BCT 2309 OOSAD: CAT- June 2020 (45 marks)

**HEZION KWENA**

**SCT212-0222/2017**

**Q1.** a) Explain how the object-oriented concepts of polymorphism and encapsulation encourage good designs in software development. (3 marks)

Object-oriented programming refers to the concept in high-level languages such as Java and Python that uses Objects and classes in their implementations. OOP has four major building blocks which are, Polymorphism, Encapsulation, Abstraction, and Inheritance.

Polymorphism means existing in many forms. Variables, functions, and objects can exist in multiple forms while Encapsulation is a programming style where implementation details are hidden.

* Encapsulation reduces software development complexity greatly since only methods are exposed. The programmer does not have to worry about implementation details but is only concerned with the operations. For example, if a developer wants to use a dynamic link library to display date and time, he does not have to worry about the codes in the date and time class rather he would simply use the data and time class by using public variables to call it up.
* Polymorphism in Object-oriented programming serves to enforce simplicity, making codes more extendable and easily maintaining applications. example of Polymorphism in Object-oriented programing is a cursor behavior. A cursor may take different forms like an arrow, a line, cross, or other shapes depending on the behavior of the user or the program mode. With polymorphism, a method or subclass can define its behaviors and attributes while retaining some of the functionality of its parent class. This means you can have a class that displays date and time, and then you can create a method to inherit the class but should display a welcome message alongside the date and time.

b) What is a ***use case*** in regard to systems development? Discuss the importance of use cases in system development (5 marks)

In software and systems engineering, a **use case** is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language (UML) as an *actor*) and a system to achieve a goal. The actor can be a human or other external system. A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity.

**IMPORTANCES**

* Use case can benefit developers by revealing how a system should behave while also helping identify any errors that could arise in the process.
* The list of goals created in the use case writing process can be used to establish the complexity and cost of the system.
* By focusing both on the user and the system, real system needs can be identified earlier in the design process.
* Since use cases are written primarily in a narrative language they are easily understood by stakeholders, including customers, users and executives -- not just by developers and testers.
* The creation of extending use cases and the identification of exceptions to successful use case scenarios saves developers time by making it easier to define subtle system requirements.
* By identifying system boundaries in the design scope of the use case, developers can avoid scope creep.
* Premature design can be avoided by focusing on what the system should do rather than how it should do it.

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

* 1. Something is associated with many other things

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:

n

1

SALES ORDER

CUSTOMER

1. 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:

PLAYERS

TEAM

1. Something is a kind (or type) of something else (6 marks)

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.

INSTITUTION

DEPARTMENT

d) Explain the term ***modularity*** as used in systems design. Give two suitable examples of its application. (3 marks)

-This is a design practice that subdivides a system into smaller parts called *modules* which can be independently created, modified, replaced or exchanged between different systems

**EXAMPLES**

* 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
* Modularity enables programmers to use the same code in the module over and over again by simply referencing the code to perform the specific action in the module at different locations of the program. Creating units or classes ensures this. Thus, enabling reusing of code.
* A program with millions of lines of code will present a huge task if not in a module when debugging is required. You can imagine what it is like to search through such a huge environment to look for errors in a program. Having each task in its discrete module makes the process a lot easier to carry out. Thus, **A faulty function can easily be checked for errors when in a module.**

e) Describe the term ***requirements traceability*** and show its importance in development of systems. (3 marks)

Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases.)

**IMPORTANCE**

* Helps ensure the team is not only building the product right, but are also building the right product.
* All goals are traced as per the completeness of requirements
* Change management ensures easier impact analysis, and improved risk management
* Helps avoid quality and acceptance problems
* Enables further improvement of both the product and its development processes
* Save time, costs, and effort during development
* Reduces wasted time due to fewer failures, and a decreased chance of project overrun
* Helps ensure compliance with industry standards (ISO, IEC, FDA, CMMI, etc.)
* Promotes quicker impact analysis for risk time and cost deviations

**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? (3 marks)

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.* (8 marks)

1

1

1

n

+checkUnits()

+selectUnits()

lecturerName

unitsTeaching

+enrollToCourse()

+registerUnits()

+changeUnits()

+deferUnit

studentName

studentNumber

courseName

courseUnits

courseDuration

courseType

courseCoordination

STUDENT

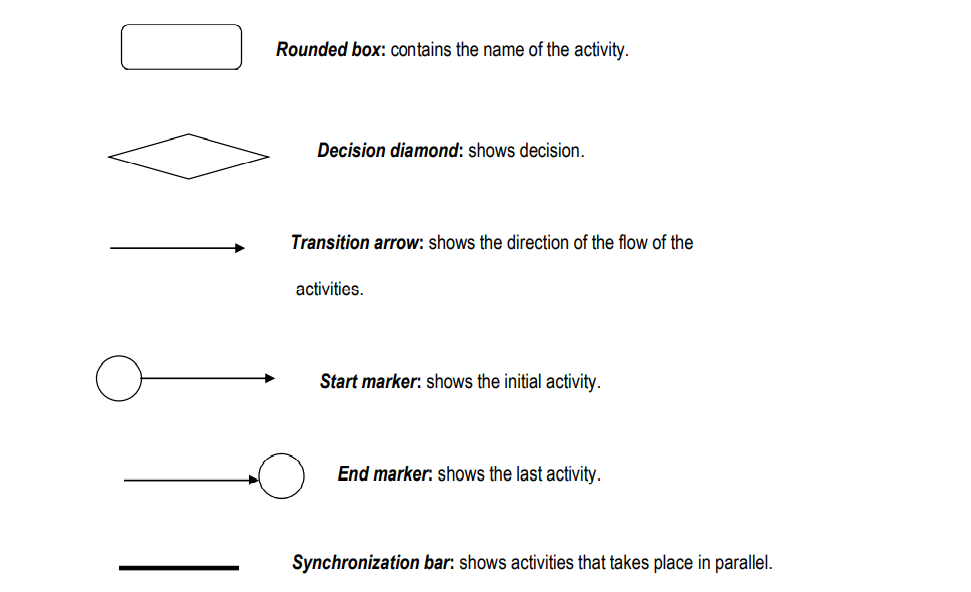
LECTURER

COURSE

**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. (6 marks)

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.* (8 marks)

**