CS 430 Virtual Tables

Consider the following Java-like code snippet (all non-primitive variables are reference types).

```
abstract class Animal {
    boolean sleeping = false;
    void sleepToggle() { sleeping = !sleeping; }
    abstract void speak();
}

class Dog extends Animal {
    void speak() { println("Bark! Bark!");
}

class Cat extends Animal {
    int lol_power = 100000;
    void speak() { println("I CAN HAZ CHEEZBURGER"); }
}

class Evil extends Cat {
    void pillage() { println("I DESTROYZ UR THATCHEDROOF COTTAGES!!1!");
    void sleepToggle() { sleeping = false; }
}

void poke(Animal animal) {
    animal.speak(); // A
}

void main() {
    Animal bonnie = new Dog();
    Animal garfield = new Cat();
    Cat trogdor = new Evil();
    poke(trogdor);
}
```

1. Explain how the compiler implements the call to `animal.speak()` at comment A and how this allows for polymorphic behavior.

2. Draw a diagram of the layout of the stack, heap, and code sections of the program when it reaches comment A. You should include a class invocation record for each instance and a virtual table for each class placed in the appropriate section of memory.