Question 3 [11 Marks]

Consider the following (partial) implementation of a system allowing farmers to manage their flock of animals. Note that a HashSet is an implementation of the Set interface that is backed by a hash table. Appendix 1 provides segments of the Java 1.5.0 API that may be useful for this question.

```
public class Flock
{
      Set<Sheep> aFlock = new HashSet<Sheep>();

      public void add ( Sheep a ) { aFlock.add( a ); }
      public boolean present ( Sheep a ) { return aFlock.contains(a); }
}

class Sheep
{
      private String aName;
      private Date aBirth;    // Date/Time of birth
      public Sheep(String name) { aName = name; aBirth = new Date();}
      public void rename(String name) { aName = name; }
      public String getName() { return aName; }
      public void vocalize() { System.out.println("Baahaahaaah"); }
}
```

a) Given the current implementation, if you add the same Sheep object multiple times to a Flock, it will only be added once. Explain why (provide specific details of the mechanism at work here). [2]

b) At some points scientists discover how to clone sheep. Consider that a cloned sheep is "born" at the time when it is cloned. The system is thus modified by making the Sheep class implement the Cloneable interface, and by adding the following method to class Sheep:

```
public Object clone() {
   try { return super.clone(); }
   catch( CloneNotSupportedException e ) { return null; }
}
```

Briefly explain what happens when this method is called, and why this behavior is probably incorrect with respect to the representation of cloned sheep. [2]

c) Rewrite the clone method so that the problem identified in (c) is fixed. [2]

d) The users of the system find it too inconvenient to allow multiple sheep with the same name in a flock. The system is thus changed so an equals and hashCode methods are implemented in Sheep so that equals simply compares sheep names, and hashCode simply returns aName.hashCode(). Why does this make the rename method problematic? [2]