

Equality and Inheritance

Dec 11

Teaching evaluations are used for three purposes:

- 1) they provide feedback that should be helpful to the course instructor in general, or when this course is taught again;
- 2) they are reviewed by various faculty members in the evaluation of teaching when the College considers an instructor for reappointment, tenure, or promotion;
- 3) they provide information that helps the College evaluate larger curricular initiatives and goals.

Both the instructor and the College appreciate thoughtful, candid, comments whether positive or negative, and constructive suggestions for improvement.

Please spend 20 minutes providing responses to the questions on this form. Your answers will be strictly confidential. The instructor will see the results only after course grades have been submitted to the Registrar.

Classes can have instances of other classes

```
public class Flight {  
    private GTDate theDate;  
    private String flightNumber;  
    private int departDelay;  
    private int arriveDelay;  
    private String originCity;  
    private String destCity;  
    public String getDate() {  
        return theDate.toString();  
    }  
}
```

What about the constructor?

Other Usage?

Classes can have instances of other classes

```
public class Flight {  
    private int year;  
    private int month;  
    private int date;  
    private String flightNumber;  
    private int departDelay;  
    private int arriveDelay;  
    private String originCity;  
    private String destCity;  
    public String getDate() {  
        return year + "/" + month +  
        "/" + date;  
    }  
}
```

For instance, replace this with in instance of

```
public class GTDate {  
    int month;  
    int day;  
    int year;  
  
    public GTDate(int mt, int dt, int yr) {  
        month = mt;  
        day = dt;  
        year = yr;  
    }  
  
    // Override  
    public String toString() {  
        return month + "/" + day + "/" + year;  
    }  
  
    //Override  
    public boolean equals(Object ob) { ... }  
}
```

???

Overriding .equals

!!!!

- Every class / object has a .equals method
 - default is same as ==
- Like toString, you can write your own

```
@Override
public boolean equals(Object ob) {
    if (ob instanceof String) {
        String oString = (String) ob;
        //return oString.equals(toString()); // works but fragile
        // break the string up into components
        String[] ss = oString.split("/");
        if (ss.length == 3) {
            try {
                int d=Integer.parseInt(ss[0]);
                int m = Integer.parseInt(ss[1]);
                int y = Integer.parseInt(ss[2]);
                return y==year && m==month && d==day;
            } catch (Exception ee) {
                return false;
            }
        } else {
            return false;
        }
    }
    if (ob instanceof GTDate) {
        GTDate gtd = (GTDate) ob;
        return month == gtd.month && day == gtd.day && year == gtd.year;
    }
    return super.equals(ob);
}
```

???


Overriding AND Overloading

- You can do BOTH!
- Move those awkward "instanceof" out from equals
- make language deal with multiple types

```
@Override
public boolean equals(Object ob) {
    System.out.println("EqualsObject");
    return super.equals(ob);
}

public boolean equals(String oString) {
    System.out.println("Equals String");
    String[] ss = oString.split("/");
    if (ss.length == 3) {
        try {
            int d = Integer.parseInt(ss[0]);
            int m = Integer.parseInt(ss[1]);
            int y = Integer.parseInt(ss[2]);
            return y == year && m == month && d == day;
        } catch (Exception ee) {
            return false;
        }
    } else {
        return false;
    }
}

public boolean equals(GTDate other) {
    System.out.println("Date Comparison");
    return other.day == day && other.month == month && other.year == year;
}
```



Allowing European Dates

putting the day before the month

- Put a lot of effort into the GTDate class
 - so I want to be able to use / reuse it
 - without just copying my code to a new file
- `public class EuropeanDate extends GTDate`
- What needs to change -- be added
 - Constructors are not inherited!
 - `super`
 - `toString?`, `equals?`
 - changes to `GTDate?`