OBJECT ORIENTED PROGRAMMING USING JAVA

Workshop Instructions

2 – Creating Simple Java Classes

CREATING SIMPLE JAVA CLASSES

Objectives

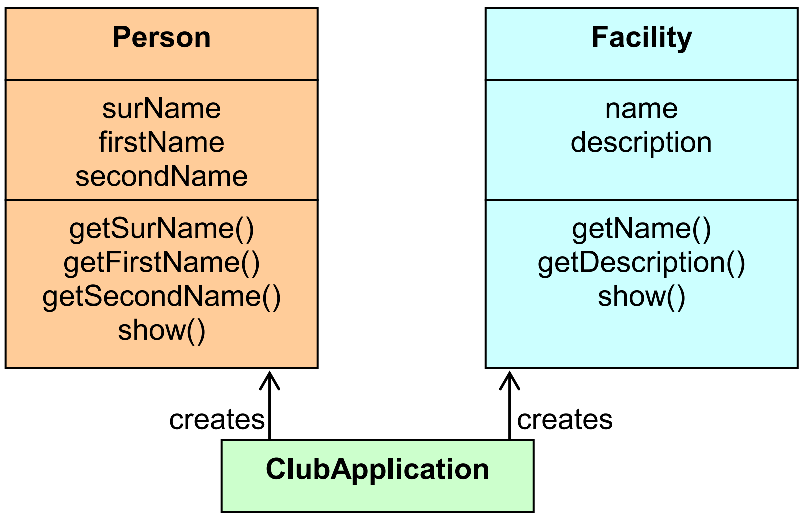
The objective of this workshop is to create some basic Java classes, add attribute and methods, and use these classes by instantiating and utilising objects.

The Application: Club Manager

This workshop, and the ones to follow, will allow you to practice what you learnt in class by building up a small system written in Java. This is a simple application that keeps records of members in a club. You will start building this application from scratch. Each assignment will cover different aspects of the Java language and environment.

At the end of each workshop, you should have completed as much of the assignment as possible. If the assignment comprises a considerable amount of work, you should not worry if you cannot finish it all- though you will benefit from doing so. Before you start on the following workshop, you will be provided with the solution from the previous one, so that you can start from a common base of finished code. This will ensure that everybody starts from the same level, regardless of whether they have completed all parts of the previous assignment.

Exercise



Setting up the application skeleton

1) Create a new Java Project named **`ClubManager`** in your Eclipse workspace.

2) Create a class named **`ClubApplication`**. Initially, this class will only contain a **`main()`** routine which prints a simple message to the console.

3) Run the program, and verify it works.

Creating the Person Class

4) Your next task will be to create a class **`Person`**. This will be the base on which to build the club members class.

5) Give the **`Person`** class three attributes: surname, first name and second name. Make these attributes private.

6) Add a constructor that will accept initialisation values for these three attributes.

7) In the **`ClubApplication`** class's **`main()`** method, declare and instantiate a **`Person`** object. Use arbitrary test values for the names. Make sure the system still compiles and runs. Verify that **`Person.class`** is present in your directory.

Using the Person instance

8) Add public accessor methods **`getSurName()`**, **`getFirstName()`** and **`getSecondName()`** so that the attributes of **`Person`** can be read.
Tips: Eclipse's Source > Generate Getters and Setters… is convenient.

9) Add a **`show()`** method to class **`Person`** , to check you initialised the object correctly. The **`show()`** method will print the person's full name out to the terminal.

10) Make the second name attribute optional for class **`Person`** . In other words, allow the **`Person`** class to be instantiated when a null value is passed for the second name. This would allow both "Tan Ah Beng" and "Stan Laurel" to be valid names. Make sure the **`show()`** method handles this correctly.

11) In the **`ClubApplication`** class's **`main()`** method, instantiate a few different **`Person`** objects, using different types of names. For each instance, use the **`show()`** method to display the name to the console.

12) Verify the system compiles and runs correctly.

Creating the Facility class

13) Our club is also going to have facilities (conference rooms, function rooms etc.) that can be booked.
Create a **`Facility`** class, which contains two attributes, both **`String`** objects: the name of the facility, and a short description. The description is not mandatory (it can be null), so you should provide the class with two constructors: one that accepts initialisation values for both attributes, and one that accepts only the facility's name.

14) Add public accessor methods **`getName()`**, and **`getDescription()`** so that the attributes of **`Facility`** can be read.
Tips: Eclipse's Source > Generate Getters and Setters… is convenient.

15) Provide the **`Facility`** class with a **`show()`** method that will display on the same line the facility name and, if present, the description of the facility between parenthesis.

16) In the **`ClubApplication`** class's **`main()`** method, instantiate a few different **`Facility`** objects, using both constructors. For each instance, use the **`show()`** method to display to the console.