**PSP Assignment 2015S1**

**Problem domain**:

Converting and updating an application, i.e., PRACTICEHospital, from Jade to a JavaScript-based single-page application. The Jade schema named PRACTICEHospital has been provided.

**Ground rules:**

This PSP assignment has three sub-assignments: PSP1, PSP2 and PSP3.

Each sub-assignment has tasks for you to complete.

Each sub-assignment has its own marking schedule. Your submission **MUST** meet each **MUST** requirement in the marking schedule. Otherwise, **ZERO** mark will be given for the sub-assignment related.

**PSP1: Planning & design**

**Deadline: 8am Thursday, 19/March/2015**

Tasks:

1. Load the schema provided and analyze the Jade application by drawing a class diagram showing classes (including their attributes and methods) and relationships (including multiplicity).
2. Read through all tasks (in PSP2) below to update your class diagram.

Marking schedule (**5% of final marks**):

1. Planning and design (**MUST,** have at least 5 things, 3 marks)

Class diagram, code standard, counting standard (**MUST** have)

List of functions, use cases, time budget, risk management plan, etc., which can be further discussed with your tutor.

1. Test case description in plain English (3 marks)
2. Time log (1 mark)
3. Error log (1 mark)
4. Summary (1 mark)
5. PIPs (1 marks)

**PSP2: Implementation of test cases and generating a prototype**

**Deadline: 8am Thursday, 23/April/2015**

Tasks:

1. Implement your updated class diagram in JavaScript to generate the first version of the application.

Notice that Class Doctor and Class Patient may have some common attributes and methods. If they do, to improve your program, you **should** create a super class named Person for Class Doctor and Class Patient, and move their common attributes and methods into their super Class Person. Therefore, those common attributes and methods can be inherited from Class Person instead of individually being defined in both Class Doctor and Class Patient.

1. Write a method named loadTestData to initialize the application, which includes creating an instance of the Hospital class and the following 3 Doctors using the addDoctor method that exists in the Hospital class

ID Last Name First Name Office Fees

11 Xu Jian N400 $50

13 John Emily W7 $25

12 Kumar Santosh A22 $40

1. Write a method named displayDoctors that displays only the IDs and full names of all the Doctors.

***Note that the other data for them should not be displayed.***

The required output is:

11 - Xu Jian.

12 - Kumar Santosh.

13 - John Emily.

***NOTE: The order shown above is the required order. Punctuation and spacing must also be as shown above.***

1. Write a method named addPatient for the Doctor class that can be used to create a new Patient.
2. Edit the loadTestData method so that it creates the following 5 Patients, using the addPatient method created above.

ID Last Name First Name Fees Owing Doctor ID

200 Fisher Shelley $0 11

400 Koirala Kimi $25 12

300 Chen Liang $40 12

500 Ahmed Riyaz $0 13

100 Knightly Jude $30 11

1. Write a Boolean get method named getPaidStatus for the Patient class that returns true if the FeesOwing is $0 and false otherwise.
2. Write a method named displayBillablePatients that first lists the Doctor’s ID and full name, and then lists underneath the full names of each patient who has not paid the doctor’s fees in full. This method must call the getPaidStatus method created above.

The required output is:

11 - Xu Jian.

Knightly Jude

12 - Kumar Santosh.

Chen Liang

Koirala Kimi

13 - John Emily.

***NOTE: The order shown above is the required order. Punctuation and spacing must also be as shown above.***

1. Implement a new class named Therapy that has the following attributes:

Protected

name : String[30]

cost : Real

1. Each Patient can have zero or one Therapy. Many Patients can be receiving the same Therapy. Implement this relationship in the application.

Marking schedule (**10% of final marks**):

1. Updating your planning and design (**MUST,** have at least 5 things, **1 mark**)

Wire frame, class diagram, code standard, counting standard, MVC of your application (**MUST have**)

List of functions, use cases, time budget, risk management plan, etc., which can be further discussed with your tutor.

1. Update your test case description for testing the View proposed (**MUST, no mark**)
2. Use test case description created in PSP1 to create Jasmine-based tests and validate your code against those tests (2 marks)
3. Works and JSLint validation (1 mark)
4. Code (reuse, modular, parameters, prototype-based, etc.) (3 marks)
5. Time log & summary (1 mark)
6. Error log (1 mark)
7. PIPs (1 mark)