Import the four java files into a new project. Three of these classes form an inheritance hierarchy, and the fourth is a client. Before continuing, be sure you understand which classes are parents and/or children of which other classes, and which class is the client. You should start thinking about the methods that get inherited in this hierarchy.

Right away, you will notice that Labrador has an error in it. Don’t worry about it now, we will fix it later.

DogTest is the client. It creates a Dog object and makes it speak. Run it to see what it does, and make sure you understand the code.

Now modify the files as follows:

1. Let’s fix the Labrador problem. The constructor is complaining about not being able to find a zero-parameter constructor in the super-class (Dog). This is because a constructor in a child **always** calls its parent’s constructor. If you don’t do it explicitly on the first line of the constructor (using **super(params)**), java will automatically call the zero-parameter constructor in the parent for you. The problem here is that Dog doesn’t have a zero-parameter constructor. Change Labrador’s constructor to explicitly call Dog’s constructor with the appropriate parameters.
2. In DogTest, create and print a Yorkshire and a Labrador, much like it already does for Dog. Don't change any files besides DogTest. Run it and make sure it works. You should have three different types of dogs speaking.
3. Now add code to DogTest to print the average breed weight for both the Labrador and the Yorkshire. Call avgBreedWeight() on the objects to do it. You will discover that Yorkshire doesn’t have an avgBreedWeight() method. Go ahead and add one. It should be a small number since it’s a small dog. See Labrador for an how it does avgBreedWeight().
4. Since both subclasses (Yorkshire and Labrador) now have an avgBreedWeight() method, it makes more sense to declare that in the parent. That will essentially say that all subclasses of Dog need to have an avgBreedWeight(). So, in Dog, add an abstract *avgBreedWeight()* method that returns an int. We don’t want it to be concrete because we really don’t have an average weight number for a generic Dog. Make sure you use the word *abstract* in the method header after *public*, and that you don’t have a body (just a semicolon after the parameter list). If any other errors pop up in Dog, fix them (there will definitely be an error you will need to fix). Now all subclasses of Dog are required to have an avgBreedWeight method. Since both Yorkshire and Labrador do, you should be all set.

As soon as you finish this last step, an error will pop up in DogTest. If you did everthing correctly in Dog, Dog is an abstract class now. Therefore DogTest is not allowed to construct a Dog object. Fix DogTest: you will not be able to construct a generic Dog anymore; take it out. This is fairly common to have an abstract parent that can’t be instantiated, and to only have concrete children that can be instantiated.