must cast to `Point` type before referring to fields of `Point`

4 Last part is to implement equality as appropriate (here, if x and y coordinates are equal)

Overriding `equals`

```
public class Point {
   int x,y;
   . . .
   public boolean equals(Object o) {
      if (o == null || !(o instanceof Point)) { return false; }
      Point other = (Point)o;
      return x == other.x && y == other.y
   }
}
```

Calling the `Point equals` method

```
Point p = new Point(3,4);                    p.equals(p); // ?  True

Point cp =
   new ColoredPoint(3,4,"black");            p.equals(cp); // ? True

String s = "(3,4)";                          p.equals(s); // ?  False

Point p2 = new Point(4,5);                   p.equals(p2); // ? False
```