Name: ________________________________

Student ID:________________________________

Signature:________________________________

SEMESTER 1, 2018 EXAMINATIONS

COMP 1531

SOFTWARE ENGINEERING FUNDAMENTALS

(SAMPLE - PAPER)

1. TIME ALLOWED – 3 HOURS

2. READING TIME – 10 MINUTES

3. TOTAL NUMBER OF QUESTIONS – PART A (10), PART B (30), PART C (60)

4. ANSWER **ALL** QUESTIONS

5. TOTAL MARKS AVAILABLE – 100

6. MARKS AVAILABLE FOR EACH QUESTION ARE SHOWN IN THE EXAMINATION PAPER

7. NO MATERIALS MAY BE TAKEN INTO THE EXAMINATION ROOM

ALL ANSWERS MUST BE PROVIDED ONLINE EXCEPT WHERE THEY ARE EXPRESSLY REQUIRED TO BE DONE IN A SEPARATE SHEET OF PAPER. PENCILS MAY BE USED ONLY FOR DRAWING, SKETCHING OR GRAPHICAL WORK

PART A (Multiple Choice) – 10 marks

EACH QUESTION IS WORTH **1** MARK..

1. **Which of the following is not true in relation classes and objects?**
   a. Two object instances from the same class share the same properties, behaviour and object identity
   b. Objects have state but classes don't
   c. An object is said to be instantiated from a class as an instance of the class
   d. A class is a blue-print to logically group objects that share the same semantics, properties and behaviour from which an object is created
   e. An object is allocated memory when it is created, but a class is not allocated memory when it is created.

2. **Choose the correct response for the following statement.**
   ___________ is not a valid practice to ensure high quality in XP projects

   a. pair programming, where the roles of the partners change frequently
   b. simple design and refactoring of code
   c. open work space
   d. test-driven development
   f. long cycles of development

3. **Which of the following is true about software design?**

   a. Low coupling occurs when one module **A** depends on the internal workings of another module B and is affected by internal changes to module B
   b. Coupling is defined as the the degree to which all elements of a component or class work together as a functional unit
   c. High cohesive classes are easier to maintain
   d. Good software design aims for building a system with high coupling, low cohesiveness
   e. Low coupling leads to eventual software rot

4. **Choose the incorrect statement.**

   a. Assertions should be used for checking pre-conditions, post-conditions and program-invariants
   b. Assertions should be used for validating user-provided input
   c. Assertions can be turned off globally in production to optimise code
   d. The code snippet below is an incorrect use of asserts in Python

```
if not isinstance(x, int):
    raise AssertionError("not an int")
```

   e. Asserts are useful to check the correctness of your code

5. **Which of the following statements is incorrect about Single Responsibility Principle (SRP) ?**
    a. Business context is the driving force to coining SRP
    b. SRP implies just do “one” thing
    c. The objective of SRP is to achieve low coupling and high cohesion
    d. In the context of SRP, a responsibility is defined as a reason for change
    e. Every class should have only one responsibility

6.
7.
8.
9.
10.

Part B (Short Answer) – 30 marks

Note: The marks for each question varies.

Question 1 (6 marks)

In the context of software development life-cycle, different methodologies can be used to develop software applications. Identify and describe THREE differences between traditional Waterfall and Agile software development methodologies

Answer:

- A waterfall model follows a linear sequential model consisting of four phases namely, requirements analysis, design, implementation and testing where each phase must be completed prior to the start of the next phase, whereas an agile model flips the linear axis sideways, builds software in iterations where each iteration implements all the four phases on a set of features
- A waterfall model is rigid and not open to changes in requirements whereas an agile software model is open and adaptable to changing requirements
- Customer involvement in a waterfall model is typically at the start and end of the software life-cycle, while agile methods are characterised by continuous involvement throughout the life-cycle, prioritizing work-items and providing feedback on each iteration deliverable

Question 2 (6 marks)

