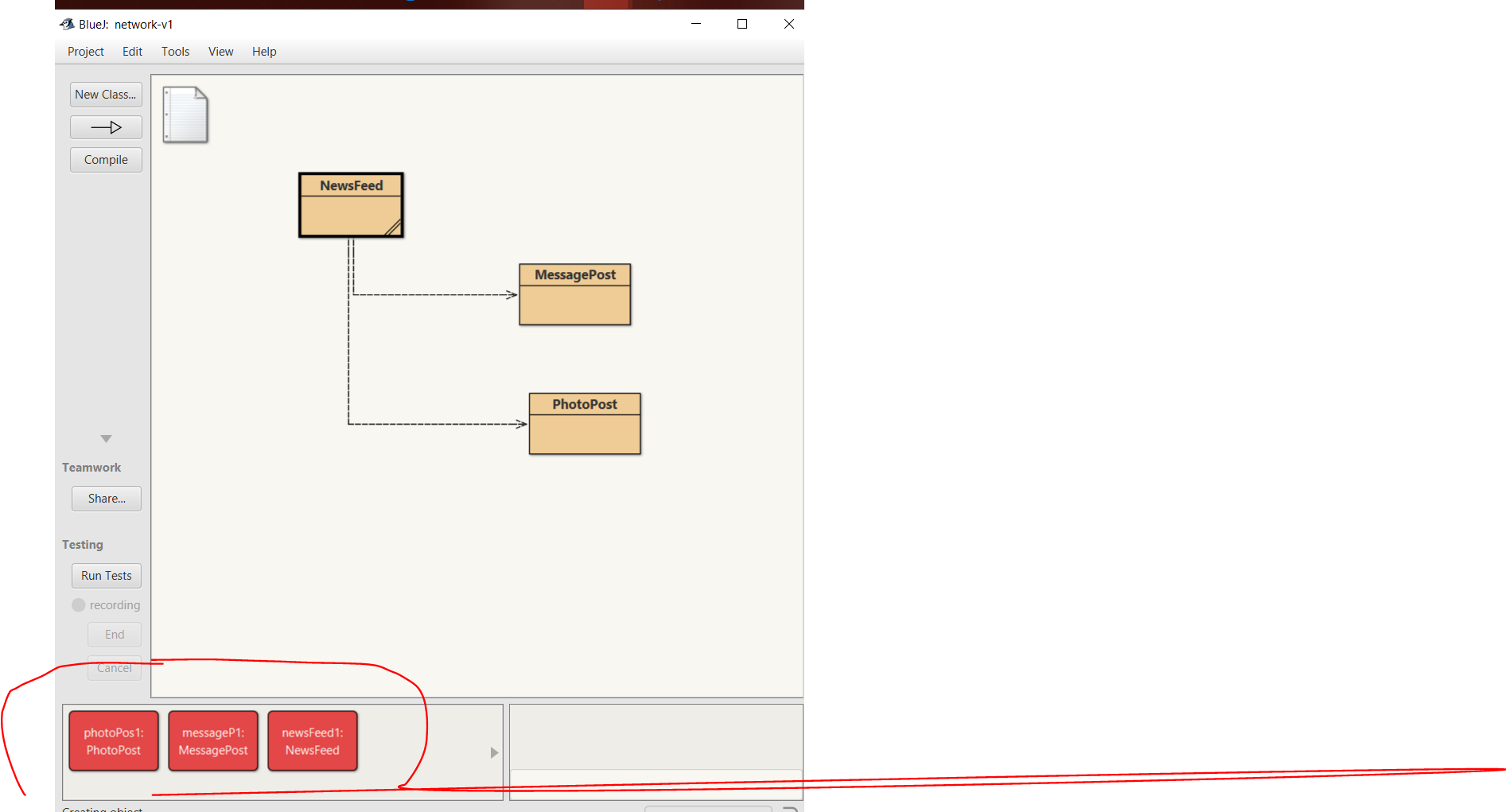
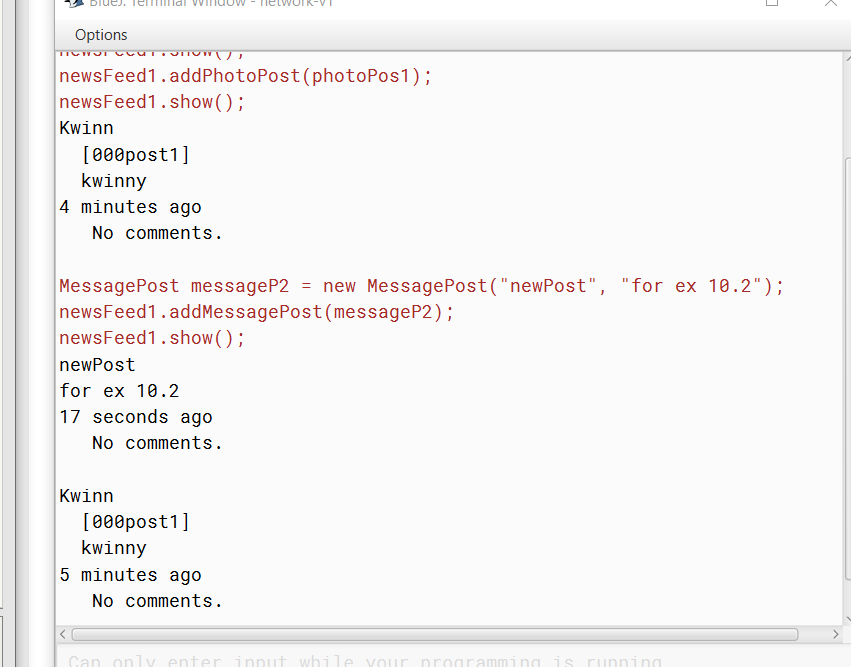
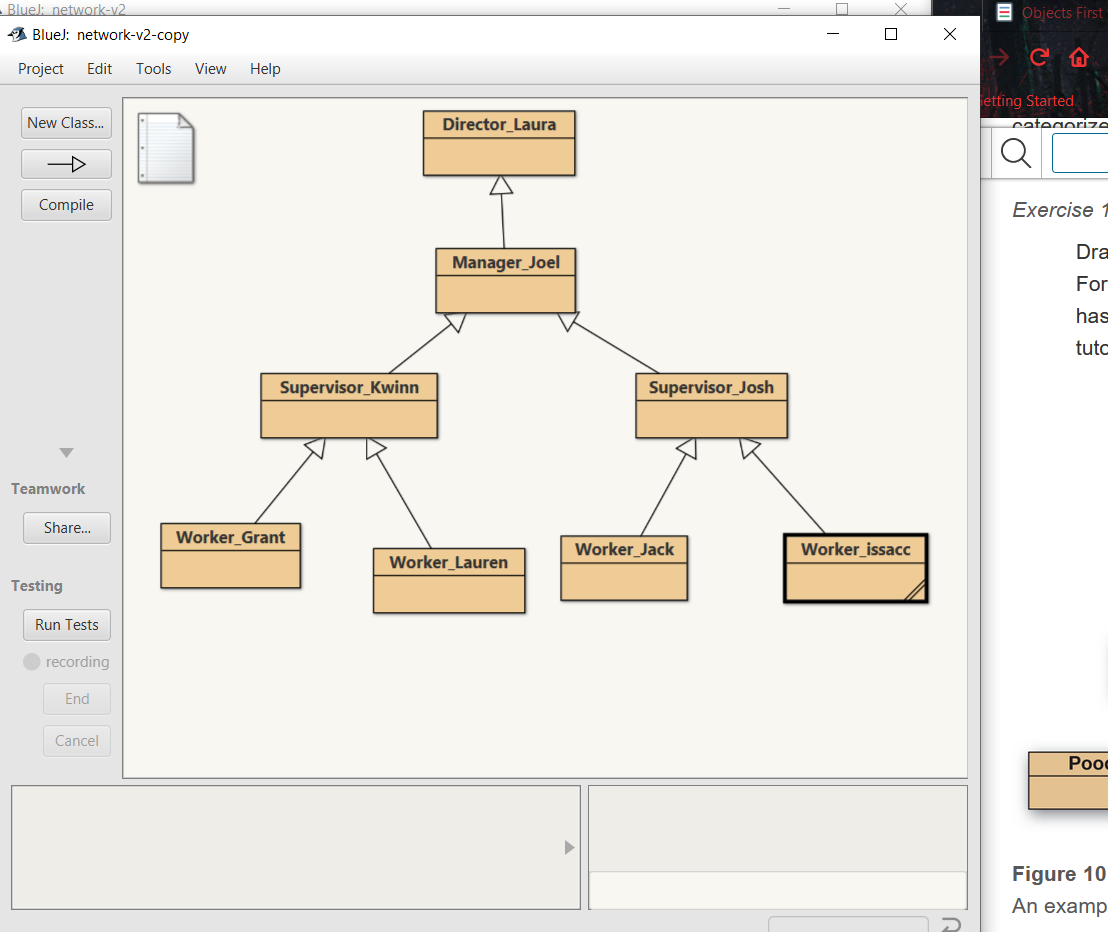
10.1



