Cat and Dog are concrete subclasses of the abstract Pet class. Instances are given a (String) name by the constructor. Pet has public String getName(). Pets are either happy or unhappy. Methods public setHappy(boolean b) and public boolean isHappy() are provided. Pets are initially happy. For example the following prints “Carl is unhappy”:

    Pet p = new Cat("Carl");
    p.setHappy(false);
    System.err.print(p.getName());
    if (p.isHappy())
      System.err.println(" is one happy pet");
    else
      System.err.println(" is unhappy");

There is an abstract class of Visitor to different subclasses of Pet. Pet has method abstract public void accept(Visitor v). The subclass SoundV uses System.err.println to add “sound” (later it could play .mp3 files). SoundV prints “prrr” when it visits a happy Cat versus “hsss” if it is unhappy, and when visiting a Dog, either “woof” if it is happy, otherwise “grrr”.

1) [10] Given that Pet p has been assigned some Cat or Dog instance, provide one line of Java code that will cause either “prrr”, “woof”, “hsss” or “grrr” to be printed. Do not use “instanceof” or “if”.

        SoundV s = new SoundV();

2) [50] Write Pet, Cat, Visitor, and SoundV (but not Dog). Use “abstract” where necessary. Do not use “instanceof”. Use “if” only in SoundV.

3) [15] Write Turtle as a new subclass of Pet. Turtles are always happy. Without modifying any code in Pet, make certain setHappy for Turtles will have no effect. That is, the following prints “Churchy is one happy pet”:

        Pet p = new Turtle("Churchy");
        p.setHappy(false);
        if (p.isHappy())
          System.err.println(p.getName() + " is one happy pet");

4) [25] Next, all Pets can visit with each other. A Cat visiting another Cat causes them both to become happy. When a Cat visits a Dog (or vice versa), the Cat becomes unhappy while the Dog becomes happy. When a Cat visits a Turtle (or vice versa) the Cat becomes happy. Describe all changes necessary to add this capability. You can either describe in English or directly mod-
ify your code. You are encouraged to clarify your design first using UML class diagrams. Do not write the Dog class (it would be straightforward).