

The University of Melbourne

School of Computing and Information Systems

SWEN20003
Object Oriented Software Development
Semester 2
Mid-semester Test

Length: This paper has 8 pages including this cover page.

Authorised materials: None

Time: 40 minutes, with 10 minutes reading time

Instructions to students: This exam is worth a total of 40 marks and counts for 10% of your final grade. Please answer all questions in the provided spaces on the test page; you may use the additional space provided for rough work. Please write your student ID in the space below. The test may not be removed from the test venue.

Advice: You do not need to write comments, but your answers must be **legible**; if we can't read it, we can't mark it. All worded answers must be written in English, and all code questions answered in Java. Make sure you read the **entire** test before starting.

Student ID:

Examiner's use only:

<i>Q1</i>	<i>Q2</i>	<i>Q3</i>

Define

Q1. Give **brief** answers for the following questions. [10 MARKS]

(a) What does an *association* between two objects mean in a UML diagram? [2 MARK]

(b) Briefly describe **two** advantages of Object Oriented Programming over “function oriented” programming. [2 MARK]

(c) Explain why we use the `equals` method when equating objects, and not `==`. [2 MARK]

(d) Define an instance variable called *frequency*, with initial value [0, 0, 0, 0, 0]. [2 MARK]

(e) Write a line of code to check equality for *family1* and *family2*, two arrays of **Persons**. [2 MARK]

Develop

Q2. In this question we will step you through implementing *classes* and *methods* in Java for a new social media platform called MyFace. MyFace allows *users* to join different *groups*, where they may communicate with each other. A user may also be an *admin* of one or more groups.

You can assume that the classes or methods in earlier questions “exist” in later questions, even if you haven’t answered the question. You must follow proper Object Oriented Design principles, and Java programming conventions.

Write **all** class declarations. **Do not** write method signatures that are given to you. [30 MARKS]

- (a) Implement a **Member** class. A **Member** is defined by their name, and each of the **Groups** they have joined; a **Member** may join *at most* 10 groups.

Your class should include a constructor, *appropriate* getters and setters, and a **toString** method that returns the **Member**’s name. [7 MARKS]

- (b) Implement a **Group** class. A **Group** is defined by its name, and each of the **Members** that have joined; a **Group** may have *at most* 100 **Members**.

Your class should include a constructor, and a **toString** method that returns the **Group**'s name, followed by a "list" of names of all its **Members**.


You **do not** need to write getters and setters

[5 MARKS]

- (c) Implement an **Admin** class, including the class declaration. An **Admin** is defined by their name, and each of the **Groups** they have joined; an **Admin** may join *at most* 10 groups.

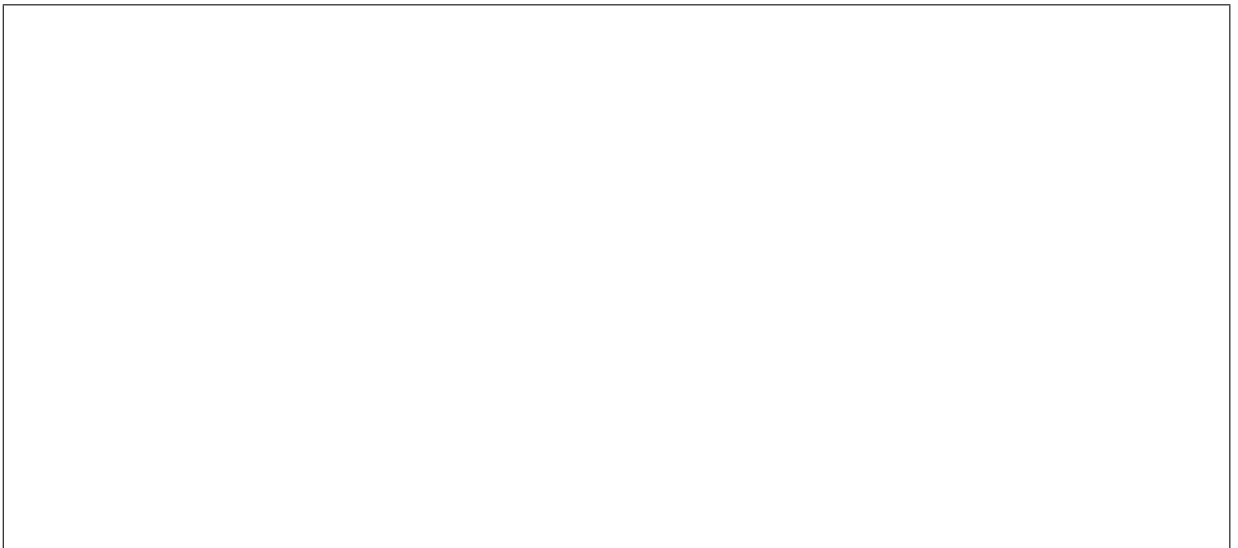
Your class should include a constructor, *appropriate* getters and setters, and a **toString** method that returns the **Admin**'s name, followed by **<admin>** (i.e. "Matt **<admin>**"). [4 MARKS]

- (d) Implement the `boolean isMember(Member member)` method for the `Group` class. This method should return `true` if `group` contains a `member` with the same name, and `false` otherwise. [4 MARKS]



- (e) Implement the `void addMember(Member member)` method for the `Group` class. This method should modify the `Group` to contain `member` as a member, and should modify `member` to indicate it is a member of the group. If `member` is already part of the group, this method should do nothing.

You may assume the `Member` class has the method `addGroup(Group group)`, which adds `group` to the groups the `Member` has joined. [3 MARKS]




- (f) Implement the `void removeMemberByName(String name)` method for the `Group` class. This method should modify the group by removing/deleting the `Member` with name `name`. You do **not** need to worry about removing the group from the `Member`. Make sure to think about what needs to happen to the rest of the data once a `Member` is removed. [4 MARKS]

- (g) Implement the `void removeMember(String groupName, String memberName)` method for the `Admin` class. This method should search the admin for groups with name `groupName`, and remove the `Member` with name `memberName`. If there is no group with the given name, this method should do nothing. [3 MARKS]

Q3. Bonus Question

Draw Matt a picture. Half marks for a reasonable attempt, full marks if it's pretty or makes us laugh. [2 MARKS]



Extra Space