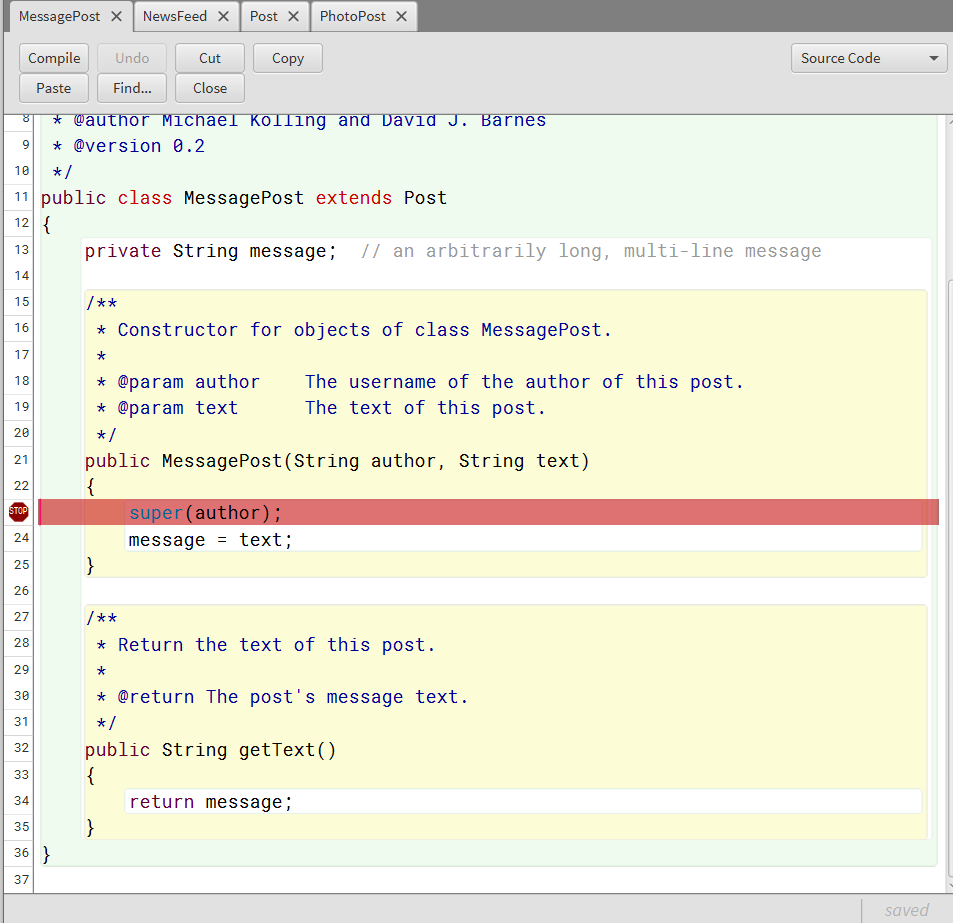
10.2

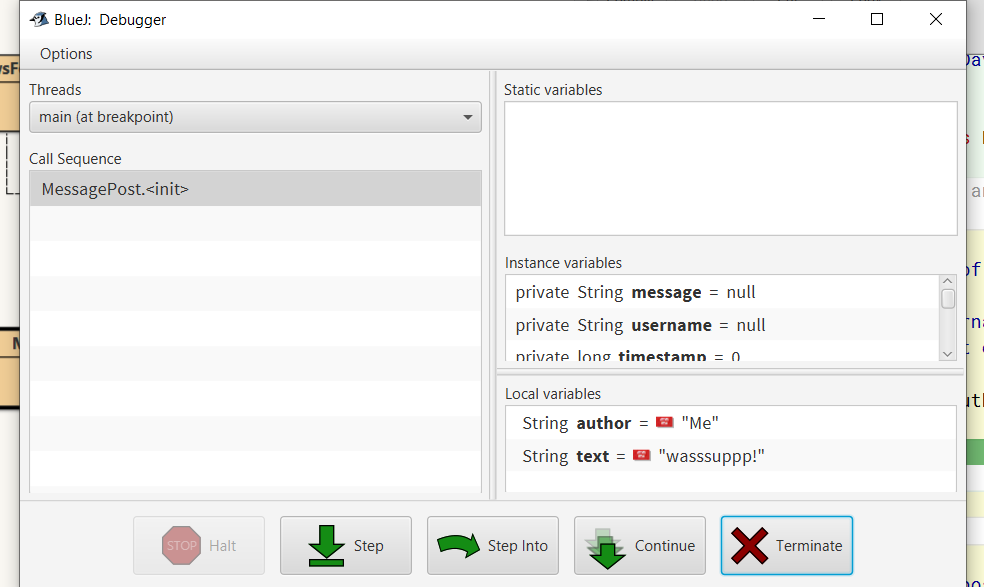
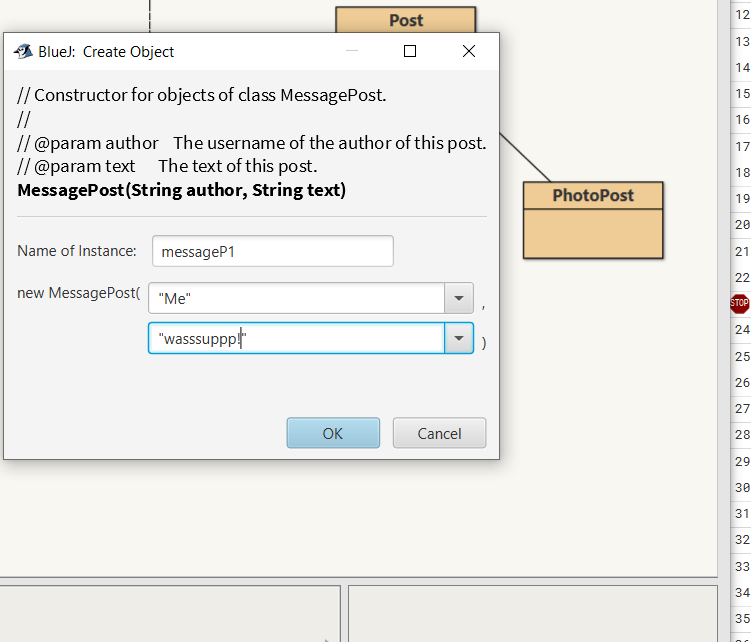


