**NAME: UYE TIMILEYIN**

**MATRIC NO: 17/SCI01/081**

**CORSE CODE: CSC 302**

**QUESTION**

1. With the aid of a shema distinguish between a modular and an object oriented programming paradigm.

2. Use this schema to illustrate the modifications  required to translate an airline reservation program from a modular to an object oriented design

**SOLUTION**

QUESTION 1

Difference between Procedural Programming and Object Oriented Programming:

| MODULAR PROGRAMMING | OBJECT ORIENTED PROGRAMMING |
| --- | --- |
| In this programming, program is divided into small parts called *functions*. | In object oriented programming, program is divided into small parts called *objects*. |
| Modular programming follows *top down approach*. | Object oriented programming follows *bottom up approach*. |
| There is no access specifier in modular programming. | Object oriented programming have access specifiers like private, public, protected etc. |
| Adding new data and function is not easy. | Adding new data and function is easy. |
| Modular programming does not have any proper way for hiding data so it is *less secure*. | Object oriented programming provides data hiding so it is *more secure*. |
| In Modular programming, overloading is not possible. | Overloading is possible in object oriented programming. |
| In modular programming, function is more important than data. | In object oriented programming, data is more important than function. |
| modular programming is based on *unreal world*. | Object oriented programming is based on *real world*. |
| Examples: C, FORTRAN, Pascal, Basic etc. | Examples: C++, Java, Python, C# etc. |

**QUESTION 2**

Using the schema translate an airline reservation program from a modular to an object oriented design

|  |  |  |
| --- | --- | --- |
| **s/n** | **Modular Programming Paradigm** | **Object Oriented Programming Paradigm (OOP)** |
| 1 | Here, It is known that modular programming uses functions and each functionalities of the airline reservation system will be categorized into modules in a way that every module contains information necessary for executing the system proposed. For example the functionality that allows for the picking of random seats is able to function exclusively | This program gives us a chance to imitate real life airline reservation structure. |
| 2 | 1) What information is needed -  the personal information of the passenger(s) is essential for instance their age, name, marital status, height  -The program needs to know the information about the flight to be  booked such as: where the passenger(s) are leaving from, the passenger(s) destination, the cost of tickets bought by a passenger , if a return ticket is going to be bought etc          2) What is to be done with the information –  The program will use personal information to create a passenger’s profile , from this profile using some algorithms other reservation suggestions may be made to the customer. -The program will use the flight information to know which flight to be picked for a customer and when it should be picked for the customer  3) What is expected(Outputs) The output of this program may include: - The flight ticket(s) -The general flight details etc | ENTITIES  The customer entities The customer is very key and the most important object. Every other entity in the model revolves around what the customer is and what it can do. This is the key player because it is at the center of any object relationships that can exist. Here are the defining attributes (properties) and Relate  Functionality (The Methods) of the Employee Entity. -Defining Attributes: The customer entity will need an customerID, his name, address, city, state, zip code, telephone number, email and hourly rate attributes. -Related Functionality: The customer needs to be able to do several things as far as it's defining attributes go as well as the ability to call the functionality of the other entities for timekeeping and and calculations. As such, the methods needed will be. EntercustomerID, Savecustomer, Loadcustomer, Findcustomer and Printcustomer. 2) The flight entities The flight Entity should be made to work a week based format where all the days are there so that it can be easy to get a quick overview of what the week is like currently. We will add a method to our object to make sure that the airlines flight data is completely entered before we go ahead and print the flight times just as a precaution because of the importance of the data that is being handled. -Defining Attributes: When you think of flight data, it doesn't take too long to determine the general information that you would need. The properties are: planeID, DayOfWeek, DayDate, landingTime, maintenanceTime, refuelingTime, timeOfNextFlight. customerID is needed to connect a flight record to an customer record in the master employee data file. The rest of the properties specifically relate to the flight Entity itself. -Related Functionality: As mentioned in the problem description, the flight entity will need to perform several types of actions. The names of the methods described here should help state clearly what the methods which helps make the object definition that much clearer. such, here are these methods: GetFlightData, SaveFlightData, LoadFlightData, PrintFlightData, WeeklDataCompleted, PrintWeeklyFlightData, CalculateRegularHours, CalculateOverTimeHours, CalculateHolidayHours, CalculateRegularAmount, CalculateHolidayAmount 3) The report system entities The reporting system is present because on an entity based problem solving approach, every method needs to find it's place withing an object model. In most cases, printing related functionality is very often isolated into a separate entity and sometimes even an independent application (so that it can be executed on a separate system on the network and print the report while users can continue to do their other activities uninterrupted by the printing process). -Defining Attributes: Since the report system entity creates no data files, all it needs is three attributes to perform it's task. These attributes are the customerID, a StartDate and an EndDate properties so that it can accumulate all the flightRecords that fall between these two dates.    -Related Functionality: Ultimately, we could have provided all printing functionality in the report system entity, which means that the printing of the customer Data could have also been added as a related functionality. |
| 3 | Here, the modular approach is closely related to the functionality need in problem identification. This involves categorizing into smaller specific procedures and functions is how Modular Programming offers to manage the complexity of a program. | The OOP Approach to problem solving differs from the Modular Approach. With the extensive knowledge we have on both methods |