# COIT20256 Tutorial/Laboratory

# Week 1

Preparation **before** the tutorial class

This tutorial (and future tutorials) assumes that you have access to the following software:

* NetBeans 8.2
* Java 8(Oracle)
* JavaFX SceneBuilder 8.5.0 (required to start assessment item 1 and required in the week 4 tutorial)

You will also need this software to complete the assignment work.

See the resources block on the right-hand side of the unit website for links to documentation explaining how to download and install the software on your own computer. Note that this resources section also has useful documentation about NetBeans, Guidelines for Java Programming, and a link to a video describing how to use the debugger in NetBeans. Violet is also a useful tool to download. You should also become familiar with all these tools and documentation.)

Note that the software is also available in the labs on campus.

**Make sure that you use the versions of the software specified above** or your marker may not be able to mark your assignment work.

**To check that you have all the software installed and working on your computer** complete question 1 (the simple “hello world” exercise) at the start of the week 4 tutorial questions.

## Questions

**Question 1**

a) In NetBeans, implement the UML diagram created in Activity One of the lecture as described below. A flower has:

• name – string

• category - String

• price – float

b) Include a toString() method with correct access specifier to display the details of the flower.

c) Create a driver class with main() method and **test your flower class**.

d) Implement the UML diagram created in Activity Two. Modify the part (a) implementation to include:

• A no-argument constructor

• A parameterised constructor taking all three arguments

• A parameterised constructor taking the name and category

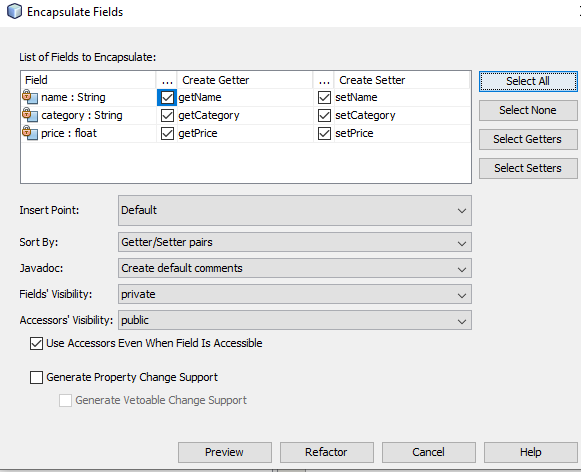
• A parameterised constructor taking only the name

• accessor and mutator methods for the instance variables.

Notes:

1. Although we are including getter and setter methods (accessor and mutator methods) for our Flower class, **in practice these should be used sparingly as they are exposing the underlying implementation of the data in the class.** This violates the concept of information hiding. It is therefore better to avoid the use of getters and setters as much as possible and only use them when they are needed. Getters and setters are useful if they are needed to perform additional functions such as enforcing constraints on values or updating other related values that need to be modified when the instance variable is changed.
2. Remember that you can use NetBeans to ensure indentation is consistent. To do this, select the section of code to be indented and hit <alt><shift>F
3. You can also use NetBeans to help create your getter and setter methods easily by right clicking on the class, select refactor->encapsulate fields and then selecting the getters and setters you want to code.

For example, after right clicking on your Flower class and selecting refactor->encapsulate fields you will see:



You can “select all” as shown above and then click on refactor to generate basic getters and setters.

In this exercise you are also required to include validation checks in the set methods (i.e. a flower name should not be null and should only contain alphabetical characters). (Hint: use regular expressions and see the example for more hints in activity 2 in the lecture slides.)

e) Add to the flower, a flower variable called count to store the number of flowers created. (Hint: static). Implement a static method getCount to return the number of flowers (value in count). Test your implementation of Flower by creating four flowers. Each flower is to be created using a different constructor. Display the information about each flower.

f) Use the getCount() method to find out the number of flowers (ie. Value in count) and print out a message saying the number of flowers on three occastions:

* Before any flowers have been created (use Flower.getCount()
* After you have created two flowers (use an instance of a flower to access the getCount method)
* After all the flowers have been created.

Your output should be similar to the following:

The number of flowers is now 0

< daisy filler $2.00>

< unknown unknown $0.00>

The number of flowers is now 2

< rose focal $0.00>

< snap dragon unknown $0.00>

The number of flowers is now 4

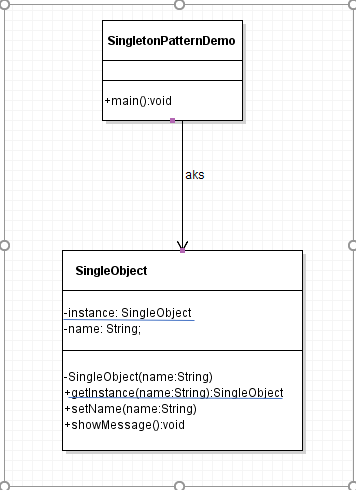
**Your tutor will discuss the answers to question 1 in class.**

**Question 2**

It is often necessary to ensure that only one instance of a particular class is created in an application (e.g. a database connection). This can be enforced using the Singleton pattern which is one of the simplest design patterns. A design pattern is a repeatable, reusable solution to a software development problem. You will learn more about design patterns in your next software development unit (Software Engineering).

The following diagram is based on the diagram from

<http://www.tutorialspoint.com/design_pattern/singleton_pattern.htm>



1. **You are to implement, run and test the program illustrated in the above diagram**. Note that the SingleObject getInstance method is static and the constructor is private. The instance variable called instance should be initialized to null.

The SingleObject getInstance method should test to see if instance is null. If it is, it should call the private constructor to create a new instance of SingleObject, assign it to instance and return instance. If it was not null, then it should return the already existing instance.

The showMessage method should print out the value stored in name followed by “says Hello World!”

For example if the main method contains the following code:

public static void main(String[] args) {

//Get the only object available

SingleObject object1 = SingleObject.getInstance("Mairi");

//show the message

object1.showMessage();

}

It should output the following:

Mairi says Hello World!

1. Why is the getInstance method static?
2. Why is the class variable SingleObject instance static (i.e. why is it a class variable)?
3. Why is the constructor private?

To read more (and for a partial solution to a slightly different version of SingleObject) see:

<http://www.tutorialspoint.com/design_pattern/singleton_pattern.htm>

**Questions 3 - 5**

Solutions will be provided for the following practical exercises, questions 3-5, on the unit website at the end of week 1.

**Question 3**

Write a Java application that asks a user to input the radius of a circle and uses a static method called circleArea to calculate the area of the circle. You are to enter the data to test your program via the standard input (i.e. the keyboard).

**Question 4**

Design and implement a class that has two overloaded static methods for calculating the areas of two geometric shapes (i) Circles; and (ii) Rectangles. When these two methods are to be overloaded, they should each have the same method name but different parameter sets.

Write a demonstration program to show how the overloaded static methods work.

**Question 5**

Write a Java application that checks whether a password input is valid or not. The criteria for a valid password are as follows.

(i) The password must have at least one digit.

(ii) The password must contain at least one lowercase letter and at least one uppercase letter.

(iii) The password length must contain at least eight characters.

You can implement this program with a class named PasswordChecker, which has a data member - password (a String) and a method to verify the validity of the password. (Note: Refer to Regular expressions from Introduction to Programing)