ADESINA ALAMEEN .B

18/SCI01/099

CSC 302

SURVEY OF PROGRAMMING LANGUAFE

SCHEMA TO DISTINGUISH BETWEEN MODULAR AND OBJECTED ORIENTED PROGRAMMING PARADIGM

S/n		Modular Programming Paradigm	Object Oriented Programming Paradigm (OOP)
1	Definition	Modular Programming (aka 'stepwise refinement' and 'top-down design' paradigm) is a software designing technique that emphasizes separating the functionalities of a program into independent and meaningful **modules**, such that each module contains everything necessary for executing the one (and only one) aspect of the desired functionality!	An object oriented program contains different types of objects, each corresponding to a complex real world objects or any complex data or a concept such as a bank customer, a bank account or any departmental store.
2	How it works	Modular programming was devised in a predominately procedural world. Modules drew a box around a set of procedures and required that box to be independently deploy-able. It was an organization scheme for keeping procedures neat and tidy. Different languages might offer different mechanisms for implementing that	Objects form a box around a set of procedures, which qualifies them as modules. But they do much more than that, and some languages use the term module to refer to even higher levels of code organization.

		scheme.	
3	How it solves problems	In the modular programming approach, the verbs in the problem description are the ones considered first. Typically, these verbs represent the work to be done and therefore the likely subroutines that we would have to implement in order to solve the problem. The names in the description represent the information (variables and datatypes) that will will be acting on or producing at the end. Modular programming is based on three things, which are: 1. The information needed before the process can start, 2. The work to be done with that information and 3. The expected output of a procedure.	In Object Oriented Programming, the names, not the verbs, are the first to be considered because in O.O.P. the first task is to define entities, not actions. O.O.P. starts by defining the players (or actors if you will) of a given scene (the problem) and them proceeds to defining how the actors are described (the properties) and what the actors can do (the methods). Once all these are defined for all playing actors, then the scene (the main part of the program controls what each players has and what it does. Only when you are defining the methods of the object can you actually start answering the same three questions that the modular approach lets you define right from the start.

Translating an airline reservation program from a modular to an object oriented design

S/N	Modular Programming Paradigm	Objected Oriented Programming Paradigm
1	The functionalities of the airline reservation program will be separated into independent and meaningful modules such that each module contains everything necessary for executing the one (and only one) aspect of the desired functionality. For example the functionality that allows for the picking of random seats is able to function exclusively	The program allows for the reservation program to mirror real life airline reservation structure. Eg, each plane seat to be reserved is an object
2	1) **What information is needed** -The program need to know the personal information of the passenger(s) such as their age, name, marital status, height -The program needs to know the information about the flight to be	**THE ENTITIES** 1) **The customer entities** The customer is the main actor of the scene. 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 Related

	booked such as: where the passenger(s) are leaving from, where the passenger(s) are going to, how much the passenger is willing to spend on a seat, if the passenger is buying a return ticket etc 2) **What is to be done with the information** -The program will use the personal information part to create and keep a customer profile, from this profile using some algorithms other travel/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	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. Au 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	As you can see, the modular approach is closely related to the functionality needed in the problem description. You think in terms of	As you can see, the Object Oriented Approach to problem solving is quite different from the Modular Approach. With all this information you now have on the two methods, you might be wondering if there are certain

	what needs to be done and usually can pretty quickly devise a workable complete solution to a problem by consecutively answering these three questions (some of these answer may require that you breakdown the system into smaller answers to a subset of these three question depending on how complex the system gets. This breaking down into smaller more specific procedures and functions is how Modular Programming offers to manage the complexity of a program.	projects, or certain parts of a big project that could benefit from the Modular Approach and likewise for the Object Oriented Approach. Is there situations where you would be better off using one method over another. In this next section, I will discuss this subject, based on my own personal experiences with both methods.