CS 213 : Software Methodology

Spring 2019

Sesh Venugopal

Lecture 6: Feb 7
Inheritance: Object Class/equals method - 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`

Method Overloading/Overriding

Method **OVERLOADING**:

Two methods in a class have the same name but different numbers, types, or sequences of parameters

```
class Test {
   int m(int x) {…}
   int m(float y) {…}
}
```

Overloaded method `m`

```
class Test {
   int m(int x) {…}
   float m(float y) {…}
}
```

Overloaded method `m`

```
class Test {
   int m(int x) {…}
   float m(int y) {…}
}
```

Error

Two or more methods in a class are **overloaded** if they have the same name but different signatures

signature = name + params (return type NOT included in signature)

Method **OVERRIDING**:

A method in a subclass has the same signature as in the superclass

Writing library 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, `equals` will be called on `Point` – either the one inherited from `Object`, or – hopefully – the overriding one.)

Implementing `equals` — Rookie Version

Implementing `equals` — Rookie Version

Rookie attempt to implement `equals` (e.g. in `Point`):

```
public boolean equals(Point p) {
    return x == p.x && y == p.y;
}
```

```
Point p = new Point(3,4);
```

```
p.equals(p); // ? True
```

```
Point cp =
   new ColoredPoint(3,4,"black");
```

```
p.equals(cp); // ? True
```

Ok, inherited `equals(Point p)` in `ColoredPoint` is called

Dynamic type `ColoredPoint` argument at run time matches static type `Point` parameter

---

```
Point p2 = new Point(4,5);
```

```
p.equals(p2); // ? False
```

```
String s = "(3,4)";
```

```
p.equals(s); // ? FALSE!!
```

The inherited `Object equals(Object o)` is called!!!
Otherwise, this should give a compiler error

`equals(Point p)` does NOT override `Object equals(Object o)`

Implementing `equals` — Rookie Version

Rookie attempt to implement `equals` (e.g. in `Point`):

```
public boolean equals(Point p) {
    return x == p.x && y == p.y;
}
```

```
Point p = new Point(3,4);

ColoredPoint cp =
   new ColoredPoint(3,4,"black");
```

`cp.equals(p); // ?` True

Ok, inherited `equals(Point p)` in `ColoredPoint` is called

```
Object op = new Point(3,4);
```

`p.equals(op); // ?` FALSE!!

The inherited `Object equals(Object o)` is called!!!
Because the STATIC type of parameter is `Object`, which matches the `Object` parameter type of inherited `equals`

Moral of the story: You `MUST override Object equals(Object o)`

Implementing `equals` — Grad Version

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)

Single Version: 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

`equals` overload + override (both versions present)

```
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
   }

   public boolean equals(Point p) {
     if (p == null) {
        return false;
     }
     return x == p.x && y == p.y
   }
}
```

With the following setup:

```
Point p = new Point(3,4);

Object o = new Object();

Object op = new Point(3,4);
```

Which method is called in each case, and what's the result of the call?:

```
p.equals(p); // ? True

p.equals(o); // ? False

p.equals(op); // ? True
```