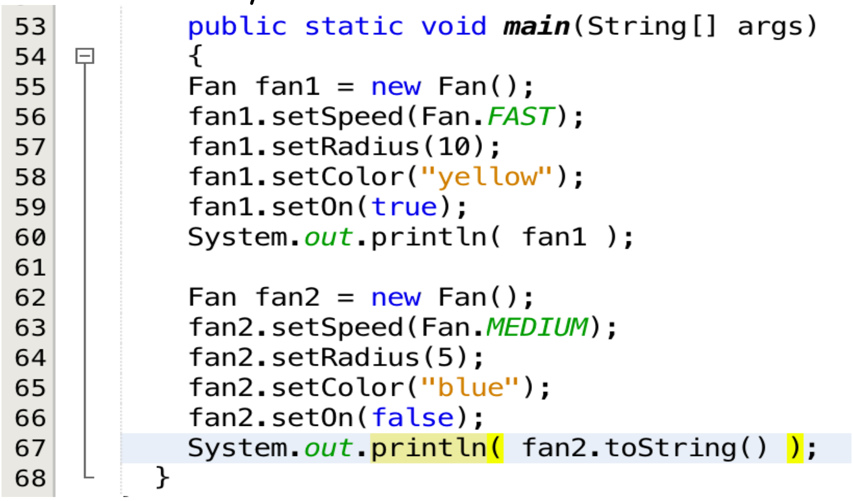
ADSV 2420, Advanced Programming I
Lab 3 -- Classes, Arrays of Objects

1. Create a new project lab3. Use Netbeans to create the code and Javadoc for the *Fan class* specifications:

The Fan class

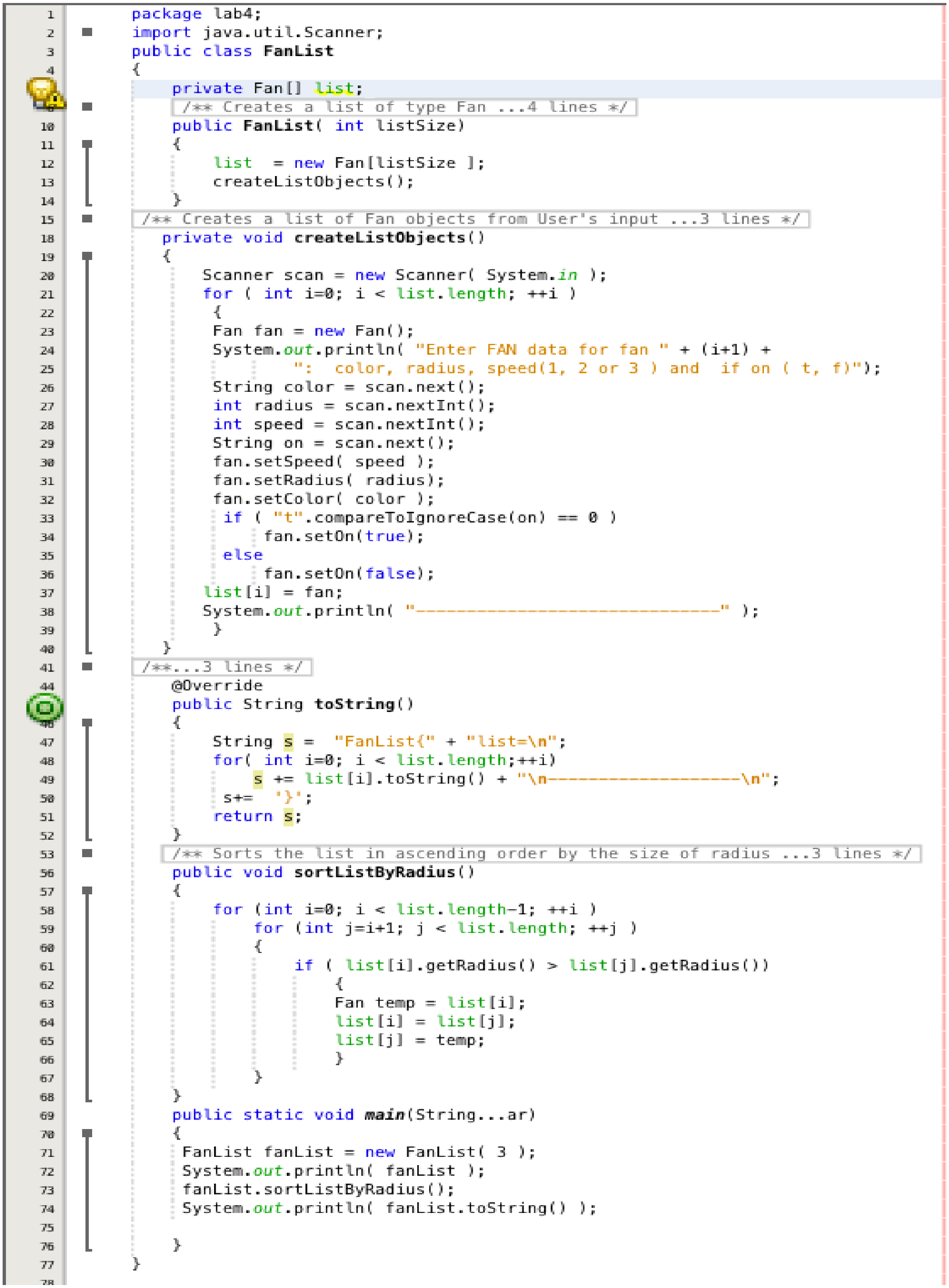
a) Three public constants named SLOW , MEDIUM , and FAST with the values 1 , 2 , and 3 to denote the fan speed (static AND FINAL NEVER initialized in the constructor).
b) A private int data field named *speed* that specifies the speed of the fan (the
c) default is SLOW).
d) A private boolean data field named *on* that specifies whether the fan is on (the
e) default is false).
f) A private double data field named *radius* that specifies the radius of the fan
g) (the default is 5).
h) A string data field named *color* that specifies the color of the fan (the default
i) is blue).
j) The accessor(getters) and mutator(setters) methods for all four data fields(skip public)..
k) A no-arg constructor that creates a default Fan.
l) A method named toString() that returns a string description for the fan.
m) If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string “fan is off” in one combined string.
n) Add the method main to your Fan calls to test it as shown:



```
    public static void main(String[] args)
    {
    Fan fan1 = new Fan();
    fan1.setSpeed(Fan.FAST);
    fan1.setRadius(10);
    fan1.setColor("yellow");
    fan1.setOn(true);
    System.out.println( fan1 );

    Fan fan2 = new Fan();
    fan2.setSpeed(Fan.MEDIUM);
    fan2.setRadius(5);
    fan2.setColor("blue");
    fan2.setOn(false);
    System.out.println( fan2.toString() );
  }
```