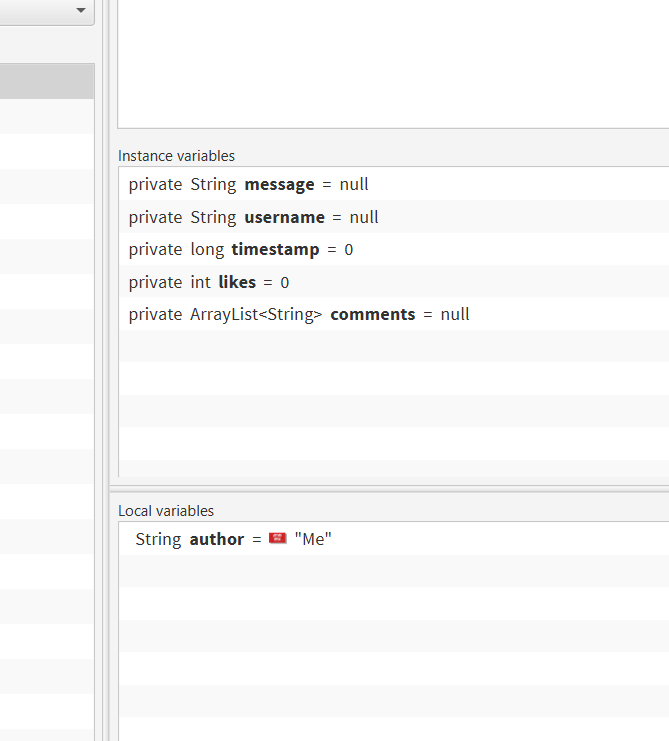
10.3



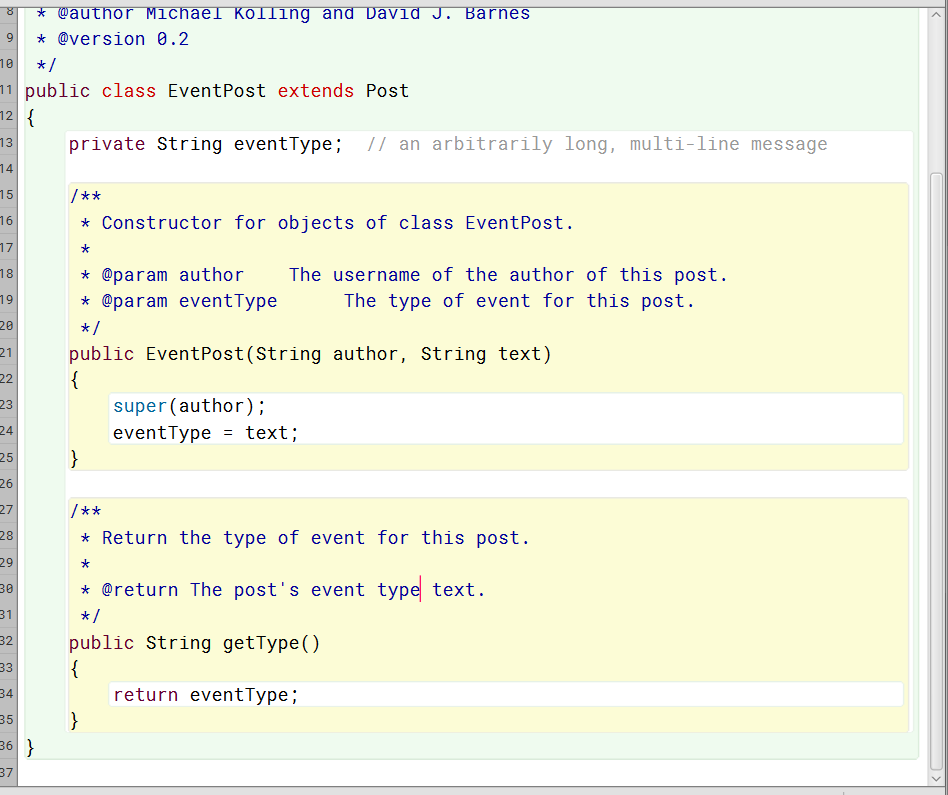
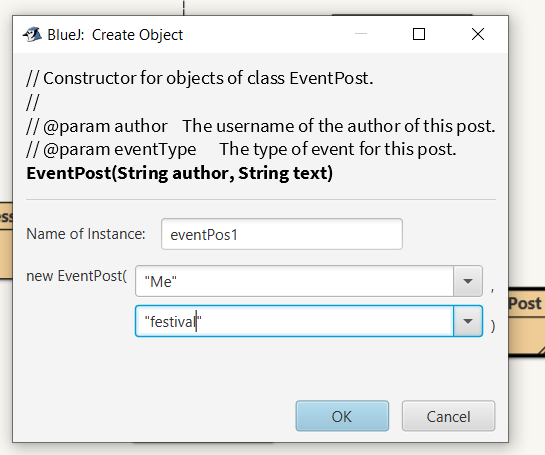
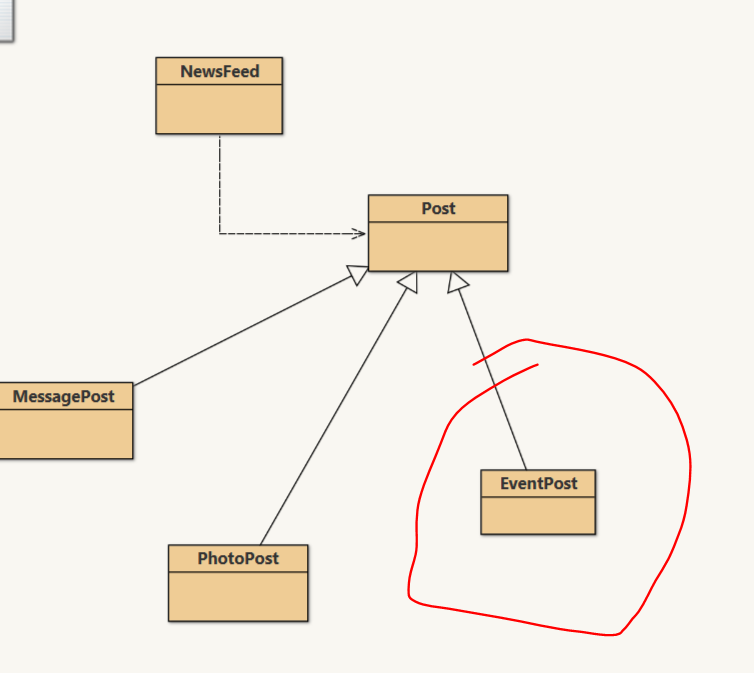
10.7