(a) An application is required to perform a variety of tasks of varying complexity on a data stream in parallel. Suggest a suitable architecture for this application and why (3 marks)

Tutors: You do not need to cover this as this is being discussed in the lectures on Wednesday. Students with an early lab will not have heard this

Answer: Pipe and Filter Architectural Style

This is a suitable architecture, as the different processing tasks can be broken into separate components or filters, each performing a different task. The filters can be combined into a pipeline and also allow concurrent processing of data stream

(b) (3 marks)

Name the architectural style used in each of the following applications

a. Real-time updates on stock prices, weather updates, sporting results
b. Traditionally, the updates for any Windows device were delivered directly from Microsoft's *Windows Update* servers. While this is the most secure way of getting un-tampered files, it's not the fastest delivery method that you can use. Windows 10 computers and devices can connect to each other and get updates not only from Microsoft's dedicated servers, but also from other Windows 10 devices that have already downloaded parts of the updates.
c. A radio station where people tune into their favourite programs
d. Amazon web services

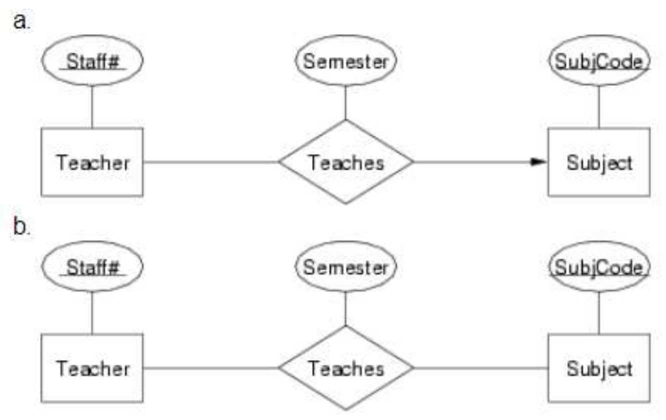
e. ls marks | grep student_id
f. Napster

Answer:

a. Publish-Subscribe – Subscribe to the topic you are interested
b. Peer to Peer architecture – A P2P network is an adaptation of a client-server architecture. Unlike a client-server architecture, which has dedicated client and network components, with defined roles, P2P relies on an ad-hoc network of components, where each peer component behaves both like a server and a client. P2P networks are the cheapest method of distributing content because they use the bandwidth of peers, not the bandwidth of the content's creator.
c. Publish-Subscribe: Broadcaster publishes their program and have no idea of who the subscribers are. Subscribers respond by tuning into their favourite program.
d. Service Oriented Architecture
e. Pipe & Filter
f. Peer to Peer Architecture

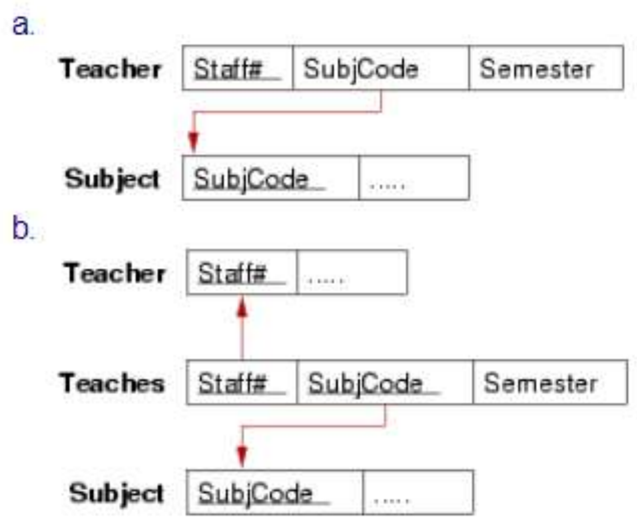
Question 3 (6 marks)

(a) Convert the following ER design fragments into a relational data model expressed as a box-and-arrow diagram:



Answer:

