# JAVA PROGRAMMING PRACTICE – DOG KENNELS EXERCISE

This exercise is designed to give you experience in creating a simple Java program using an OOP approach.

If you can complete this task and understand what you are doing, then you will find that the slightly more complex tasks from Week 9 onwards will seem a lot easier.

You might need to go back and examine the structure of a couple of programs that you have already looked at – **MiniCalc** (Week 2), **Car** (Week 5) and **Rectangle** (Week 6). The last one is probably the most useful to examine at this point.

If you are asked to do something here that you don’t understand, then this is your chance to go and look it up. Work with your colleagues in completing this exercise.

1. Using BlueJ create a new Project named **Dog\_Kennels**.
2. Create a new class called **Dog**.
3. You should remove all the skeleton code except

**public class Dog**

**{**

**public Dog**

**{**

**}**

**}**

1. Declare the following three variables/attributes for objects of the class

**dogName**, **age**, **breed**

Use appropriate datatypes, and start each declaration with the keyword **private**

ie. **private String dogName**;

1. Create a Constructor method which expects three arguments/parameters to be passed to it – the dog’s name, age and breed.
2. The Constructor should have code that sets the values for the attributes which you declared above.

Remember that the Constructor code is run only once – when a new object is created.

1. Create 6 new methods – two for each of the attributes declared above. One will change the value in a variable when the program is running and one will get the value – otherwise, called **setters** and **getters**,

You may need help with this but see the example class **Rectangle**.

It is important to get the heading for each method correct. Do you need parameters passed to the method? You will need to use the **return** keyword in the methods that get the value in a variable.

1. Create a **main** method in the **Dog** class

In the **main** method:

* 1. Create a new object of the **Dog** class. Make sure you give the dog a name, age and breed.
  2. Using your ‘getter’ methods get the new object’s name, age and breed. You will need to create some new local variables to hold these values and which have scope ONLY in the **main** method.
  3. Use these variables to display the dog’s details in the terminal.

For example, “Your dog is called Fido. Fido is 6 years old and he is a Cocker Spaniel”

* 1. Use the setter method to change the dog’s name and display the revised information.
  2. You are going to create some new dogs. Before you do this. you should create a new variable in the **Dog** class to keep a running total of the number of dogs. This will be a class variable NOT an object variable and its value will be accessible to all objects of the **Dog** class

Use something like:

**public static int numOfDogs = 0;**

Add this to your Constructor:

**numOfDogs ++;**

* 1. Now add some code to your **main** method to access the **numOfDogs** variable and display its value in the terminal.
  2. Create two more **Dog** objects and display all three dog details in the terminal.
  3. You should now adapt your code in the **main** method to ask the user to input the dog details using the **Scanner** class.

The user should input the details al the three dogs and all three dog details should be displayed in the terminal together with the number of dogs.

1. The next stage is to create a new class and name it **DogDemo**. In BlueJ you should now see two ‘files’ – **Dog** and **DogDemo**.
2. Remove all the code in the **DogDemo** class except for:

**public class DogDemo**

**{**

**}**

1. Cut the **main** method from the **Dog** class and paste it into the **DogDemo** class.
2. Compile and run the **DogDemo** class. Your output in the terminal window should be exactly the same as before.
3. You have now successfully created an **orchestrating** class for the **Dog** class

**EXTRA**

The dog kennels business has only 10 kennels. A single kennel provides sufficient space to house either one large or medium-sized dog or two small dogs.

You are needing to enter dog details until the kennels are full. When the kennels are full an appropriate message should be displayed in the terminal window.

1. Add another attribute/variable to the **Dog** class which stores the size of the dog – large, medium or small.
2. Adapt the Constructor method to accept another parameter – the dog size.
3. Create a new orchestrating class allowing the user to input and display details of dogs until the kennels are full. Remember that a single kennel can house two small dogs.
4. Display each dog’s details, the total number of dogs housed and a message saying the kennels are now full.