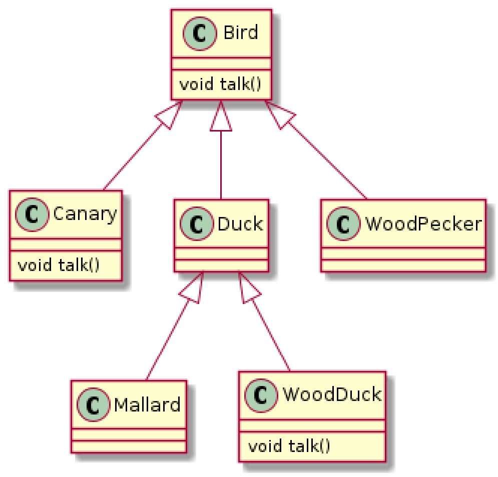
1. Consider the following class relationship diagram:



   (a) In the following declaration/assignment statement, identify the *apparent* type of the variable decoy and the *actual* type of its value at the end of the *assignment*. (10)

   ```
   Bird decoy = new WoodDuck();
   ```

   (b) After `Bird mmm = new Mallard();`, which version of `talk()` will be called with `mmm.talk();`? (5)

   (c) With the following declaration/assignments

   ```
   Bird b_x = new Canary();          Bird b_z = new WoodPecker();
   Bird b_y = new Mallard();         Duck d_x;
   Canary c_x;                       Mallard m_x new Mallard();
   ```

   Explain how you know whether or not each of the following assignments will compile. (These are **independent** questions: each takes place right after the declarations above.)

   i. `c_x = b_y;` (2)

   ii. `c_x = b_x;` (2)

   iii. `m_x = b_y;` (2)

   iv. `d_x = m_x;` (2)

   v. `b_z = new Canary();` (2)

2. Declare an *array* variable that can hold up to 73 of the any species of bird. Also declare an **`int`** variable to hold the actual number of elements in the array. (10)

3. Assuming that there is at least one bird in your array, write the *simplest* code you can to take the bird with the **highest** index out of the array. (10)

4. Assuming that there is at least one bird in your array, write code to take the bird with the **lowest** index out of the array. (10)

5. Write one line of code that would appear *inside* of the `Bird` class, declaring a `String` **field** called `name` such that any class in the diagram above could read or write `name` but no other class in your project can. You only get one (1) line. (5)

Points earned: ________
out of a possible 60 points

6. Consider: (10)

```
class LinkedList {
  class LinkedListNode {public String data; public LinkedListNode next;}
  public LinkedListNode head;
  public append(String valueToAppend) ...
  public append(LinkedListNode curr, String valueToAppend) ...
}
```

   Describe how the compiler (and you) can figure out which version of `append` is called in the following code:

```
LinkedList qqq = new LinkedList();
qqq.append(qqq.head, "Bob the Builder");
```

   It is not enough to just say which one is called.

7. Back to the `Birds`, describe how Java (and you) can figure out which version of `talk` is called in the following code: (10)

```
Bird rrr = new WoodDuck();
rrr.talk();
```

   It is not enough to just say which one is called.

8. Pretend `LinkedList.append(String)` works to append elements at the end of the list. Draw a **picture** of the contents of the list after the insertion of the following words, in order: Lima, Moscow, London, Tokyo, Melbourne, Phoenix, Nairobi (10)

9. Picture the contents **after** removing Tokyo from the list. (10)

Points earned: ______
out of a possible 40 points