Every Wasp, Bird, Plane, and Tomato is a kind of Flyer. Every Flyer has public void methods fly() and land(). Wasp and Bird are kinds of Animal. Animal and Tomato are kinds of Organism. Every Organism has a public boolean method isAlive(). Organism and Plane are kinds of Thing. Every Thing has a method float getWeight(). The following show correct and incorrect code:

```java
Bird b = new Bird();
Thing t = new Wasp();
Flyer f = new Tomato();
System.err.println(t.getWeight());
f.fly();
```

```java
Thing t2 = new Thing(); // does not compile;
Flyer f2 = new Flyer(); // does not compile;
Organism o2 = new Organism(); // does not compile;
Animal a2 = new Animal(); // does not compile;
```

1) [15%] Using UML and a written explanation that refers to your UML as necessary, design classes and interfaces consistent with the above. Include Wasp, Bird, Plane, Tomato, Flyer, Animal, Organism. Use UML symbology for inheritance, interfaces, abstract, and include the methods isAlive(), getWeight(), fly() and land(), with visibility modifiers and return types.
2) [5%] Consider:

```java
Organism o = new Wasp();
if (o.isAlive()) {
    Thing t = o;
    System.out.println("living wasp's weight = " + t.getWeight());
}
```

Is there any Java error with any of the above statements? If so, what?

3) [5%] Presume `Thing t2` references some `Thing`. Provide a couple of lines of Java code to assign that object to a `Wasp w`, if (and only if) it is a `Wasp`.

4) [10%] How would you add `Penguin` to your design? Note that it is `Bird` and yet not a `Flyer`. Be specific. What if you need to add many other non-flying Birds to your hierarchy (Kiwi, Ostrich, Emu, etc.)? Describe two basic techniques to handle this problem.
5) [10%] Suppose interface I has method m1(), and class C1 implements I, and C2 extends C1 and has additional method m2(). Given
   C1 c1 = new C2();
   I i = c1;
   C2 c2 = (C2)i;

   a) [5%] Explain whether or not i.m2() is valid.

   b) [5%] Is there any differences between (C2)i and c2? Explain.

6) [5%] Could Organism and Thing both be interfaces? If not, explain why not. If so, provide simple code for each.
7) [40%] New capabilities will be given to every Animal using the Visitor Pattern. Presume we only have the two Animals Wasp and Bird. There would be many different kinds of Visitor. One kind of Visitor is SoundVisitor.

a) [5%] Add to your UML diagram the additional methods and classes needed.

b) [5%] Write the method that needs to be added to Wasp:

c) [5%] Complete the following code so that it results in sv visiting w:

```java
SoundVisitor sv = new SoundVisitor();
Wasp w = new Wasp();
```

d) [10%] Write Visitor.

e) [15%] Write SoundVisitor, which for simplicity just prints a specific string (“bzzz” when it visits a Wasp and “chirp” when it visits a Bird). Do not use instanceof nor any usages of conditionals (either explicit such as “if” or “while” or implicit such as boolean expressions, etc.).
8) [5%] What are some criteria for deciding on whether to make a given level in a hierarchy a class or an interface?

9) [5%] What are two reasons for making a class abstract?