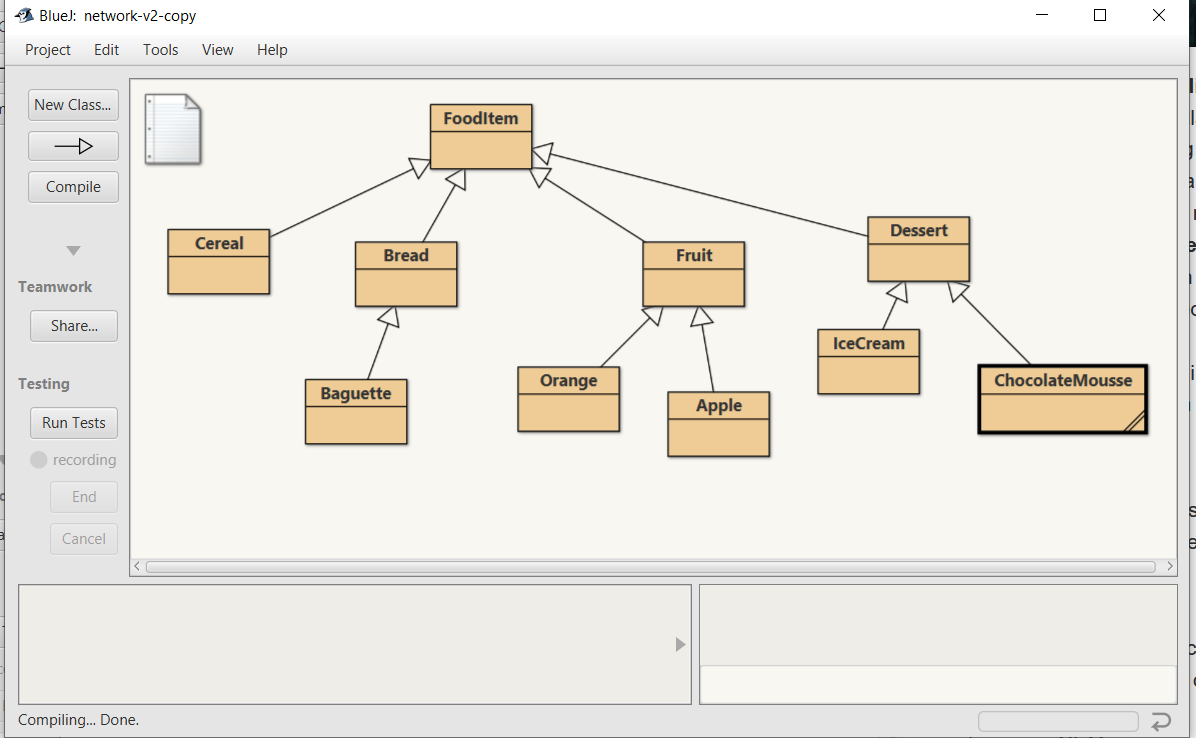


As we can see here in the constructor of the subclass MessagePost, on line 23 where I put the breakpoint at for this exercise, we make an initial call to the constructor of its parent, superclass Post. This constructor is run first and initializes much of the fields of the MessagePost object and then the rest of the MessagePost constructor is run, In this case message = text;.

10.8



10.9



10.10

I feel like a touch pad and a mouse would both be members of the same superclass for sure. Maybe an input device superclass.

10.11

The inheritance relationship between a rectangle and a square would be kind of the same as with the mouse and the touchpad. Both would be under the same superclass of polygons. Would this then make them siblings really.

10.12

A.)

Person p1 = new Student(); YES

Person p2 = new PhDStudent(); YES

PhDStudent phd1 = new Student(); NO

Teacher t1 = new Person(); NO

Student s1 = new PhDStudent(); YES

B.)

