Announcements

- P6 due Dec 1 (last Thursday of classes!)
- Final exam conflict: You must inform us by 3pm on Friday, 11/17. You need to provide your entire exam schedule during the exam period.
- CIT survey on classroom technologies available online (see course website). Please help us evaluate, and improve, the use of classroom technologies! Thanks in advance!

Previous Lecture:
- Inheritance—extending a class
- Constructor in the subclass

Today’s Lecture:
- Overriding methods
  - Using super to access members from the superclass
  - What is “polymorphism”?
- Reading:
  - Sec 7.2. Optional: Sec 7.3

Calling one constructor from another

- In a subclass’ constructor, call the superclass’ constructor with the keyword super instead of the superclass’ (constructor’s) name
- To call another constructor from a constructor in the same class, use the keyword this
- Always make a call to a constructor (super or this) as the 1st statement in a constructor in a subclass!

Overriding methods

- Subclass can override definition of inherited method
- New method in subclass must have same signature as superclass (but has different method body)
- Which method gets used??
  - The object that is used to invoke a method determines which version is used
- Method declared to be final cannot be overridden
- Do not confuse overriding with overloading!

Overridden methods: which version gets invoked?
To create TrickDice: call the TrickDice constructor, which calls the Dice constructor, which calls the roll method. Which roll method gets invoked?

```java
class Dice {
    public Dice(…) {
        roll();
    }
    public void roll() {…
        //…other methods, fields
    }
}
class TrickDice extends Dice {
    public TrickDice(…) {
        super(…);
    }
    public void roll() {…
        //…other methods, fields
    }
}
```

Accessing members in superclass super

- From constructor in subclass, call superclass’ constructor
- Access superclass’ version of a overridden method. E.g.:

```java
super.toString()
```
static methods & variables
- Do not re-declare static components!
- Same rules for inheritance (accessibility) with respect to visibility modifiers
- Static method: implicitly final
- Static variable: same memory space as superclass

Important ideas in inheritance
- Single inheritance
- Keep common features as high in the hierarchy as reasonably possible
- Use the superclass’ features as much as possible
- “Inherited” ⇒ “can be accessed as though declared locally” (private variables in superclass exist in subclasses; they just cannot be accessed directly)
- Inherited features are continually passed down the line
- Use different hierarchies for different problems

Polymorphism
- “Have many forms”
- A polymorphic reference refers to different objects (related through inheritance) at different times

Suppose class Plane extends Vehicle
Vehicle mover; //a Vehicle reference
Plane flyer; //a Plane reference
mover = new Vehicle(...);
flyer = new Plane(...);
// A plane is a vehicle
mover = new Plane(...);
mover = flyer;
// A vehicle is not a plane
flyer = new Vehicle(...); //invalid

Another polymorphic example
Vehicle[] mover = new Vehicle[5];
mover[0] = new Vehicle(...);
mover[1] = new Plane(...);
mover[2] = new Plane(...);
mover[3] = mover[1];

The reference type may not be the same as the object type!

Accessing methods/variables through a polymorphic reference
Dice d = new TrickDice(...);

Consider the reference type and object type:
1. Which type determines whether a method/variable can be accessed?
2. For an overridden method, which type determines which version gets invoked?