OBJECT ORIENTED PROGRAMMING USING JAVA

Workshop Instructions

3 – Inheritance and Arrays

USING INHERITANCE

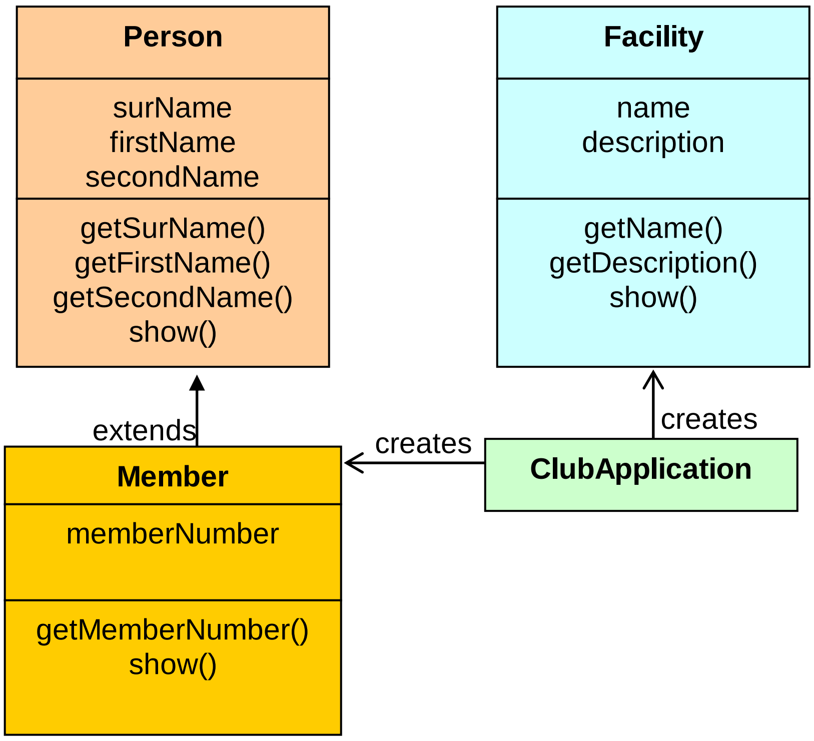
Objectives

The objective of this workshop is to practice inheritance, overriding and the use of arrays.

Exercise

1) Open the Java Project **`ClubManager`** that you created in your Eclipse workspace during previous Java workshop. Use the files as a starting point for this exercise.

Create and use the Member class



2) Derive a new class **`Member`** from class `Person` by means of inheritance. This class will add a *membership number* attribute (an integer). In other words, a member is a person with a membership number.

3) Write a constructor for this class. This method should accept an initialisation value for the membership number as well as those for the name. Make sure this constructor uses the **`Person`** constructor appropriately.

4) Include in class **`Member`**, a **`getMemberNumber()`** method which returns the membership number.
Tips: Eclipse's Source > Generate Getters and Setters… is convenient.