s1 = p1; Legal

s1 =p2; Legal

p1 = s1; Illegal

t1 = s1; Illegal

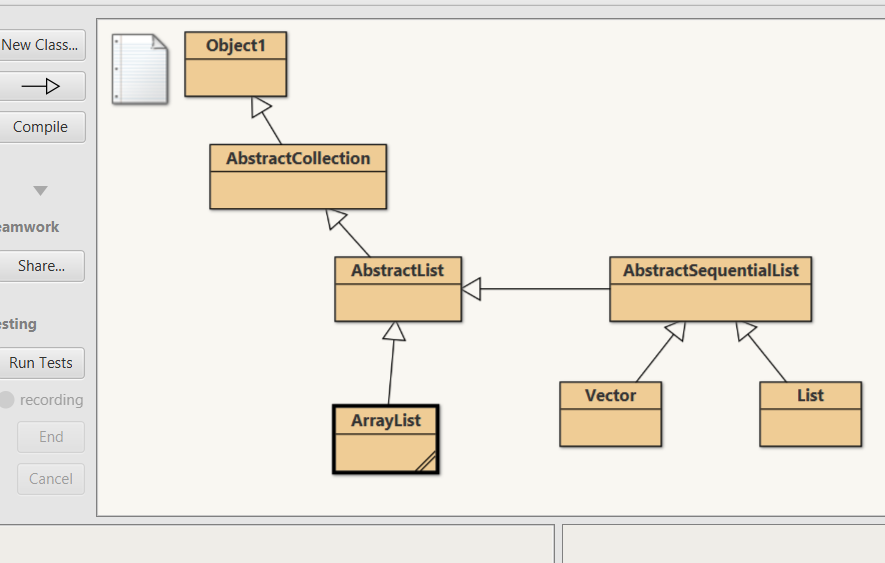
s1 = phd1; Legal

phd1 = s1; Illegal

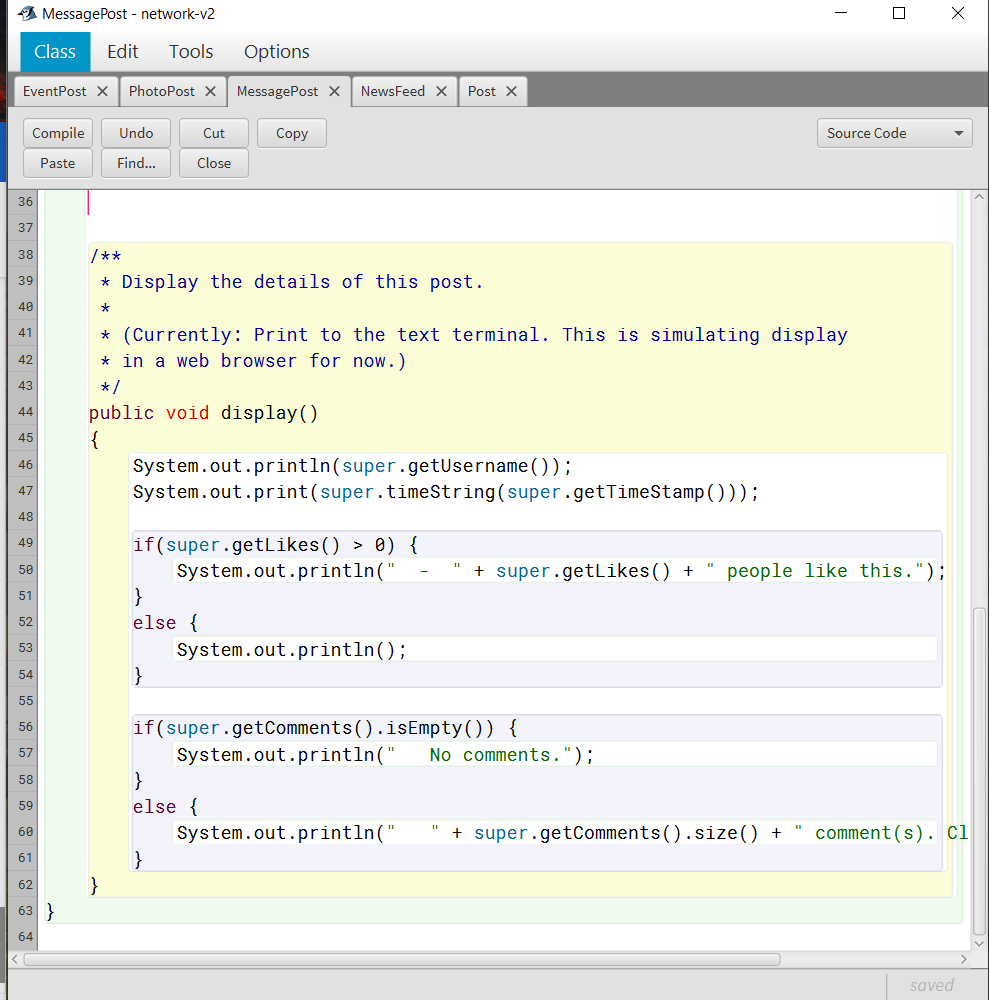
