Readings
This Week: Ch 9.4-9.5 and into Ch 10.1-10.8
(Ch 11.4-11.5 and into Ch 13 in old 2nd ed)

(Reminder: Readings are absolutely vital for
learning this stuff!)

Labs and Tutorials
This Week: Lab #9

Survey #2
- Dr. Ben Yu’s second survey for you all is up
  on WebCT now.
- Completely optional…
- But, there’s a bit of extra credit for doing all
  three surveys by their respective deadlines!
- The deadline for this one is March 29.
- Because of the server crash, if you did
  this survey before the crash, you’ll have
to do it again. ☹

Final Exam
- Wednesday, April 15, 7pm in SRC A
  - This wasn’t a good room last year, but we’re
    stuck with the date, time, and room UBC
    assigns. ☺

Programming Assignment 3
- Assignment 3 is up on WebCT!
  - Click on the “Assignments” icon.
  - Assigned Sunday evening – sorry for delay!
- Due at NOON, April 6 (Monday), via
  electronic hand in.
- Tips:
  - There is some Eclipse setup. Set-up ASAP!
  - Work in pairs. Some conceptual stuff.
  - Think carefully before coding. If concepts
    right, the coding is much much easier.
Programming Assignment 3

- I found out after class on Monday that some versions of Vista and Java don’t quite get along for some graphics programs.
- I’ve spent the last couple days pinning this down. My initial guesses weren’t correct.
- I have a fix now that works for several people. See the assignment webpage or WebCT for details.

Last Time: Inheritance

- Inheritance in real life: stuff you get from your parents, e.g.:
  - Harry Potter inherited his father’s hair and his mother’s eyes.
  - I inherited my nearsightedness from both of my parents.
  - He inherited his mother’s sense of humor.
- (Think of inheriting characteristics and behaviors, not money and real estate.)

Inheritance in Java

- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.
  ```java
  public class ChildClass extends ParentClass
  {
    ...
    put any additional fields and methods here
    ...
  }
  ```

Silly Inheritance Example

- Let’s make UBCStudent extend TalkingDoll…
Inheritance Terminology

- The child class **extends** the parent class.
- The parent class is called the superclass.
- The child class is called the subclass.
- The child class (subclass) inherits stuff from its parent (superclass), so it has more stuff (fields and methods).
- Mnemonic: your parents are your **superiors**.

In our silly example, which class was the parent? Which was the child?

Inheritance Terminology

- The child class **extends** the parent class.
- The parent class is called the superclass.
- The child class is called the subclass.
- The child class (subclass) inherits stuff from its parent (superclass), so it has more stuff (fields and methods).
- Mnemonic: your parents are your **superiors**.

What was a method that the subclass inherited from the superclass?

Inheritance Terminology

- The child class **extends** the parent class.
- The parent class is called the superclass.
- The child class is called the subclass.
- The child class (subclass) inherits stuff from its parent (superclass), so it has more stuff (fields and methods).
- Mnemonic: your parents are your **superiors**.

The subclass has more stuff than the superclass. Can you give an example?
Silly Inheritance Example

- Let's make UBCStudent extend TalkingDoll…

- Hey, UBCStudent and TalkingDoll both declare a private instance field called name. Let's eliminate the duplicate code…

Accessing the Superclass

- Even though the subclass has all the instance fields and methods of the superclass, Java still thinks of it as a separate class. So, the subclass can't directly access anything private!

Accessing the Superclass

- Even though the subclass has all the instance fields and methods of the superclass, Java still thinks of it as a separate class. So, the subclass can't directly access anything private!

- What do you do?
  - Use the public accessor/mutator methods.
  - Have the superclass make things public.
  - (Have the superclass make things protected.)

Overriding

- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.

```java
public class ChildClass extends ParentClass {
    anything new here is added to the ChildClass
    anything with same signature as in ParentClass overrides the ParentClass
}
```

Recap: Variable Scope and Shadowing

- Scope tells you which declarations you can see from which points in the program.
  - The scope of a variable is the places in the program where the variable can be accessed.
  - Or you can think of scope as the places where you can see some variable.

Shadowing and Overriding

- These are basically the same concept:
  - If you have two declarations of the same signature (name and parameters), the closer declaration wins.
Shadowing and Overriding

These are basically the same concept:
- If you have two declarations of the same signature (name and parameters), the closer declaration wins.

Shadowing

```java
public class foo {
    String name;
    public void setName(String name) {
        // "name" is shadowed!
        this.name = name;
    }
}
```

Overriding

```java
public class bar {
    public void a(double b) {...}
}
public class foo extends bar {
    public void a(double b) {...} // override!
}
```

Shadowing and Overriding

These are basically the same concept:
- If you have two declarations of the same signature (name and parameters), the closer declaration wins.

Overriding

```java
public class bar {
    public void a(double b) {...}
}
public class foo extends bar {
    public void a(int b) {...} // no override
}
```

**super and this**

- The keyword `super` works just like the keyword `this` to get around the shadowing.
- `this` means the current object (implicit parameter)
- `super` means the superclass object.

Using `super` to Avoid Overriding

These are basically the same concept:
- If you have two declarations of the same signature (name and parameters), the closer declaration wins.

Overriding

```java
public class bar {
    public void a(double b) {...}
}
public class foo extends bar {
    public void a(double b) {...} // override!
        super.a(3.14); // bypasses override
    }
```
Special Case: Superclass Constructor
- Recall the special use of this in constructors?
  - this as first line of constructor calls a different constructor for the same object
  - E.g., UBCStudent class

Similar trick to call a superclass constructor using the super keyword:
- super call must be first line of constructor
  - why?

How Child Objects Are Constructed
- The child object contains everything the superclass object does, so the superclass constructor needs to get called.
- Java automatically calls the superclass’s constructor, before the child’s constructor starts working.
  - The parameters it passes to the superclass constructor are the same as were passed to the subclass constructor.
- The special case of calling super on the first line tells Java to call the superclass constructor in the way you specify, instead of automatically with the same arguments.