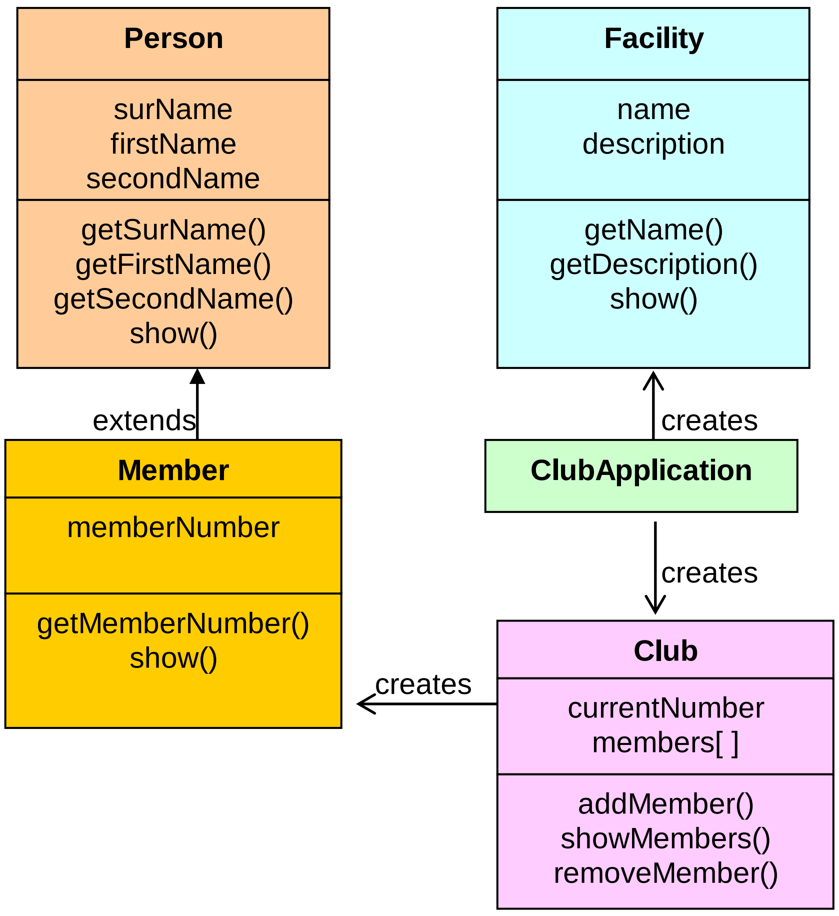
5) All classes in Java inherit the method `toString()` from the Object class. Override the `toString()` method in the `Person` class, so that it will return the full name of the person. Modify the `show()` method of the `Person` class, so that simply uses `toString()`.

6) In the `Member` class, override the `toString()` method so that, besides the full name, it attaches the membership number. The method should invoke the `toString()` method of the `Person` class.

7) Test the `Member` class: modify the `ClubApplication` class so that the test objects you create are of type `Member` and not `Person`. Modify their instantiation calls appropriately. Test these objects just as you did before, by calling their `show()` method.

8) For consistency, provide the `toString()` method in the `Facility` class too, modifying the `show()` method as appropriate.

Build a container for the members



9) Create a new class `Club`. This class's task is to keep track of membership records; first of all, it will assign unique incremental membership numbers.
Provide the class `Club` with a private attribute that contains the variable used to

assign membership numbers.
Add to class **`Club`** an array of **`Member`** objects. Make this attribute private.

10) Add a method called **`addMember()`**.
This feature will accept a person's details (surname, first and second name) and create a **`Member`** instance. It will automatically determine and assign a membership number. Members are given membership numbers in the sequence 1, 2, 3, 4...
Place it in the array, at the location corresponding to the membership number. This method should then return the newly created **`Member`** instance.

11) Use and test the **`Club`** class. Modify the **`ClubApplication`** class so that it creates an instance of the **`Club`** class, and instead of creating members directly, do so by using the **`addMember()`** method. Use the **`show()`** method of **`Member`** to show that the right membership numbers were assigned.

12) Add a **`showMembers()`** method to class **`Club`**. It should display a list with each member of the club on a separate line. Make this routine very simple, by using the **`Member`** class methods.

13) Modify class **`ClubApplication`** so it exercises the modified **`Club`** class.
Create the **`Club`** instance, and add a few members to it by using **`addMember()`**.
Use **`showMembers()`** to verify things are working.

Remove Members from the Club class

14) Add a **`removeMember()`** method to class **`Club`**.
Write **`removeMember()`** so that a member can be removed by passing its membership number as a parameter.
Make sure all other methods (such as **`showMembers()`**) can cope when some members have been removed from the array.

15) Test the **`removeMember()`** method by changing **`ClubApplication`** so that it creates a few members, and deletes one of them; after deleting that member, use **`showMembers()`** to verify that it did work.