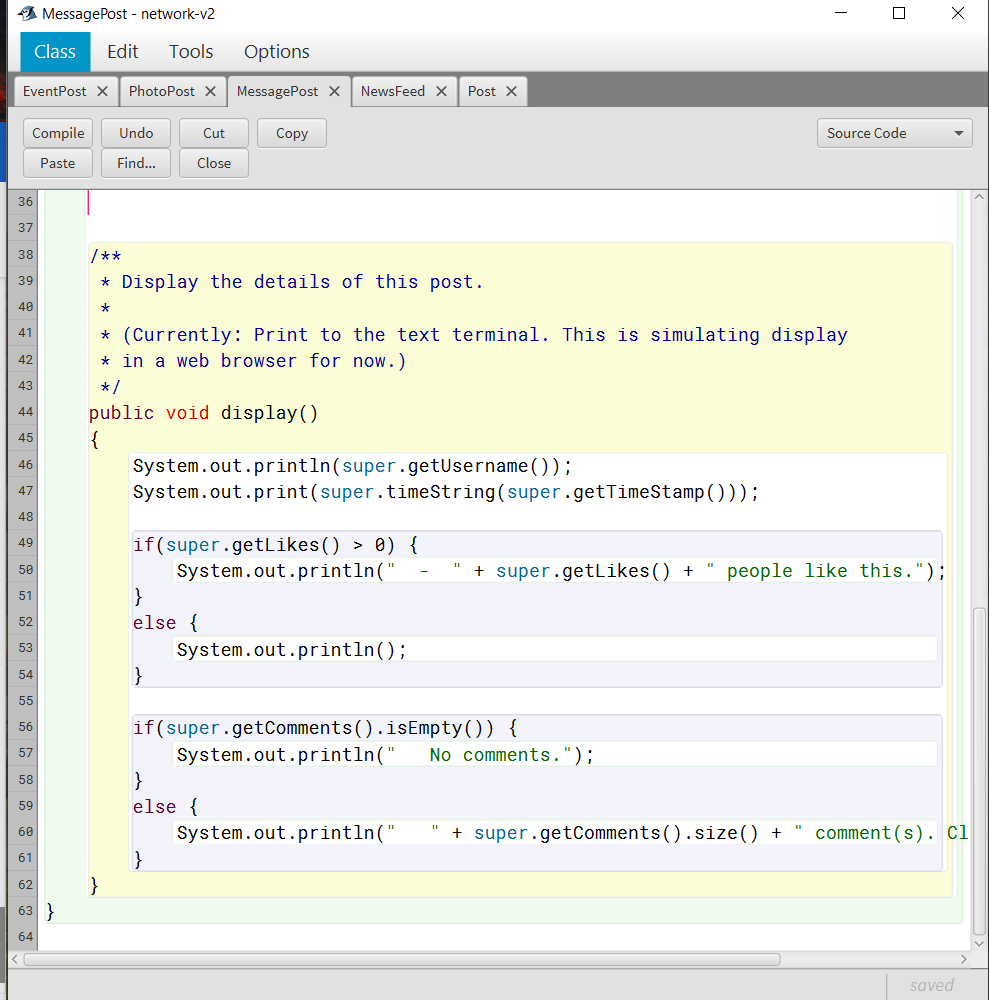
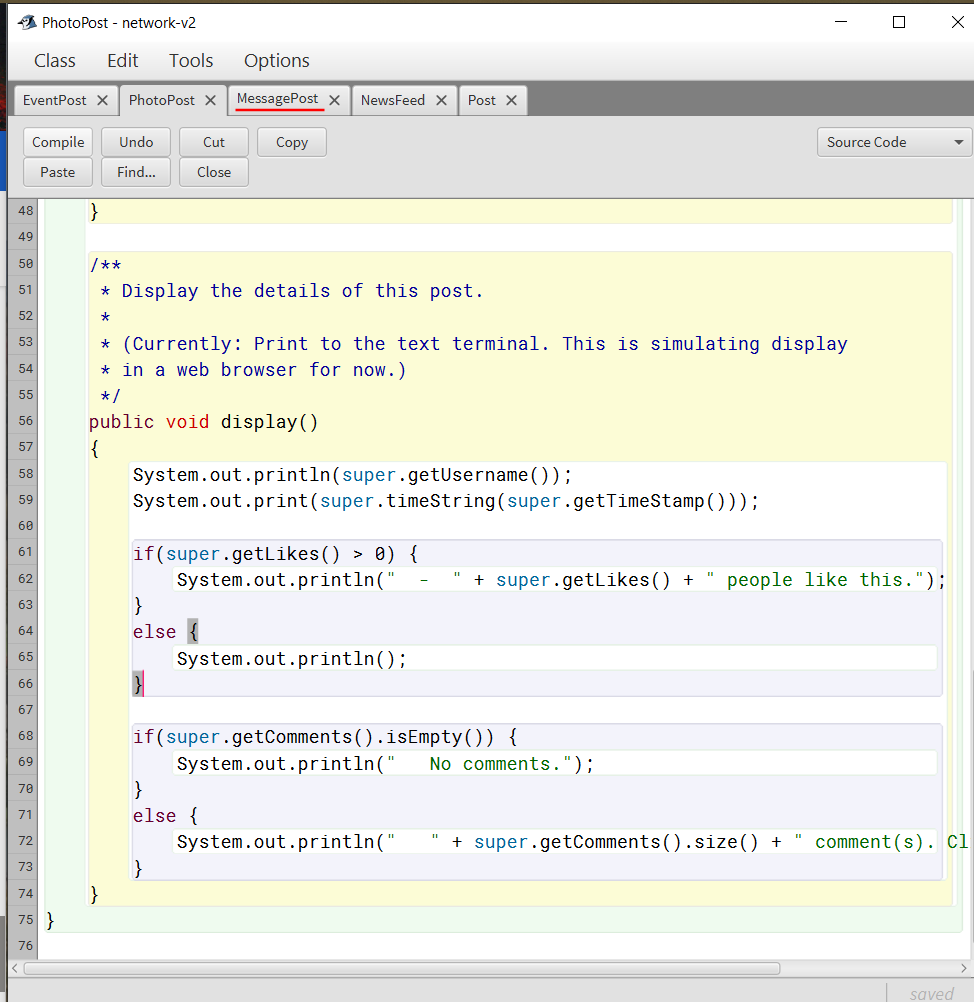
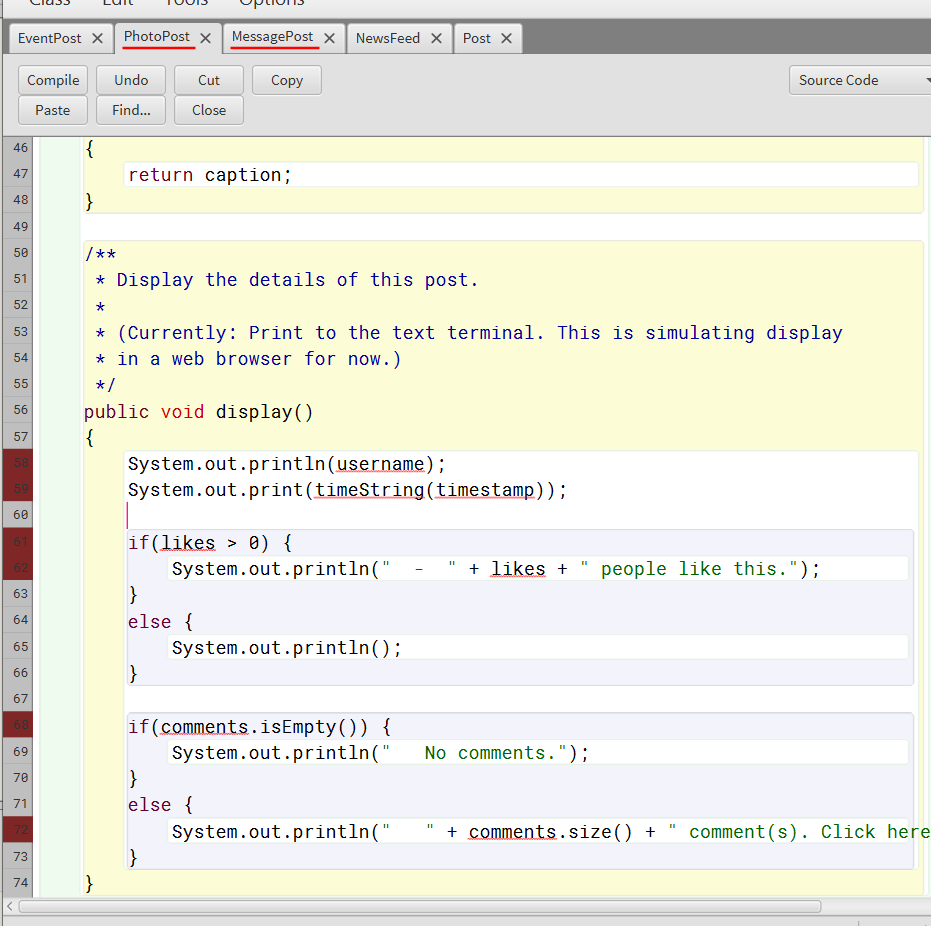
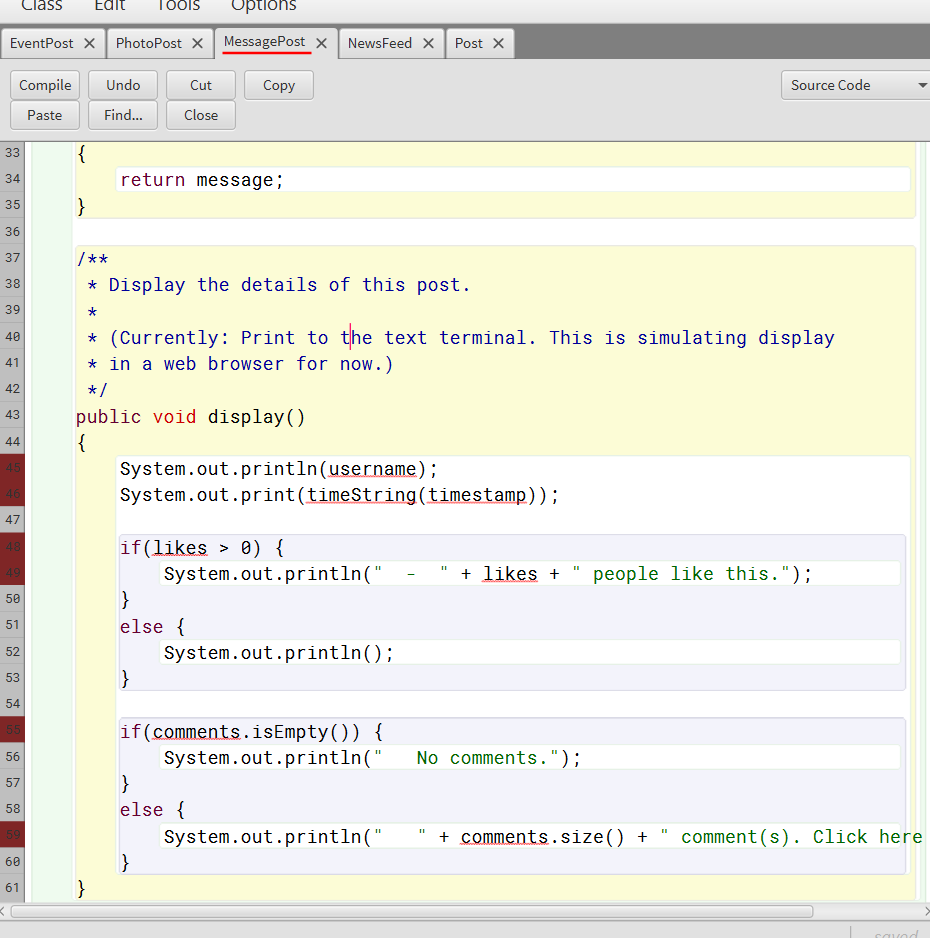
10.14

