BCT 2309 OOSAD: CAT- June 2020

Peter M. Mwanzia SCT212-0067/2017.

Q1. a) **Explain how the object-oriented concepts of polymorphism and encapsulation encourage good designs in software development.**

Encapsulation is the wrapping up of data into a single unit, it organizes data in classes where the data can only be accessed by member function of the class. Encapsulation increases flexibility, reusability of coding and testing is easily carried out.

Polymorphism is the ability is the ability of a message to be displayed in more than one form, it helps perform a single action in different ways. A good design should have the minimal code to implement a functionality, polymorphism helps achieve this with method overloading and operator overloading. It helps us define one interface and have multiple implementations.

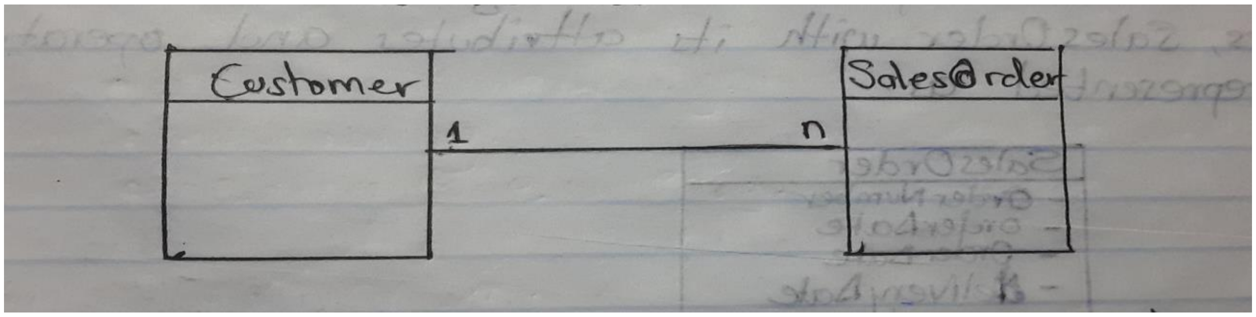
b) **What is a *use case* in regard to systems development? Discuss the importance of use cases in system development**

A use case is a list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal. Use case modelling is employed to analyze functional requirements of a system. Use cases makes the system requirements traceable, traceability is an important aspect of system documentation as it helps an analyst to see how different models are interconnected, making maintenance and modification(incase user requirements change) of the system easier.

c) **With aids of practical modeling examples illustrate ways can we depict the following in the unified modeling language (UML): -**

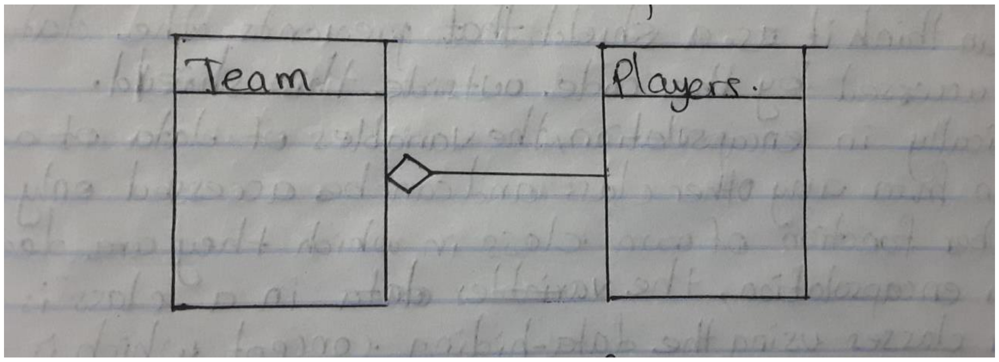
a. Something is associated with many other things

In an association between a customer and a sales order, the customer class has a role client and the sales order has the role transaction. The multiplicity of customer is one and that of the sales order is ***n.*** This means that a customer object can relate to several sales order transactions, however a sales order must relate to one and only one customer. This can be illustrated as follows in UML:



b. Something is a part of something else.

