Lecture topic: arrays. They're objects; declarations; 0-based indexing system; the length constant; it's NOT length(); Off-by-one errors.

**Some problems:**

Arrays:
1. Declare an array of 12 booleans called boo
2. Declare an array of 19 doubles called nums
3. Declare an array of 5 Infants called littleOnes

```java
public class CoffeeMug{
    private String who;  // whose mug is it?
    private int capacity;  // oz held in mug
    private boolean full;  // is cup full or not  --only full or empty

    public CoffeeMug(String w, int cap, boolean full){
        who = w;
        capacity = cap;
        this.full = full;
    }
    public String getWho(){ return who;}
    public int getCapacity(){ return capacity;}
    public boolean getFull(){ return full;}
    public void setWho(String other){who = other;}
    public void setFullStatus(boolean isFilled){full=isFilled;}
}
```

A warm-up: make a mug for jim, holds 8 oz, is empty. Now fill it up (all this in a driver class)

2. Create an array of 5 mugs called myMugs.

Now: fill myMugs with cups that are all jill's, first 3: 10 oz, last 2: 12 oz.  -- all full
Now make the 0th mug belong to wilma
Next: empty them all
Next: fill them all
Next: print a report about each mug-- owner/capacity/full?

3. Complete static method below called fullCount, that's passed an array of mugs, and returns number that are full.

```java
public static int fullCount(CoffeeMug[] m){
    int ct = 0;
    ...  // what goes in here??
    return ct;}
```

4. Write a static method called allMine that's passed an array of CoffeeMug objects and a String name and makes the owner of all of the mugs equal to the value of the name parameter.

5. Write a static method called fullCapacity that's passed an array of mugs, and returns the total number of ounces of fluid that all of the mugs, taken together, can hold. Alter this to create a method that calculates the average capacity. Now alter to find mug with highest capacity.
The NumPal problem, beginning with the driver:

```java
import javax.swing.JOptionPane;
public class NumPalDriver{
    public static void main(String[] args){
        String start = JOptionPane.showInputDialog("enter positive start value");
        NumPal p = new NumPal(start);
        int ctr = 0;
        while (!p.pal() && (ctr < 10)){
            System.out.println(p.toString());
            System.out.print("new sum: ");
            System.out.println(p.getCur() +" + " + p.getRev() + " = " + (p.getCur() + p.getRev()));
            p = p.next();
            ctr++;
        }
        System.out.println("final value: " + p.getCur());
        System.out.println("number of steps: " + ctr);
    }
}
```

What methods need to be written inside NumPal? What are their return types?

Diagram below shows NumPal transition from left object to right object via next method. How is this done? Because of the way next() works, program execution works via the evolution of a stream of NumPal objects, each pointed to (referenced by) p.

Important idea: how can a constructor have just one parameter, but be a constructor for a method with 4 attributes? (How do you load values into those attributes? And – important – in what order?)