Some notes on inheritance, review

Darrell Bethea
June 9, 2011
Announcements

- Program 4 due tomorrow

- Final exam review tomorrow
  - Bring questions

- Final exam
  - Comprehensive
  - Monday, 6/13, 8–11 AM
  - SN014
Questions?
Final exam

- Final exam will cover everything we have covered
  - Lectures
  - Readings from textbook
  - Labs
  - Programs
  - In-class exercises/worksheets
  - Midterm
Tomorrow’s Final Exam Review

- No formal review
- Look over previous assignments, lectures, midterm, etc.
- Come prepared with questions
Today in COMP 110

- Revisiting the equals method
- Array and inheritance review
We can test whether an object is of a certain class type:

```java
if (obj instanceof Student) {
    System.out.println("obj is an instance of the class Student");
}
```

**Syntax:**

`object instanceof Class_Name`

**Use this operator in the equals method**
The equals method

- For our Student class

```java
public boolean equals(Object obj)
{
    if ((obj != null) && (obj instanceof Student))
    {
        Student otherStudent = (Student) obj;
        return (this.id == otherStudent.id);
    }
    return false;
}
```

- Reminder: `null` is a special constant that can be assigned to a variable of a class type – means that the variable does not refer to anything right now
The `equals` method

- Implements an equivalence relation:
  - It is **reflexive**
    - For any non-null reference value `x`, `x.equals(x)` should return true
The equals method

- It is **symmetric**
  - For any non-null reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true
The `equals` method

- **It is transitive**
  - For any non-null reference values `x`, `y`, and `z`, if `x.equals(y)` returns true and `y.equals(z)` returns true, then `x.equals(z)` should return true.
The equals method

- It is **consistent**
  - For any non-null reference values `x` and `y`, multiple invocations of `x.equals(y)` consistently return true or consistently return false, provided no information used in equals comparisons on the objects is modified.
For any non-null reference value x, x.equals(null) should return false.
The equals method

- This implementation is not symmetric

```java
public boolean equals(Object obj) {
    if ((obj != null) && (obj instanceof Student)) {
        Student otherStudent = (Student) obj;
        return (this.id == otherStudent.id);
    }
    return false;
}
```
The equals method

- Why?
  - The `instanceof` operator will return `true` if `obj` is a subclass of `Student`

```java
public boolean equals(Object obj) {
    if ((obj != null) && (obj instanceof Student)) {
        Student otherStudent = (Student) obj;
        return (this.id == otherStudent.id);
    }
    return false;
}
```
The equals method

- Fourth try

```java
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if ((obj == null) || (obj.getClass() != this.getClass()))
        return false;

    Student otherStudent = (Student) obj;
    return (this.id == otherStudent.id);
}
```

- The `getClass` method will return the runtime class of an object, so if `obj` is a subclass of `Student` (in this case), this method will return `false`
Array and Inheritance review worksheet

- Answers posted after class
Tomorrow

- Final exam review
- Come prepared with questions!