Quiz 1 Spring, 2009

Your name: 
Your Student Number: 

There are two kinds of eating behavior or strategy: Herbivorous and Carnivorous. The corresponding two eat() methods return “I’m nibbling leaves” and “I’m eating flesh” respectively.

```java
public interface EatingBehavior {
    public String eat();
}

public class HerbivorousEating
{
}

public class CarnivorousEating
{
}
```

Lions observe Deer and Deer observe Lions. All animals have a move() method. The move method prints (by System.out.println) a String saying “I am a lion moving” or “I am a Deer moving”. But, also, whenever a Lion moves() it causes any Deer that is observing that Lion to move(). (But if a Lion moves it doesn’t cause other Lions to move. Similarly, if a Deer moves, any Lion that is observing the Deer will “notice” it and move() as a result (but one Deer moving doesn’t cause another Deer to move).

1) [10%] First, complete the driver code (adding what it takes to get bambi and leo to observe one another (hint: use addObserver).

```java
Deer bambi = new Deer();
Lion leo   = new Lion();
// add code below to get them to observe each other:

bambi.move();
```

1) [10%] What would be the SOP output? Give a few lines of output that results from bambi.move() and explain your answer (you could use a sequence diagram here to very efficiently explain what happens). Does it terminate?
3) [60%] finish writing **Animal** and the subclasses **Deer** and **Lion** using the **Strategy Pattern** for their `eat()` method and the Observer pattern for the above effect of one animal causing another to move. Remember the Java interface `Observer` is:

```java
public interface Observer { public void update(Observable o, Object obj); }
```

And remember that any Observable needs to setChanged() and notifyObservers() whenever it wishes to have its observers notified.

```java
abstract public class Animal extends implements {

    public void eat() {
        abstract public void move();
    }

public class Deer
```

```java
public class Lion
```
4. [10%] Create a **UML class diagram** showing all of the hierarchies, classes, and interfaces: Animal, Deer, Lion, Observable, Observer, EatingBehavior, HerbivorousEating, and CarnivorousEating. Show the “is-a”, “has-a” and “implements-a” connections and the major public methods (include visibility modifiers).

5. [10%] Create a **UML sequence diagram** for the code snippet:

```java
Deer bambi = new Deer();
Lion leo   = new Lion();
// add code below to get them to observe each other:
(plus your additional code that you wrote here)

bambi.move();
```