**Please read section 10.3 in your textbook and answer the following questions.**

**1. What are THREE ways a subclass can define methods of a superclass?**

You can OVERRIDE methods from the superclass by specifying a method with the SAME SIGNATURE line as the same method in the superclass.

You can INHERIT methods from the superclass, which happens automatically if you do not override it.

You can define NEW methods that do not exist in the superclass. These new methods can only be applied to subclass objects and any other classes below it – NOT its superclass.

**2. Explain how instance variables work in a subclass.  In particular, can they override instance variables in the superclass?  Can they have the same name as in the superclass?**

Unlike subclass methods, subclass instance variables DO NOT override superclass instance variables. The subclass may inherit the superclass instance variables, or new instance variables may be created, but may only be present in subclass objects. While a subclass instance variable IS ALLOWED to have the same name as a superclass instance variable, it is not a good idea, as two variables with the same name that can hold different values will likely lead to confusion.

**3. Instance variables in the superclass are private, so they cannot be directly accessed.  How does the subclass gain access to these instance variables?**

Subclasses CANNOT access superclass private instance variables. The only way a subclass can modify a private superclass instance variable is if it is put through a PUBLIC method of the superclass, which the subclass CAN access.

**4. On page 447/448, the goal is to implement the deposit method for the CheckingAccount class.  Describe how this method had to be written to work properly.**

It is not possible to just add **amount** and **balance** because balance is a private instance variable of the superclass, and the subclass cannot access that. We need to call the superclass method of **deposit**, rather than the subclass **deposit**, which would send us into an infinite loop, and pass in **amount** as a parameter. If we write **super.deposit(amount);** then the superclass **deposit** method will be called, allowing us to correctly complete the deposit method.

**5. Explain what the "Common Error 10.3" box on page 450 is trying to describe.  How would you fix this error?**

Same as above, calling simply **withdraw(amount)**  with no qualifier would call the same class that it is currently in over and over again, sending the program into an infinite loop. The super. qualifier must be added to the beginning of the line so that it calls the superclass method of **withdraw**, rather than the subclass method it is currently in. it is very important to precisely identify which withdraw method the subclass method should call, rather than leaving it blank.