To insure that you receive partial credit whenever possible, be sure to show all your work. Remember: the exam is closed book, and cellphones, calculators, sliderules, notes, talking, etc., are not permitted. All parts of questions are worth 5 points, unless otherwise noted.

The FishTank class is given at the end of this exam.

a. Create a driver class, TankDriver, that includes a statement that creates a FishTank object called myTank, with owner Jill, capacity 20, and that is 40% filled.

```java
public class TankDriver {
    public static void main(String[] args) {
        FishTank myTank = new FishTank("Jill",20,.4);
    }
}
```

b. The first statement in the FishTank constructor is:  
   `this.owner=owner;`. In a few sentences at most, explain what this means.

“this” refers to the object to be created when the class’s constructor is called

c. Add a method to the FishTank class called gallonReport, which prints to the console the integer number of gallons of water in the tank. You may either round this amount up or down. Thus if the number of gallons, computed exactly, is 11.7, your code may print either 11 or 12.

```java
public int gallonReport(){return (int)(capacity*fractionFilled);}
```

d. Add a static method called greatCap to the FishTank class. This method should be passed an array of FishTank objects as a parameter, and should return the object in the array with the greatest capacity. In case of ties, any FishTank object with greatest
capacity will do. If tanks is an array of FishTanks in the driver class you’ve created above, write a method call to the greatCap method applied to the tanks array, and assign the result of that call to a FishTank variable called bigTank. (Hint: to make a method static, begin the header line with “public static…”)

```java
public static FishTank greatCap(FishTank[] tks){
    FishTank big = tks[0];
    for(FishTank f : tks) if (f.capacity>big.capacity) big = f; return big;
}

FishTank bigTank = FishTank.greatCap(tanks);
```

e. (15) Using inheritance, extend the FishTank class to a HeatedFishTank class. This class should add one new boolean attribute, hasHeater. When this attribute is false, the FishTank does not have a heater. Your extension should add a get and a set method for this attribute. The HeatedFishTank constructor should take four parameters, owner, cap, frac, and heated.

```java
public class HeatedFishTank extends FishTank{
    private boolean hasHeater = false;
    public HeatedFishTank(String owner, int cap, double frac, boolean heated){
        super(owner,cap,frac);
        hasHeater = heated;
    }

    public boolean getHasHeater(){return hasHeater;}
    public void setHasHeater(boolean heat){hasHeater = heat;}
}
```

f. Suppose you make a HeatedFishTank object, t, in the driver class, and then you add this statement:

```java
System.out.println(t.toString());
```

Explain what happens when the driver executes this statement.

Since there’s no explicit toString method in FishTank or HeatedFishTank, the Object class toString method is used. And it returns: HeatedFishTank@<someMemAddress>. That memory address is the location of t in memory.
g. Finally, write a static method in the `HeatedFishTank` class called `avCap`, which is passed an array of `HeatedFishTank` objects, and reports the average capacity of the tanks in the array that have heaters. If none of the tanks has a heater, your method should return 0.0.

```java
public static double avCap(HeatedFishTank[] tks){
    double av = 0.0; int total = 0; int ct = 0;
    for (HeatedFishTank f : tks)
        if (f.hasHeater) {ct++; total+=f.getCapacity();}
    if (ct > 0) av = (double)total/ct;
    return av;
}
```

2. The `GreetPanel` and `WordDriver` classes are given at the end of this exam.
   a. Sketch the `DisplayWindow` when you run the `WordDriver/GreetPanel` application.

   The window displays a button labeled “Greet” at top center, and the word “hello” (partially obscured by the button).
   b. The application prints “hello” when you click the Greet button. Describe what happens when you click the button multiple times.

   You see “hello” written in a column, a little lower with each click.
   c. When you run `WordDriver`, what object serves as the listener for the greeting button?

   The panel object p in the driver.
   d. If you change the call to `drawstring` in `paintComponent` to `g.drawString("hello", vPos, 200)`, what happens when you click the “Greet” button repeatedly?

   “hello” written repeatedly horizontally, shifts right with each click.
e. (15) Now add a new “flip” button to the WordDriver/GreetPanel application. The flip button works this way. The application begins, as before, by printing “hello” whenever you click on Greet. Then, when you click on “flip”, the application will print “goodbye”, and will continue printing “goodbye” on every “Greet” click, until you click flip again. Thus the flip button switches the displayed greeting back and forth between hello and goodbye.

```java
class GreetPanel extends JPanel implements ActionListener{
    private JButton greeting = new JButton("Greet");
    private JButton flip = new JButton("Flip");
    boolean hello = true;
    private int vPos = 20;

    public GreetPanel(){
        setPreferredSize(new Dimension(700,300));
        this.add(greeting);
        greeting.addActionListener(this);
        this.add(flip);
        flip.addActionListener(this);
    }

    public void paintComponent(Graphics g){
        super.paintComponent(g);
        if (hello)
            g.drawString("hello",200,vPos);
        else
            g.drawString("goodbye",200,vPos);
    }

    public void actionPerformed(ActionEvent e){
        if (e.getSource() == greeting){
            vPos += 20;
            repaint();
        } else if (e.getSource() == flip){
            vPos += 20;
            hello = !hello;
            repaint();
        }
    }
}
```
3. Is there anything wrong with this method definition – and if so, what? In particular, does it compile? Does it compute the average properly?

```java
public int averageOfTwo(int a, int b)
    {return (a + b)/2;}
```

Compiles, but doesn’t take average of numbers, since it does integer division. Would work if denominator were 2.0

4. Suppose you want the FishTank class to implement the Comparable interface, with comparisons based on capacity: one tank precedes another if the first has smaller capacity. Do this implementation (note: you don’t need to rewrite the entire class – just provide the relevant changes / additions. Also: you need to implement the compareTo method here: public int compareTo(Object other){..})

Change header to read:
public class FishTank implements Comparable{

then add compareTo method:
```java
    public int compareTo(Object other){
        FishTank f = (FishTank)other;
        return(this.capacity - f.capacity);
    }
```

5. In a few sentences at most, explain what an abstract class is.

A class that can’t be instantiated: public abstract class C{blah blah}
Abstract classes typically have at least one abstract method:
```java
    public abstract void processLine();  // an abstract method in LineReader
```

6. Rewrite the for loop, below, as a while loop:

```java
for (int j = 0; (( j < 10) || (j % 2 == 0)); j++)
    System.out.println(j);
```
public class FishTank{
    private String owner;
    private int capacity; // in gallons
    private double fractionFilled; // fraction of tank filled. 0.5 means half filled

    public FishTank(String owner, int cap, double frac){
        this.owner = owner;
        capacity = cap;
        fractionFilled = frac;
    }

    public void setOwner(String who){ owner = who; }
    public String getOwner(){return owner; }
    public int getCapacity(){return capacity; }
    public double getFractionFilled(){return fractionFilled; }
}

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import java.awt.geom.*;

public class GreetPanel extends JPanel implements ActionListener{

    private JButton greeting = new JButton("Greet");
    private int vPos = 20;

    public GreetPanel(){
        setPreferredSize(new Dimension(700,300));
        this.add(greeting);
        greeting.addActionListener(this);
    }

    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawString("hello",200,vPos);
    }

    public void actionPerformed(ActionEvent e){
        if (e.getSource() == greeting){
            System.out.println("Greeted");
        }
    }
}
vPos += 20;
    repaint();
}
}
}

public class WordDriver{

    public static void main(String[] args){
        DisplayWindow d = new DisplayWindow();
        GreetPanel p = new GreetPanel();
        d.addPanel(p);
        d.showFrame();
    }
}
}