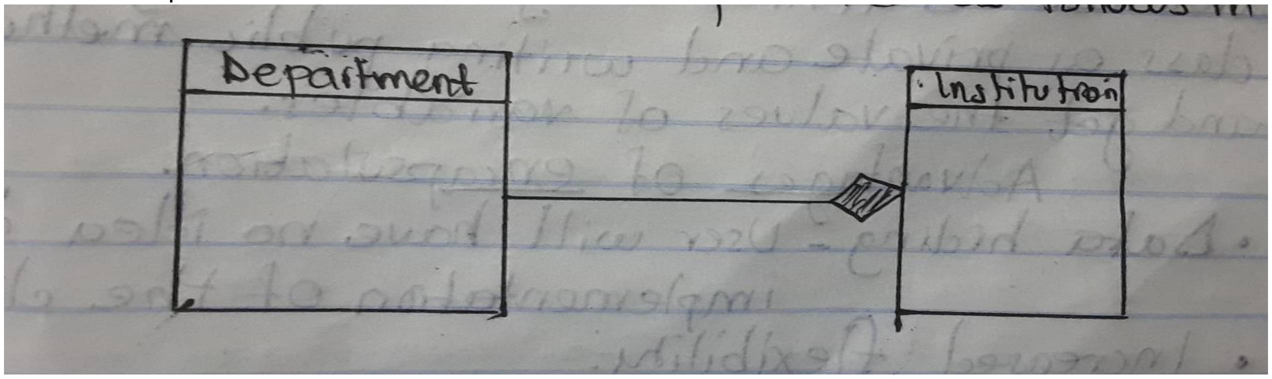
Aggregation which is a stronger form of association which forms the whole part relationship for instance a team consists of players, the team is the *whole* and the player is the *part* in this relationship. If there are no players, there is no team. In UML this can be represented as:



c. Something is a kind (or type) of something else.

Composition which is a stronger form of aggregation in which an aggregate is useless without its part and a part is meaningless without the aggregate example a department and an institution.

This can be represented as follows in UML.



d) **Explain the term *modularity* as used in systems design. Give two suitable examples of its application.**

Modularity refers to dividing a system into smaller chunks or modules Modularity can also be defined as the degree to which a system's components may be separated and recombined. Modular programming emphasizes on separating the functionality of a program into independent, interchangeable modules. For instance:

- Computer/cellphones use modularity to overcome the changing customer demands and thus making the manufacturing process adaptive to change.
- Enterprise Resource planning (ERP) it is a business management software that allows an organization to use a system of integrated resources to manage businesses and automate functions

e) **Describe the term *requirements traceability* and show its importance in development of systems.**

Requirement traceability is the ability to describe and follow the life of a requirement in both forward and backward direction that is from its origin through to its development and specification to its subsequent deployment and use and through periods of refinement and iteration in any of these phases.

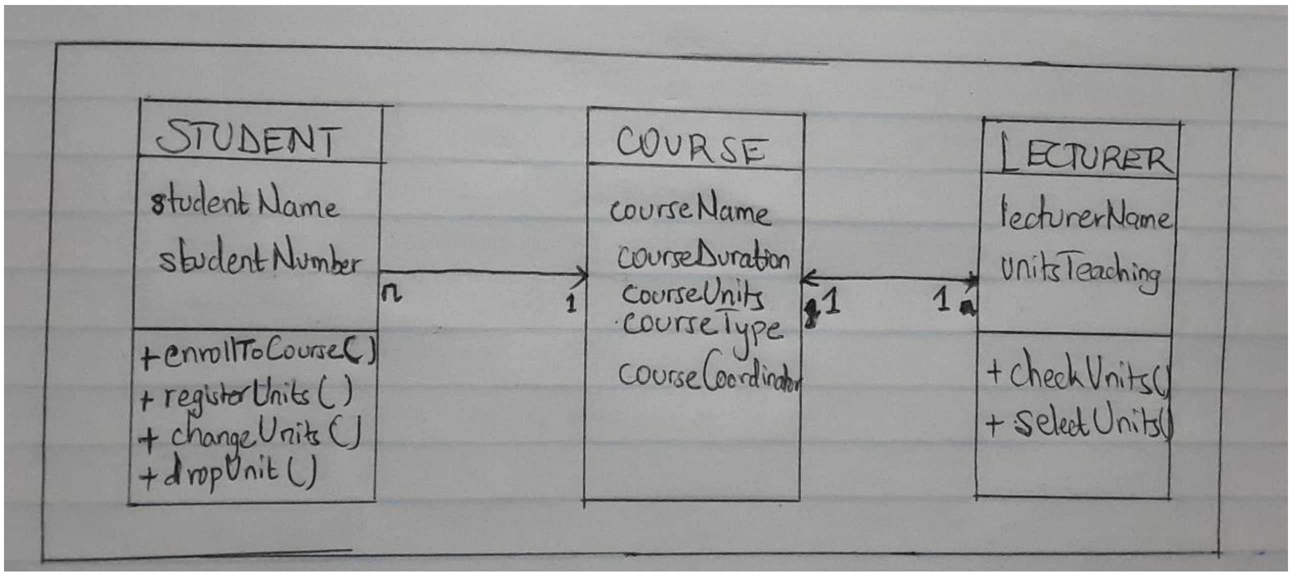
Traceability is an important aspect of system documentation, it allows analysts to see how different models are interconnected, making maintenance and modification of the system easier.

Q2.a) **What is an operation? How does it contribute to the concept of a class and what are the elements that make up an operation**?

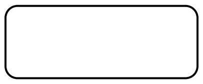
An operation is a behavioral feature that may be owned by an interface, data type or class. A class consists of operations that describe the behavior of the class. An operation is the sum of the signature and the method, where the method is the code part that describes the logic required to achieve the behavior promised by the operation.

