**Chapter 11**

Ex. 11.1 – Remove display method from the post class and add it to photo post and message post. What do you observe?

The project no longer compile. Because the display method can’t access the private fields of Post from within in the sub class.

Ex 11.2 - Implement the code form the book, Which display methods execute?

In both cases the subclass’s display method was called and Post’s was never called.

Ex 11.3 – Modify your latest version of the network project to include the super call in the display method. How does it work? What issues might this solution have?

Done and saved. When the method is used it first calls and executes the code from the superclass’s display method executes the lines defined in the sub class. The biggest issue I see with it is it kind of defeats the purpose of using inheritance because code must be added to all sub-classes and one line duplicated every time.

Ex. 11.4 – Look up the toString in the library documentation. What are its parameters and return type.

The toString method takes no parameters and returns a String.

Ex. 11.5 – Create an object of the class PhotoPost and invoke the toString method from the object submenu in the objects pop up menu

Done string returned was “PhotoPost@11ee322”.

Ex. 11.6 – Re-order the statements in the method of your network project tp get details printed the same as they are in figure 11.10.

Done and saved.

Ex. 11.7 – Make the nessacary changes to get the output generated in figure 11.11

Done and saved

Ex. 11.8 – Implement a transporter room with inhereitance in your version of the zuul project.

Done and saved.

This was way harder than it sounds. Rooms had to be put into an arrayList so that I could get a “randomly” selected room to be set as the current room.

Ex. 11.9 – Discuss how inheritance could be used in the zuul project to implement a player and a monster class.

Inheritance could be used to make player and monster classes a sub class to on called character or being, something along those lines. Common things like name such as name and race would be managed by the super class and the subclasses can manage the differences like hp, an location history.

Ex. 11.10 – Could inheritance be used to create and inheritance relationship between a character in the game and an item?

You could but it may not be wise. If there was ever a thought to expand the game past our very basic one, it would be better to separate the two. But if one wanted you could make character/player a subclass of item class. Name, description, etc. would be kept in the item class. Character would still hold inventory and other things that a normal item doesn’t have.

Ex. 11.11 – Printer is a sub-class of device. Which of these class must have a definition of the method get name?

Device dev = new Printer();

Dev.getName();

With how the code is written the getName method is defined in the device class. We can tell because when the object printer is created it is an instance of device as well as an instance of printer.

Ex. 11.12 – In the same situation as above, if both classes have an implementation of getName will be executed?

The getName method of printer will be executed. Without “super.” Before the method, it should call the subclasses method.

Ex. 11.13– Assume that you write a class Student that does not have a declared superclass. You do not write a toString method. Consider the following lines of code:

Student st = new Strudent();

String s = st.toString();

Will these lines compile?

Yes, the following should compile because in Java all classes are a subclass to Object (where the toString(); method is defined) even without explicit declaration. So the code is run it will call the method in the superclass.

Ex. 11.14– In the same situation as before (class Student, no toString method), will the following compile?

Student st = new Strudent();

System.out.println(st);

This will compile because the line “System.ou.print() overrides the toString method. The rest of the reasoning the same as above, since all classes have the superclass object the expression will not only compile but should run.

Ex. 11.15– Assume that your class Student overrides toString so that it returns the student’s name. You now have a list of students. Will the following code compile?

For(Object st : myList){

System.out.println(st);

}

Yes, it would compile and it should print the name of every student on the list. It will display their names because the toString method has been overridden in the student class.

Ex. 11.16 – write a few lines of code that result in a situation where a variable x has a static type T and the dynamic type D.

After looking at the first line of code of exercise 11.11 (“Device dev = new Printer();”) for structure the line for this exercise should be

T x = newD();