CS 211: Inheritance

Chris Kauffman

Week 5-1
Logistics

Goals Today
- Basics of inheritance
- Overriding methods

Reminder: Career Fair
- 11:00 a.m.- 4:00 p.m.
- Dewberry Hall
- Wed 2/18: Science/Tech
- Thu 2/19: Business/Non-tech

Reading: Inheritance
- Building Java Programs Ch 9
- Lab Manual Ch 7

Lab 05: Exercise
- Inheriting from PrintWriter
Warmup: String Concatenation

Diagram 1
What does

String a = "hello";
String b = " world";
String c = a+b;

actually do in memory?

Diagram 2
How about

String s = "";
for(int i=0; i<10; i++){
    s = s + i;
}

Inheritance

Warning: Inheritance is a tricky subject because...

- It’s not too bad to understand basic mechanics
- Creates behavior only observable at runtime
- **Spreads out code** to do one task into multiple places
- Advantages are not apparent until you have a large system
- Teaching examples do not reflect what inheritance is good for

Our Approach

- Spend today and part of Thursday on mechanics of inheritance
- These will involve little examples with mostly bad practice associated with it
- Then discuss good/bad applications of inheritance and why `extends` may in fact be evil
Basic Inheritance Mechanics: Animals.java

Primary reason for an inheritance hierarchy is to create a container for several kinds of things that can behave differently.

class Animal { }
class Human extends Animal { } 
class Mouse extends Animal { }

main(){
    Animal animals[] = new Animal[] {
        new Animal(),
        new Human(),
        new Mouse()
    };

    ...
}

▶ Each animal implements its own proclaim() method
▶ Each behaves differently on
public class Coord {
    public final int row;
    public final int col;
    public Coord(int ir, int ic) {
        this.row = ir;
        this.col = ic;
    }
    public String toString() {
        return String.format("(%d,%d)", row, col);
    }
    public boolean equals(Coord c) {
        return this.row == c.row &&
                this.col == c.col;
    }
}

public class Coord3D extends Coord {
    // Fields row and col are inherited
    public final int height;
    public Coord3D(int ir, int ic, int h) {
        super(ir, ic); // Required
        this.height = h;
    }
    public String toString() {
        return String.format("(%d,%d,%d)", row, col, height);
    }
    public boolean equals(Coord3D other) {
        return this.row == other.row &&
                this.col == other.col &&
                this.height == other.height;
    }
}
Note on Constructors

- Animal did not specify a constructor
- Java always provides a default 0-argument constructor if no constructors are specified
  ```java
  Animal a = new Animal();
  ```
- The constructor for Human initializes its parent class automagically as follows
  ```java
  public Human(){ // Created automatically
    super(); // Call done automatically
  }
  ```
- Coord has a two-argument constructor
  ```java
  Coord c = new Coord(1,2);
  ```
- That means it's now illegal to say
  ```java
  Coord c = new Coord();
  ```
  unless a zero-arg constructor is explicitly defined
- Coord3D must call a valid parent constructor
- Coord3D must therefore call constructor
  ```java
  super(ir,ic);
  ```
That’s super!

Keyword this gives access to present class’s fields and methods

```java
this(arg1, arg2, arg3);  // call another constructor
this.someField = stuff;  // access a field
this.doSomething(x, y);  // call a method
```

Keyword super gives access to parent class’s fields and methods

```java
super(arg1, arg2, arg3);  // call parent constructor
super.someField = stuff;  // access parent field
super.doSomething(x, y);  // call parent method
```
Equality Gets Trickier

What is printed on the right based on equals() definitions?

```java
public class Coord {
  public boolean equals(Coord c) {
    return
      this.row == c.row &&
      this.col == c.col;
  }
}

public class Coord3D extends Coord {
  public boolean equals(Coord3D other) {
    return
      this.row == other.row &&
      this.col == other.col &&
      this.height == other.height;
  }
}

Coord a = new Coord(1,2);
Coord b = new Coord(10,12);
Coord3D c = new Coord3D(1,2,3);
Coord3D d = new Coord3D(10,12,14);

System.out.println( a.equals(b) );
System.out.println( a.equals(c) );
System.out.println( a.equals(d) );
System.out.println();
System.out.println( b.equals(c) );
System.out.println( b.equals(d) );
System.out.println();
System.out.println( c.equals(d) );
System.out.println();
String s = "(1,2)";
System.out.println( s.equals(a) );
System.out.println( a.equals(s) );
```