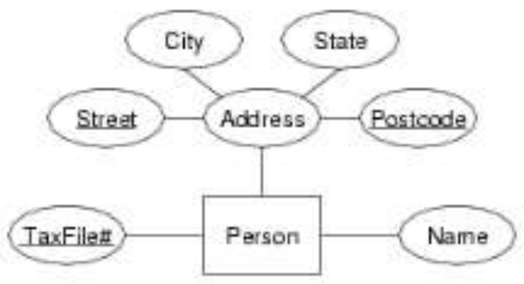
Relational models for the three Teacher-Teaches-Subject scenarios:



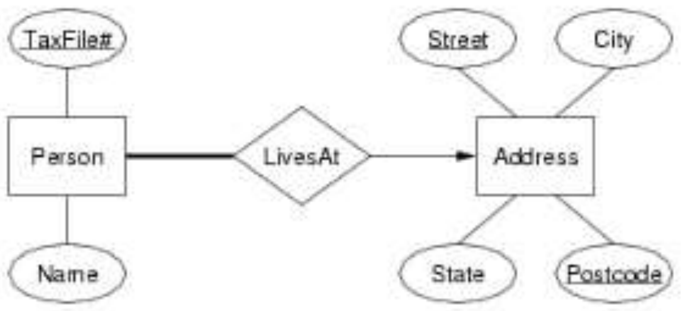
(b) The following two ER diagrams give alternate design choices for depicting Person and their addresses, one with Address as an attribute, the other with Address as an entity. Assume we have another entity type ElectricCompany. Only one of these companies supply power to each home address. Which ER representation is suitable and why?

Answer:

Address as attribute



Address as entity



Answer:

If the Address is an attribute then it is not possible to relate electric companies directly to a home address, instead they can only be related to persons, and through them indirectly related to addresses. This does not quite capture the information that we wish to model. We wish to relate electric companies directly to home addresses, so we would model the latter as an entity rather than as an attribute.

Question 4 (6 marks) –

Question 5 (6 marks) –

Part C – 60 marks

Note: The marks for each question varies.

Question 1 (15 marks)

Consider the following requirement for a field user-name: "the length of the user-name >=1 and <=25 and not contain a space".

(a) Design a function validate_user_name() that takes in as input a user-name and performs the above validation. If the validation fails, the function should throw a user-defined exception, "UserInputError". The skeleton code for this function is provided here. Complete the function and add the user-defined exception to be raised in the event of an invalid user input

```
#Add the user-defined exception

#business logic
def validate_user_name(name):
    # Add the validation logic
```

Answer:

```
#User-defined exceptions
class UserInputError(Exception):
        "Raised when the user-name field is invalid"
        pass

#business logic
def validate_user_name(name):

    if len(name) != 25 or name.find(' ') != -1:
                raise UserInputError("user input error")
    # Do something with the user-name, e.g., add to database
    return True
```

(b) Identify the equivalence classes of inputs for testing the above function and list the corresponding test-cases

Answer:

Field	Description
Valid equivalence class	
User Name	Length >=1 and <=25 not containing a space
InValid equivalence class	
User Name	Empty
User Name	Length > 25 (No spaces)
User name	Length >=1 and <= 25 and containing a space

Test-Case #	Input Value	Expected Output
1	“apple345”	True
2	“”	UserInputError
3	“longpassword”	UserInputError
4	“a pple34”	UserInputError

(c) Implement the test-cases above using PyTest.

```python
from validate import UserInputError,validate_user_name
import pytest

def test_validate_correct_input():
    input = "apple345"
    assert validate_user_name(input) == True

def test_validate_empty_user_name():
    input = ""
    with pytest.raises(UserInputError) as info:
        validate_user_name(input)

def test_validate_space():
    input = "appl 345"
    with pytest.raises(UserInputError) as info:
        validate_user_name(input)

def test_validate_incorrect_length():
    input = "somelongpassword"
    with pytest.raises(UserInputError) as info:
        validate_user_name(input)
```

Question 2 (15 marks)

a. Provide an example of a **static** UML diagram and why is this diagram said to be static?