2.Create the **class FanList** that creates and array of Fan and uses it. Understand it or ASK!!!



```java
package lab4;
import java.util.Scanner;
public class FanList
{
    private Fan[] list;
    /** Creates a list of type Fan ...4 lines */
    public FanList( int listSize)
    {
        list  = new Fan[listSize ];
        createListObjects();
    }
/** Creates a list of Fan objects from User's input ...3 lines */
   private void createListObjects()
   {
       Scanner scan = new Scanner( System.in );
       for ( int i=0; i < list.length; ++i )
        {
        Fan fan = new Fan();
        System.out.println( "Enter FAN data for fan " + (i+1) +
                ":  color, radius, speed(1, 2 or 3 ) and  if on ( t, f)");
        String color = scan.next();
        int radius = scan.nextInt();
        int speed = scan.nextInt();
        String on = scan.next();
        fan.setSpeed( speed );
        fan.setRadius( radius);
        fan.setColor( color );
         if ( "t".compareToIgnoreCase(on) == 0 )
             fan.setOn(true);
         else
             fan.setOn(false);
       list[i] = fan;
       System.out.println( "------------------------------" );
        }
   }
/**...3 lines */
    @Override
    public String toString()
    {
        String s =  "FanList{" + "list=\n";
        for( int i=0; i < list.length;++i)
            s += list[i].toString() + "\n-------------------\n";
         s+=  '}';
        return s;
    }
   /** Sorts the list in ascending order by the size of radius ...3 lines */
    public void sortListByRadius()
    {
        for (int i=0; i < list.length-1; ++i )
            for (int j=i+1; j < list.length; ++j )
            {
                if ( list[i].getRadius() > list[j].getRadius())
                    {
                    Fan temp = list[i];
                    list[i] = list[j];
                    list[j] = temp;
                    }
            }
    }
    public static void main(String...ar)
    {
     FanList fanList = new FanList( 3 );
     System.out.println( fanList );
     fanList.sortListByRadius();
     System.out.println( fanList.toString() );

    }
}
```

3. **The Stock class**

Similarly to the implementation of the Fan class, use Netbeans Insert-Code and implement the class Stock that contains:

a) A string data field named *symbol* for the stock's symbol.
b) A string data field named *name* for the stock's name.
c) A double data field named *previousClosingPrice* that stores the stock
d) price for the previous day.
e) A double data field named *currentPrice* that stores the stock price for the
f) current time.
g) The accessor(getters) and mutator(setters) methods for all four data fields.
h) A constructor that creates a stock with the specified symbol and name.
i) A method named getChangePercent() that returns the percentage changed
j) from previousClosingPrice to currentPrice .
k) A method named toString() that returns a string description for the Stock.
l) Add the method main to your and write code that creates a Stock object with the stock symbol ORCL , the name Oracle Corporation , and the previous closing price of 34.5 . Set a new current price to 34.35 and display the price-change percentage. Then, display the description of the class using the toString() method

4. Create a class **StockList**

Its behavior should be identical to FanList. However, instead of the method sortByRadius() you should have a method *sortByChangeInPercentage.* Add its main method inside of it : create the list, print the list, sort the list, and then print it again as in the main of FanList.

5. Study the JAVADOC (and textbook) of the **GregorianCalendar** class in the java.util package, which you can use to obtain the year, month, and day of a date. The no-arg constructor constructs an instance for the current date, and the methods get(GregorianCalendar.YEAR) , get(GregorianCalendar.MONTH) , and get(GregorianCalendar.DAY_OF_MONTH) return the year, month, and day.

Inside main of class Lab3

a) Display the current year, month, and day.
b) The GregorianCalendar class has the setTimeInMillis(long) , which can be used to set a specified elapsed time since January 1, 1970. Set the value to 1234567898765L and display the year, month, and day.