Really nothing has to change in the NewsFeed class when another Post subclass is added. This is the beauty of using sub classes in a situation like this.

10.19

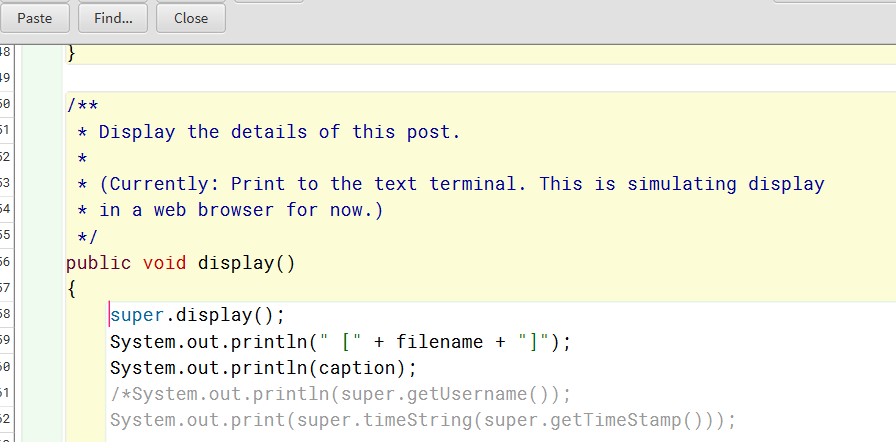
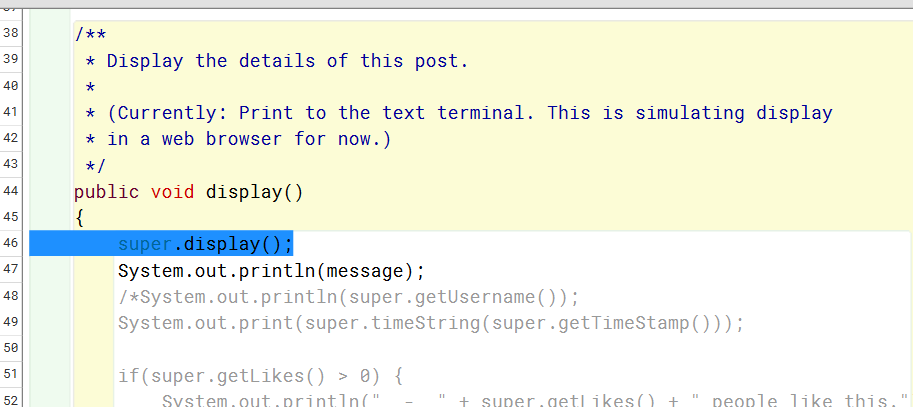


11.1



After working with the display methods and getting them to work in each of the different classes for the types of posts. I found this last picture to show the problem... We will have to find another way to call the display method in this way from the news feed.

11.3

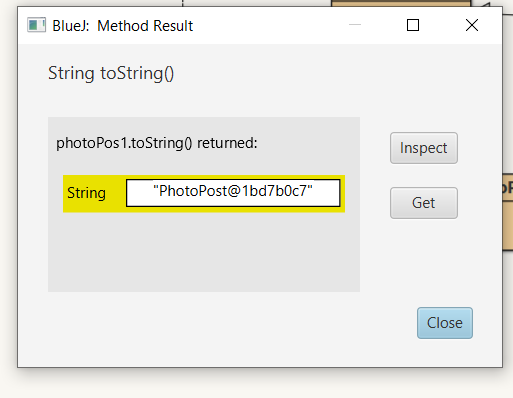


Using the super key word here did seem to do the job.

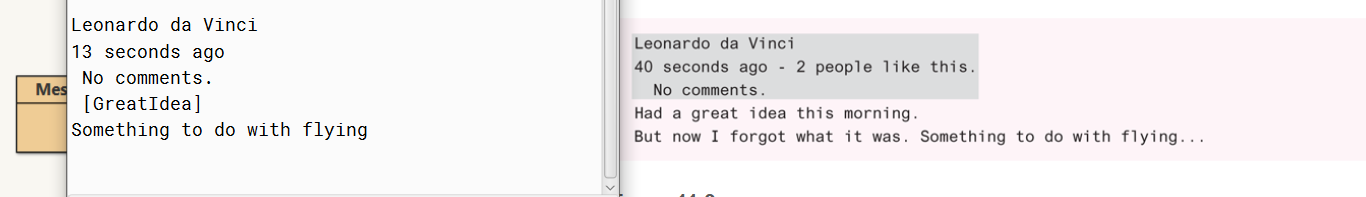
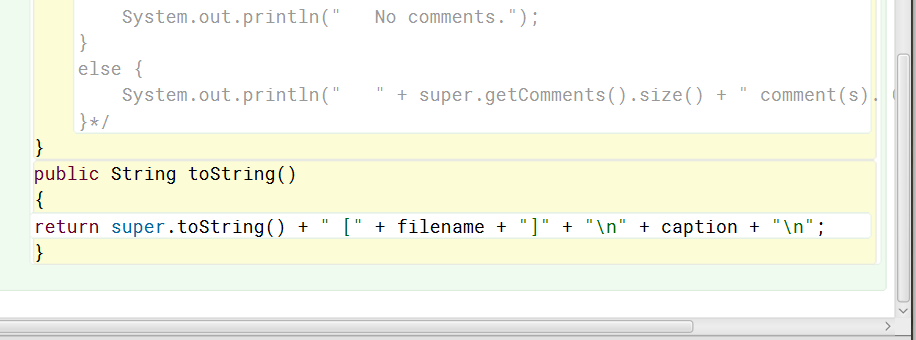
11.4

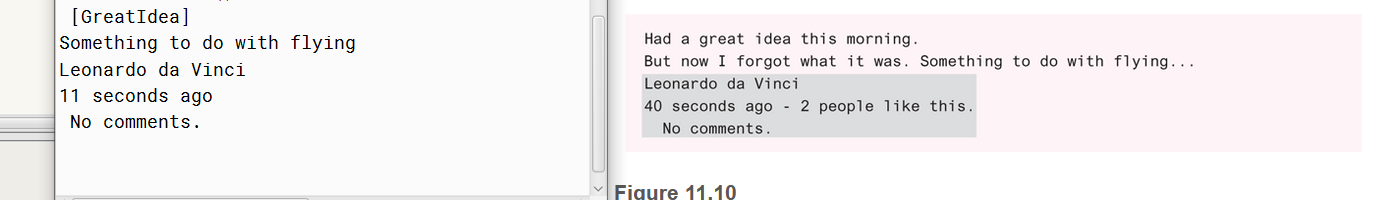
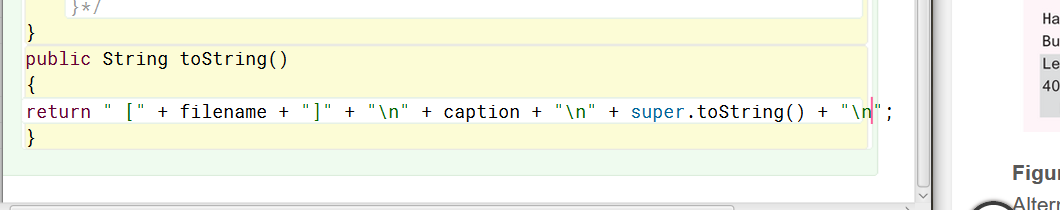
ToString is a method of the object class. ToString can be used to return a string representaition of an object. The toString method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. Ie... (getClass().getName() + '@' + Integer.toHexString(hashCode())). It does not take any parameters.