Answer:

A class diagram is an example of a static UML diagram that provides a view of the entities in the “system-to-be” and the relationships between these entities. It is said to be static because it depicts the elements of the proposed system irrespective of time and does not model any interactions between these elements.

Tutors: Mention to students here that a sequence diagram is an example of a dynamic UML diagram as it models the sequence of interactions between the elements of the proposed system

b. In the context of a UML class diagram, describe the meaning of a **composition** relationship between two classes and give an example.

Read the following case-study and answer questions (c) and (d). A car washing company, Mega Car Wash, offers car washing services to customers. The company has requested you to design a job management system that stores information of the customers’ car washing jobs and notifies the user once the job for the user has been completed. The job management system should be able to cater for the following requirements:

1. The system must maintain a list of current jobs
2. For each job, the system should store an id, date and customer information.
3. Customer information should include the customer’s name and contact details.
4. After a job is completed:
   - the system must delete the job from its list of current jobs
   - the system must notify the customer through the contact details provided. A customer can choose to receive the notification through an Email or SMS.
5. The system can add a job or delete an existing job.
6. The system can obtain details of a job, including its id, date and customer information.

c. Draw a conceptual UML class diagram of the car wash system. **The conceptual class diagram needs to show class names, relationships, attributes and methods.**

Map the class diagram to **Python classes** with their corresponding attributes and methods. Implement the constructor for each class to be able to instantiate an appropriate object instance. For all the other methods, you do not need to provide an implementation; only the method signature needs to be defined. The implemented classes must conform to OCP design principles.

(a) Answer:
Conceptual Class Diagram:
See attached file – carwash_conceptual_model.pdf

(b) Answer: Python Classes

```
class Customer(object):

    def __init__(self, name, contact):
        self._name = name
        self._contact = contact

    @property
    def name(self):
        return self._name

    @property
    def contact(self):
        return self._contact
import time
from Customer import Customer

class Job(object):

    def __init__(self, job_id, customer_name, contact_detail):
        self._job_id = job_id
        self._customer = Customer(customer_name, contact_detail)
        self._date_of_job = time.time

    def __eq__(self, other):
        return self._job_id == other.job_id

    @property
    def job_id(self):
        return self._job_id

    @property
    def date_of_job(self):
        return self._date_of_job

    @property
    def customer(self):
        return self._customer

class JobList(object):

    def __init__(self):
        self.jobs = []

    def add_job(self, job):
        if job not in self.jobs:
            print("Adding job %s... Successful: Customer Name %s, Customer
Contact %s"%(job.job_id, job.customer.name, job.customer.contact))
            self.jobs.append(job)
        else:
            print("Adding job %s... Failed: Job %s already
exists!"%(job.job_id,job.job_id))

    #def job_notification(self, job, notifier):
    #    self.jobs[self.jobs.index(job)].send_notification(notifier)

    def finish_job(self, job):
```

```python
        del self.jobs[self.jobs.index(job)]

    def job_notification(self, job, notifier):
        notifier.notify(job.customer)

from abc import ABC,abstractmethod


class Notifier(ABC):

    @abstractmethod
    def notify(self, customer):
        pass


class EmailNotifier(Notifier):

    def notify(self, customer):
        print("""Job finished. Sending Email to %s....
                Dear %s, your car is ready to be collected!"""
                %(customer.contact, customer.name))


class SMSNotifier(Notifier):

    def notify(self, customer):
        print("""Job finished. Sending SMS to %s....
                Dear %s, your car is ready to be collected!"""
                %(customer.contact, customer.name))

from JobList  import JobList

class CarWashSystem(object):

    def __init__(self):
        self._job_list = JobList()

    def new_wash_job(self, job):
        self._job_list.add_job(job)

    def finish_wash_job(self, job, notifier):
        #self._finish_job_notification(job, notifier)
        self._job_list.finish_job(job)
        self._job_list.job_notification(job,notifier)
```

Question 3 –

Question 4 –

END OF EXAM PAPER