Here is the definition of the Farm class:

```
public class Farm{
  private String owner;
  private int acres; // farm size in acres

  public Farm(String who, int acres){
    owner = who;
    this.acres = acres;
  }

  public String getOwner(){return owner;}
  public int getAcres(){return acres;}
}
```

1. (10) The hectare is the metric equivalent of the acre in English measure. One hectare = 2.47 acres (so a hectare is more than twice as large as an acre). Add a getHectares method to the Farm class. It should take no arguments, and should return a double value – the size of the calling Farm object in hectares.

```
public double getHectares(){return (acres/2.47);}
```

2. (10) Now add a setOwner method to the Farm class. This method should take a new owner name as an argument, and should change the value of the calling Farm object's owner field.

```
public void setOwner(String n){owner = n;}
```

3. (10) Now add a method to the Farm class called averageAcreage, which is static, is passed an array of Farm objects, and which returns a double value, the average number of acres in all the farms in the array.

```
public static double avAcre(Farm[] farms){
        double sum = 0.0;
        for(Farm f : farms)sum += f.acres;
        return (sum/farms.length); }
```

4. (15) Now add a new static method to the Farm class called smallFarm. It is passed an array of Farm objects, and returns the Farm object in the array with the smallest number of acres (in case of ties any Farm with smallest acreage will do).

```
public static Farm small(Farm[] farms){
        Farm s = farms[0];
        for(Farm f : farms) if (f.acres < s.acres) s = f;
        return s; }
```

5. (10) Suppose farmArray is an array of Farm objects. Rewrite the while loop below using a for-each (extended for) loop:

```
int k = 0;
while(k < farmArray.length){
 System.out.println(farmArray [k].getOwner());
 k++;
}
```

```
for (Farm f : farmArray) SOP(f.getOwner());
```

6. (15) Now suppose you are interested in a particular kind of farm, namely a dairy farm. Create a new class, DairyFarm, which is a subclass of Farm, and which adds two new attributes to the Farm class, **cowCount**, an int (the number of cows on the farm), and **milkProduction**, also an int (the number of pounds of milk the farm produces per day). The DairyFarm constructor should be passed four parameters, the owner, the number of acres, the cow count, and the milk production value.

Besides a constructor, your class should include get and set methods for cowCount and milkProduction.

```
public class DairyFarm extends Farm{
        private int cc;
        private int mp;

public DairyFarm(String w, int a, int c, int m){
        super(w,a);
        Cc = c; mp = m; }

public int getCC(return cc;}
public void setCC(int c){cc = c;}
Etc.
}
```

7. (10) The method mystery, below, is recursive.

```
public static void mystery (String s){
  if (s.length() > 0){
   System.out.print(s.charAt(0));
   System.out.print(s.charAt(0));
   mystery(s.substring(1));
  }
 else System.out.println(); }
```

What is the output of the method on input string "Bark"?

BBaarrkk

8. (20) Write a complete one class program called ReverseRandom that does the following: your program should first read in an integer, say N, from the keyboard, which you may assume is a positive value; then your program should generate N random numbers – integers – between 0 and 10 (both 0 and 10 should be possible); these values should be stored in an array. Finally, your program should print out the array values in a column, in reverse order (the last one first, then the next-to-last one, and so forth).

```
Import java.util.*;
Public class RR{
        Psvm(String[] args){
                Random r = new Random();
                Scanner s = new Scanner(System.in);
                int n = s.nextInt();
                int[] nums = new int[n];
                for(int j = 0; j < n; j++) nums[j] = r.nextInt(11);
                for(int j = n-1; n>=0; n--)SOP(nums[j]);
}}
```