11.5



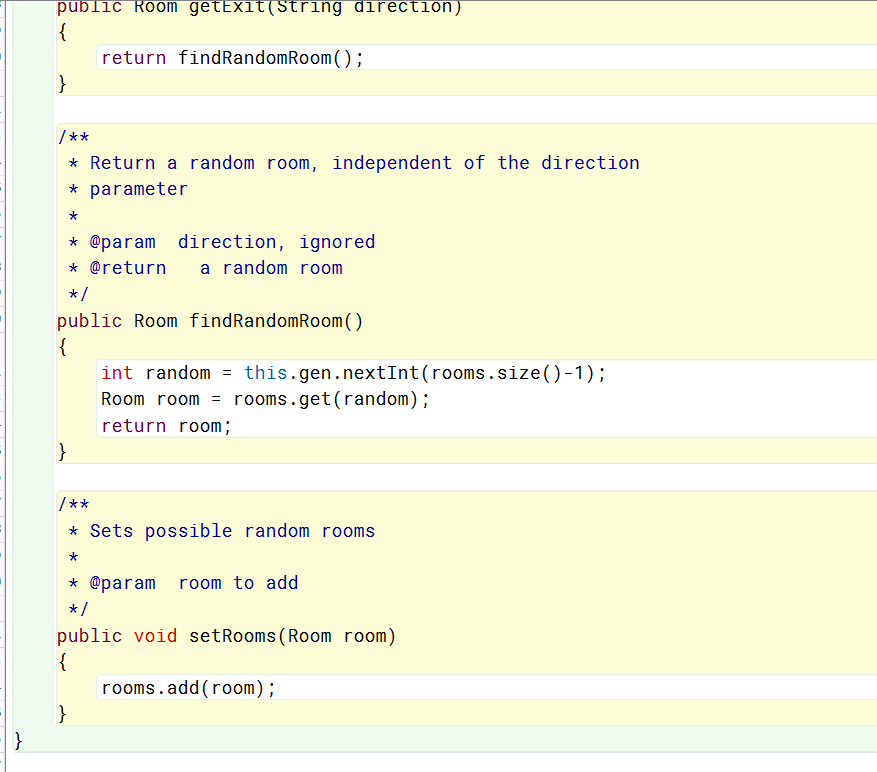
11.6

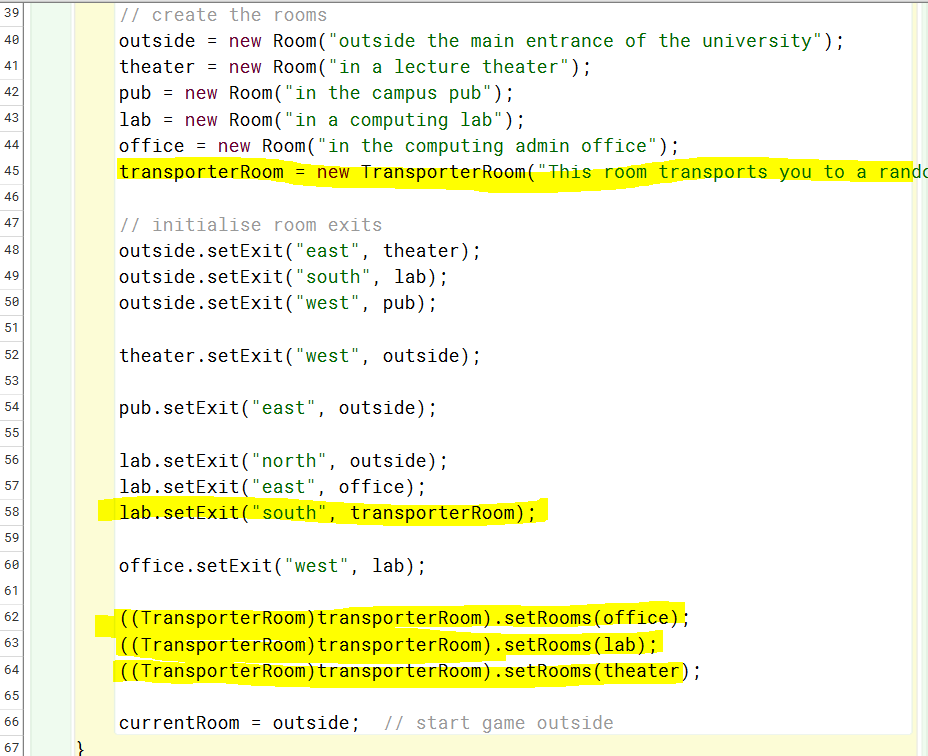


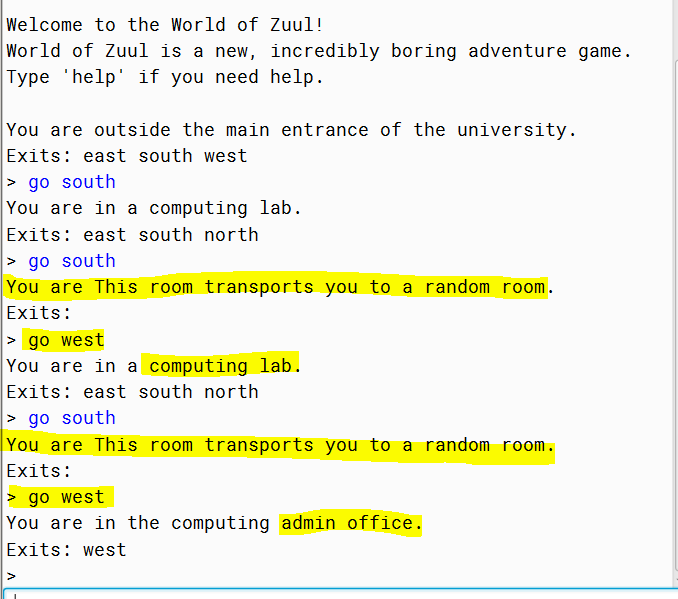


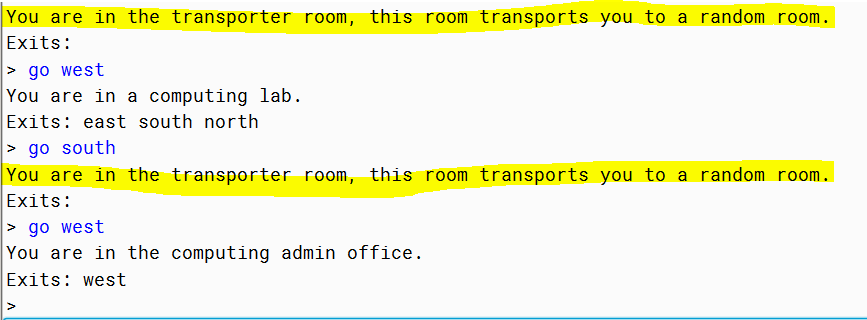
11.8









Fixed typo\*\*

11.9

Just as we did with the room and the transporter room, we could use inheritance to create different types of beings. We could have a super class of being that could have subclasses of monster, player, or friendly creature. Where being would be the parent class and player, monster and friendly creature would all be siblings.

11.10

I feel like unless there was an item with an awful lot of creature like qualities or shared attributes. No, item should have its own separate super-class since it has many differing attributes to a creature. I would say there should be one super-class Item as well as another super-class Creature in this case. Although, all objects do go back to the super-class Object. Making playerOne and itemBook at least cousins where item and creature would be siblings with a parent Object.

11.11

Since Device dev has been assigned the value of a new Printer. The getName() method would have to be a member of the Device class for this to work as written. Otherwise, if getName() was a member of the Printer class it could still be used if written like this: ((Printer)dev).getName();

11.12

In this case because dev is instantiated as a Device object first this will be the getName() method that will run. Unless written: ((Printer)dev).getName();

11.13

So, no this will not cause any problem with compiling. Although, this statement string s = st.toString() doesn’t do much on its own. S is just now a String representation of the Student object st. It would look something like, “Student@12d6f87”.

11.14

Yes, this statement System.out.println(st) will compile. The String representation of Student object st will be displayed on the screen. It would look something like this: “Student@12d6f87”.

11.15

Yes, this should compile just fine and will print the name of each student if the toString() method was overridden to return the value in the name field in class student.

11.16

T x = new D;