1. Write a complete, stand alone program in a single class called `Names`, which works this way.

Include in your program this array of names:

```java
String[] names = {
```

Your program should read in a non-negative `int` value - say `n` - from the keyboard, and then print one of the names from the array using the value of `n` as the index into the array. If `n` is 0, for example, your program should print `Jo`; if `n` is 1, print `Jim`, if `n` is 2, print `Dana`, and so forth. The name you print out is determined by the array index you enter. If `n` is larger than 9, then the name you print is the one determined by the last digit of `n`. Thus if `n` is 911 (or 12341) print `Jim` -- the name with index 1; if `n` is 1002 print `Dana` - the name with index 2. In general, use the mod operator `"%"` to figure the last digit, and then select that name from the list. (Thus 911%10 = 1, 12345%10 = 5, and so forth).

```java
import java.util.*;
public class Names{
    public static void main(String[] args){
        String[] names = {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        n = n % 10;
        System.out.println(names[n]);
    }
}
```

A final requirement: your solution must make explicit use of the `names` array (in other words, don't solve the problem by using 10 if statements).

You are in the shed business - you rent and sell utility sheds (e.g., garden sheds) to people and businesses throughout Western MA. You have decided to automate some aspects of your business. This problem is intended to get you started. Your first stab at a `Shed` class is given below, along with an associated driver class.

```java
public class Shed{
    private String owner;
    private int size;

    public Shed(String who, int area){
        owner = who;
        size = area;
    }

    public String getOwner(){return owner;}
    public int getSize(){return size;}

    public String toString()
    {return (owner: " + owner + ", size: " + size);}
}
```

Here is the driver. Notice that so far, it contains no statements.

```java
public class ShedDriver
{
    public static void main(String[] args)
    {
        // driver statements go here
    }
}
```
2. Add a setOwner method to the Shed class. This method should take one parameter, a String, and should reset the owner field of the class to this value.
   ```java
   public void setOwner(String s){owner = s;}
   ```

3. Many of your sheds are unsold. To reflect this fact, add a second constructor to the Shed class, which by default sets the owner field of the constructed object to "unsold". This constructor should have the following header line:
   ```java
   public Shed(int area){ owner = "unsold"; size = area;}
   ```

4. Some sheds you sell are big; others are small. A big shed is larger than 250 square feet. Add a method to the Shed class called bigShed, which takes no arguments and which returns true if the calling Shed object has size larger than 250; otherwise the method should return false.
   ```java
   public boolean bigShed(){return (this.area > 250); }
   ```

5. Some sheds you install have electricity, while others do not. Create a subclass of the Shed class using inheritance called ElectricShed. This class should have one additional attribute, electric, of type boolean. When the attribute electric is true, the shed has electricity; if electric is false, the shed does not have electricity. The class should have just one constructor, with three parameters, owner, size, and electrified (boolean). Be sure to use super in your code in the derived class constructor. Also, be sure to include a getElectric method with no arguments, which returns a boolean value that tells if the calling object is electric.
   ```java
   public class ElectricShed extends Shed{
   private boolean electric;
   public ElectricShed(String owner, int size, Boolean e){
   super(owner, size);
   Electric = e;
   }
   public Boolean getElectric(){return electric;}
   }
   ```

6. Finally add one additional method to the ElectricShed class, a static method called electricOwners. This method should be passed an array of ElectricShed, and should print to the console the names of all owners who own sheds that have electricity. If one person owns more than one electric shed, it's ok to display their names multiple times. NOTE: Assume the ElectricShed class has a method called isElectrified() that returns the value of the variable electric.
   ```java
   public static void electricOwners(ElectricShed[] sheds){
   for(ElectricShed e : sheds)(if e.electric)System.out.println(e.owner);
   }
   ```

7. Definition of the Shed class is above. Consider the following for loop, where sheds is an array of Shed objects
   ```java
   for(Shed s : sheds)
   if (s.getSize() > 250) System.out.println(s.getOwner());
   ```
   Rewrite this code as a while loop.
   ```java
   int j = 0;
   while(j < sheds.length){
   If(sheds[j].getSize() > 250) System.out.println(sheds[j].getOwner());
   j++
   }
   ```

8. Consider the following definition for a method called h.
   ```java
   static int h(int a, int b){ if (a > b) return 0; else return(b + h(a,b-1)); }
   ```
   In a sentence or two at most, tell what the method h does. Hint: what value is returned when a is 2 and b is 3? When a is 2 and b is 4? (Do NOT simply narrate the code step by step; the
method does a recognizable chore - what is it?).

Sums numbers from a to b inclusive

9 If you divide 12 by 5 repeatedly, it takes two divisions to reach 0: 12/5 = 2, 2/5 = 0. If you divide 121 by 5 repeatedly, it takes three divisions to reach 0: 121/5 = 24, 24/5 = 4, 4/5 = 0. Write a while loop that calculates the number of divisions by 5 it takes to reach 0, starting from 344,699.

```java
int ct = 0; int k = 344699;
while (k > 0){
    ct++;  k = k / 5;
}
System.out.println(ct);
```

10 Suppose d is a String that consists of all digits. For example, d might reference "1239", "338", or "222". Write several Java statements that report whether an arbitrary such string of digits d represents an odd or even number. Thus if d is "1239" your code should print d is odd. On the other hand, if d is "222" your code should print d is even. In your code you may assume that d has been defined to be a String, and that it has already been assigned a string of digits.

```java
Int k = Integer.parseInt(d);  if (d % 2 == 0) System.out.println("yes"); else System.out.println("no");
```

11 Consider the following definition for the method what.

```java
Random r = new Random();
int s = 0;
int ct = 0;
while (s < 1000){  s = s + r.nextInt(10);  ct++; }
System.out.println(ct);
```

In a sentence or two at most, tell what this loop does. (Do NOT simply narrate the code step by step; the code performs a recognizable chore - what is it?).

Generates random numbers 0-9; reports how many it takes to sum to 1000 or more

12 Suppose the following method has been added to the Shed class, above

```java
public static String huh(Shed[] sheds){
    int start = 0;
    for(int j = 1; j < sheds.length; j++)
        if (sheds[j].size < sheds[start].size) start = j;
    return (sheds[start].owner); }
```

In a sentence or two at most, tell what the method huh does. (Do NOT simply narrate the code step by step; the code performs a recognizable chore - what is it?)

Reports owner of smallest shed