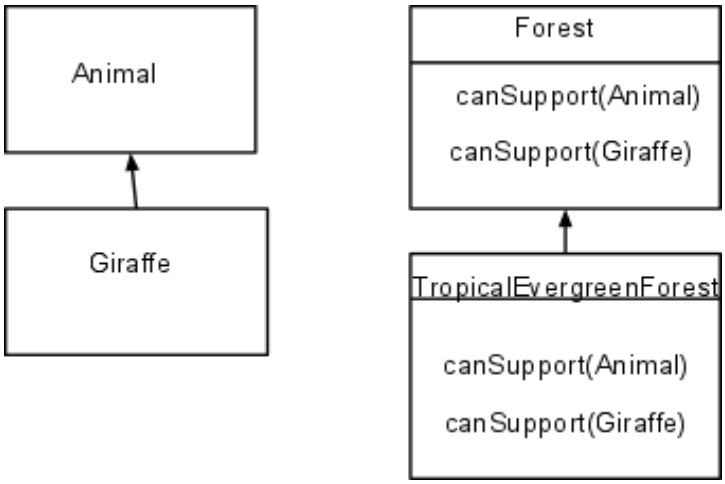
CS 350 Homework 4

1. Exercise 4, Chapter 6. Ports are implemented using declarative, secure, unbundled ADT. Implement Port using explicit state, secure, unbundled ADT in Oz.

2. Explain how to implement protected methods in the Java sense, using Oz. Explain with the help of a reference hierarchy.

3.Visitor Pattern and Double Dispatch: One disadvantage of the method dispatch in many OO languages (e.g. Java, Smalltalk and C++ virtual functions) is that the dynamic method dispatch is done based only on the runtime type of the object. The method lookup in overloading is based on compile time type of the arguments.

For example, consider the following inheritance hierarchy.



If, we have code that does the following in a Java-like syntax

```
Animal g = new Giraffe(); //declared type of g is Animal, runtime type is Giraffe.
Forest f = new TropicalEvergreenForest(); //declared type of f is Forest,
                                          //runtime type is  TropicalEvergeenForest
f.canSupport(g); //will call TropicalEvergreenForest.canSupport(Animal)
                 //instead of TropicalEvergreenForest.canSupport(Giraffe)
```

Using the visitor pattern (see http://en.wikipedia.org/wiki/Visitor_pattern) to implement the above hierarchy, with double dispatch, in Java/C++.

4. A Chunk is a limited record, which supports only two operations: one to create a chunk -
C ={Chunk.new anyrecord(a:fielda b:fieldb c:fieldc)}
and field selection
{Browse C.a}

In particular, procedures like Label and Arity are not defined on Chunks. If Chunk were not a data type that's given to you, give an implementation of chunks using the secure unbundled ADT based implementations, using procedures and names.

5. Implementing classes from first principles: Consider a class Counter, defined as follows:

```
class Counter
```

```
    attr val
    meth inc(V)
        val:=V
    end
    meth init
        val:=0
    end
end
```

Explain how you can implement this class using chunks. A class is a stateless value. A class in Oz is just a chunk that has

1. a collection of methods in a method table
2. a description of attributes that each ***instance*** of the class will possess.
3. a description of features that each instance of the class will possess.