Pigeon is a kind of Bird. The Boeing B777 is a class of Airplane. Bird and Airplane are abstract... but Pigeons and B777's can be constructed. They are not members of a common hierarchy.

Both Pigeon and B777 have a method to fly(); that is, they are Flyers. All Birds and Airplanes own instances of Wing. Think of Pigeon and B777 as constructing their Wings within their constructors. Both Bird and Airplane have a method getWing() which returns a Wing.

Both Pigeon and B777 also have the ability to take(Message m). In a Pigeon the message is tied to the leg, in a B777 it's taken on board. Another class, Email, also has a method take(Message m), but no fly() method.

Pigeon, B777, and Email are all different kinds of Carrier. Any class that is a Carrier has a Message (but does not construct it); the message is retrieved by getMessage().

Assume the following compiles and works correctly:

Flyer f;
Carrier c;
Email e = new Email();
B777 b = new B777();
Pigeon p = new Pigeon();
Message m = new Message("having wonderful time, wish you were here");

c = p;
c.take(m);
f = (Pigeon)c;
f.fly();

Some of the following works and some fails, revealing further design aspects.

((Bird)p).fly(); // fails to compile...Think penguin.
((Bird)p).take(m); // this is ok... penguins can take
                  // messages but not fly!
((Carrier)p).take(m); // this is ok too.
e.fly();              // this fails to compile, but you'd
            // expect that, right?
Airplane airplane = b;// works as you would expect.
airplane.take(m);    // works too... so all airplanes are
            // carriers
airplane.fly();      // also works... all airplanes, not
            // just B777 are flyers

1. (50) Draw a UML class diagram for Pigeon, Bird, B77, Air-
plane, Flyer, Wing, Carrier, Email, and Message consistent with
the above. Place the methods fly(), getWing() and take(Message
m) within this diagram, using UML annotation style for the re-
turn types and visibility.

2. (50) Write the class definitions in Java, but don't worry
about including any code within the methods, just have {}.