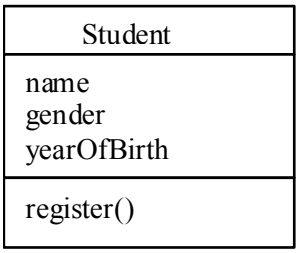
ECS705/ECS717

Lab Sheet 4: OO basics

Essential exercises:

Exercise 22.

a) Create a Student class according to the UML diagram below. The register method simply prints "You have been registered". Name your program *Student.java*.



b) Write a test class for your Student class. Your test class should create two student objects. One called John Smith, who is male and his year of birth is 1985. Another one is you (your own name, gender and year of birth). After creating the two student objects, print off the name and call the register method of both of them. Name your program *StudentTest.java*.

Exercise 23.

Create a class *Square.java*.

i. It should have one instance variable x: the length of the side.

ii. It should have one method that calculates the area of the square and returns it to the caller.

iii. Create a main method *in the same class* for testing purpose. In the main method create one square with x = 5, and another square with x = 29. Call your area method on each square and print the result off.

Exercise 24.

Download `Circle.class` and `CircleTest.java` from the course website. Put these two files in the same directory. Compile and run `CircleTest`.
Circle has two methods for manipulating its private instance variables. They are:

setRadius

`public void` **`setRadius`**`(int rad)`

Sets the radius of the circle. Valid values of radius are 1-4.
Parameters:
`rad` - The radius of the circle (1, 2, 3 or 4).

setColour

`public void` **`setColour`**`(java.lang.String col)`

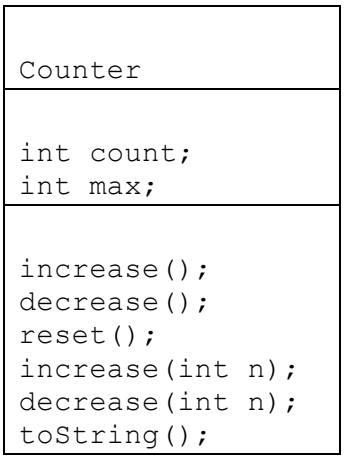
Sets the colour of the circle. Allowable colors are red, blue, orange and yellow.

Parameters:
`col` - The colour of the circle

Create 4 new circles (use different colours and radiuses). The line `Circle.displayMyCircles();` must be the last line of the main method! These circles are instances of the class Circle. They are 4 unique objects. Notice if you change the values of radius or colour for a particular instance of a circle only that circle changes!

Exercise 25.

Write a class `Counter` that represents a simple counter as defined in the following UML class diagram. Write a class `CounterTest` to test it. Follow the steps below may help you:



i. Write the `Counter` class. The `Counter` class should have two `private` instant variables: `count` and `max`. (`count` is the current count, `max` is the maximum count allowed). Add a default constructor for the `Counter` class. This constructor should have no parameters; it should assign value 0 to `count` and value 10 to `max` as default values.

ii. Add accessors(getters) and mutators(setters) for the two private instant variables. The get methods should be named `getCount()` and `getMax()`. The set methods should be named `setCount(int n)` and `setMax(int n)`.

iii. Write the `CounterTest` class with a `main` method. Create a `Counter` object, call the getter and setter methods to test them. Compile and run your program.

iv. Add three methods in `Counter` class: `increase()` method to increase the count by 1; `decrease()` method to decrease the count by 1; `reset()` method to reset the count to 0 and print a message "Counter Reset!". Test these methods by invoking them from `CounterTest` class.

v. Add a `toString()` method in `Counter` class. Test this method in `CounterTest` class.

vi. Modify the `increase()` method so that it should reset the counter when the `count` reaches the max value. Modify the `decrease()` method so that it should NOT decrease the `count` when the count reaches 0. Test these methods in `CounterTest` class. [hint: you may use a loop in the `CounterTest` class to invoke `increase()` and `decrease()` methods many times]

vii. Method overloading: Add another two methods `increase(int n)` and `decrease(int n)`. These methods give the `count` n increments/decrements. Test these methods in `CounterTest` class.

viii. Finally, check if you have written sufficient comments for your code. You need to write JavaDoc comments /**….*/ for the class and for each method. Write some inline comments // or /*…*/ to explain your code if necessary. Generate JavaDoc of the `Counter` class.

Desirable exercises:

Exercise 26.

Download all starter files for this exercise from the course website.
Colours in graphical applications are often defined in terms of three separate red, green and blue colour component values. Normal values for these components typically lie in the range **0 … 255**. The code provided in `ColorDisplayMain.java` will generate a window whose main background is coloured with a particular set of RGB values that are displayed within it. The window is implemented by a `ColorDisplay` object whose API contains the following method:

```
public void changeColor (int red, int green, int blue)
```

Sending a `changeColor` message to a `ColorDisplay` object will cause it to use the given values to change its background colour.

a) Add a ***for*** loop to the main method in `ColorDisplayMain.java` so that the value of the red component is successively changed from 0 to 255 over its full range. The `ColorDisplay` class pauses after each colour change to make it easier to discern the changes. The length of this delay may be changed by sending it a `setDelay` message whose argument is the number of milliseconds to pause between each change.
b) Add further nested ***for*** loops to your answer so that every combination of red, green and blue value in the range **0 .. 255** is used.
c) Modify the ***for*** loops in your previous solution so that different step sizes are used for the red, green and blue components. For instance, the red component might be changed by 10 units on each iteration, while the green component might be changed by 15.

Optional exercises:

Exercise 27.

Character values may be compared using the relational operators in the same way as integers. Write a class named `CharRange` that contains two methods: `isCapital` and `isRoman`. `isCapital` method returns true if a given character value is an uppercase character, in the range ‘A’ to ‘Z’. `isRoman` returns true if the character value is one of the characters: ‘M’, ‘D’, ‘C’, ‘L’, ‘V’ or ‘I’. Write a main method in the class or write a test class to test it.