

Introductory Java Programming

School of Electronic Engineering
and Computer Science

Course Code: EBU4201

Lab Sheet 3: OO Programming – Objects and Classes

1. Download the files **Dog.java** and **DogTest.java** from the course website.
 - i) Compile and run the two Java files above.
 - ii) Add access modifiers to the **Dog** class, conforming to the principles of information hiding (see lecture notes)¹. Change the code in the **DogTest** class to assign the **name** and **speed** values for the dog object, and print out the **name** and **speed** of the dog object.
 - iii) Write a constructor for the **Dog** class. This constructor should initialise all instance variables to the values passed in as parameters to the constructor. Rewrite the code in the file **DogTest.java** to use the new constructor.
 - iv) Create the two **Dog** objects described below, using the constructor you wrote in *part iii*) and print out:
 - the **name** and **colour** of **dog1** and call **dog1** to run in a straight line for 10 minutes;
 - the **name** and **colour** of **dog2** and call **dog2** to run in a zigzag for 5 minutes.

dog1

```
name = "Buddy"
tail = true
speed = 500
furType = "short"
colour = Color.BLACK
```

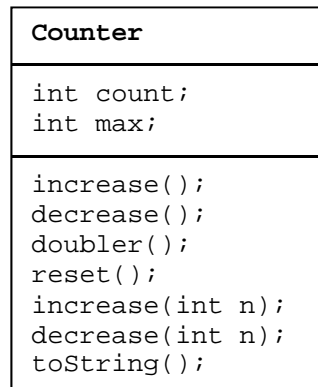
dog2

```
name = "Beethoven"
tail = false
speed = 400
furType = "long"
colour = Color.GRAY
```

2. Create a class **Rectangle** (and store it in a file called **Rectangle.java**), such that:
 - i) It should have two instance variables **l** (the rectangle's length) and **w** (the rectangle's width).
 - ii) It should have one method that calculates the area of the rectangle and returns it to the caller.
 - iii) Create a **main()** method *in the same class* for testing purposes. In the **main()** method, create one rectangle with the dimensions **(l,w)=(8,6)**, and another rectangle with the dimensions **(l,w)=(7,7)**. Call your area method on each rectangle and print the corresponding result.

¹ You will have to declare instance variables as **private** and provide *getters* and *setters* in the **Dog** class.

3. Write a class **Counter** that represents a simple counter (as defined in the UML class diagram below) and write a class **CounterTest** to test it. Following the steps below may help you.



- i) Write the **Counter** class which should have two **private** instance variables: **count** and **max**. Add a default constructor to the **Counter** class, such that:
 - it has no parameters;
 - it assigns value **0** to **count** and value **10** to **max** as their default values.
- ii) Add *getters* and *setters* for the two private instance variables. The *getter* methods should be named **getCount()** and **getMax()**, whereas the *setter* methods should be named **setCount(int n)** and **setMax(int n)**.
- iii) Write the **CounterTest** class with a **main()** method. In this class, create a **Counter** object and call the *getter* and *setter* methods to test them. Compile and run your program.
- iv) Add four methods to the **Counter** class:
 - **increase()** method to increase the **count** value by **2**;
 - **decrease()** method to decrease the **count** value by **1**;
 - **doubler()** method to double the **count** value;
 - **reset()** method to reset the **count** value to **0** and print the message "**Counter Reset!**".

Test these four methods by invoking them in the **CounterTest** class.

- v) Add a **toString()** method in the **Counter** class and test this method in the **CounterTest** class. This could simply display the current values of **count** and **max**.
- vi) Modify the **increase()** method so that it resets the counter when the **count** value reaches the **max** value. Modify the **decrease()** method so that it does NOT decrease the **count** value when **count** reaches **0**. Test both of these methods in the **CounterTest** class².
- vii) *Method overloading*: Add another two methods, **increase(int n)** and **decrease(int n)**. These methods increment and decrement the **count** value by an amount **n**, respectively. Test both of these methods in the **CounterTest** class.
- viii) Check that you have written sufficient comments in your code; you need to write:
 - *Javadoc* comments (notation: **/** ... */**) for the class and for each method;
 - some inline comments (notation: **//** or **/* ... */**) to explain your code if necessary.

Ensure all your programs contain both internal comments and *Javadoc* comments.

² **Hint:** You may want to use a loop in the **CounterTest** class to invoke the **increase()** and **decrease()** methods many times.