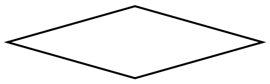
b) **Using the narrative below, identify the classes and their characteristics i.e. attributes and operations (where applicable) and hence draw the corresponding class diagram.**

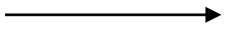
In a student registration system each student has a student number and a name. Each student must be enrolled on a particular course e.g. BSc Computer Technology. Courses have duration (number of semesters) and may be part time or full time. Each course contains a number of units that student selects. Lecturers teach one or more units and some are appointed as course coordinators .A lecturer may only be the coordinator of one course at a time. The system should allow students to select, change or drop units while lecturers should be able to check ad select the units they wish to take.

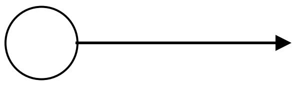


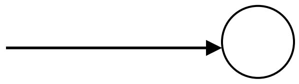
Q3 a). **Name and describe each of the elements of an activity diagram, including their meanings and the symbols used to represent them. Describe the benefits of using activity diagrams.**

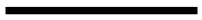

Rounded box: contains the name of the activity.


Decision diamond: shows decision.


Transition arrow: shows the direction of the flow of the activities.


Start marker: shows the initial activity.


End marker: shows the last activity.

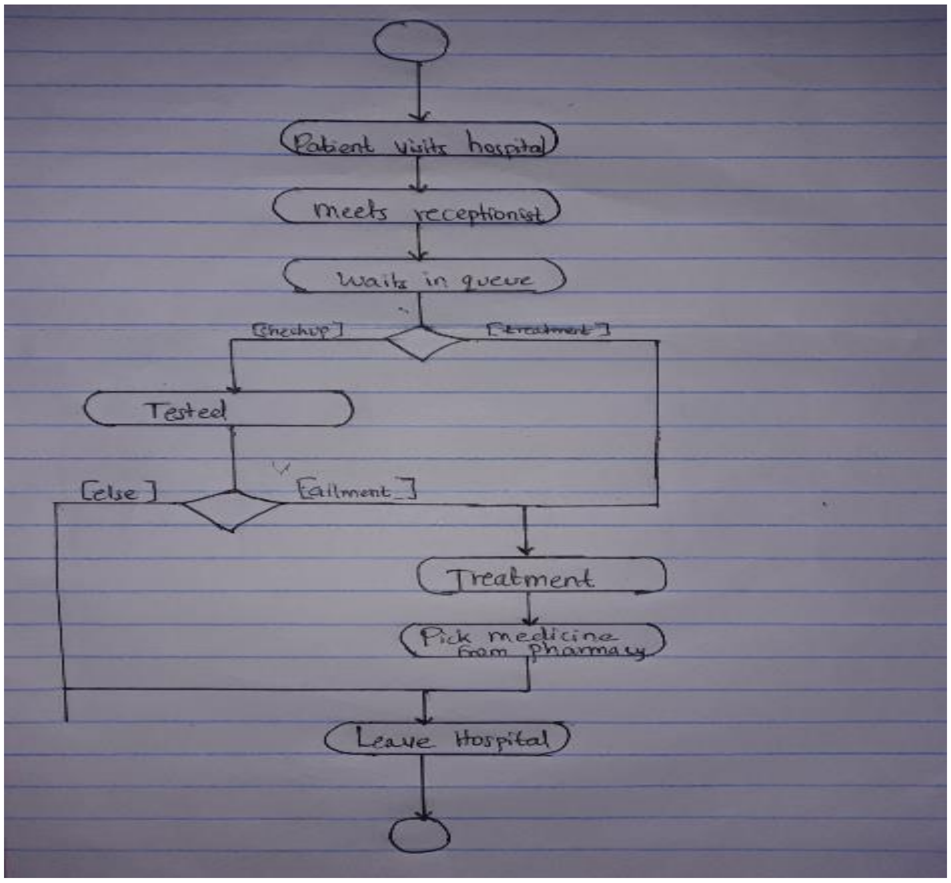

Synchronization bar: shows activities that takes place in parallel.

Benefits of using an activity diagram

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation.

b) **Draw a fully adorned UML activity diagram based on the following narrative.**

A patient may visit a hospital for a check up or for treatment. If she comes for a check up or for treatment the patient must meet the receptionist to record her visit. In either case the patient must join the waiting queue for her turn. In case of a check up the lab technician conducts the test and passes the results to the doctor. If ailment is detected the patient must be treated otherwise she I allowed to leave the hospital. If the patient came for treatment then she sees the doctor for treatment then picks her medicine from the pharmacy before she leaves the hospital.


Patient visits hospital
meets receptionist
Waits in queue
[checkup]
[treatment]
Tested
[else]
[ailment]
Treatment
Pick medicine from pharmacy
Leave Hospital