NASIR FIRDAUS OPEYEMI
17/sci01/051
CSC 302

1. Firstly, Modular programming and object programming are two safe approaches to the logical organisation of a program, permitting the reusability and the modifiability of software components. Programming with objects in Objective CAML allows parametric polymorphism (parameterized classes) and *inclusion/subtype polymorphism* (sending of messages) thanks to late binding and subtyping, with restrictions due to

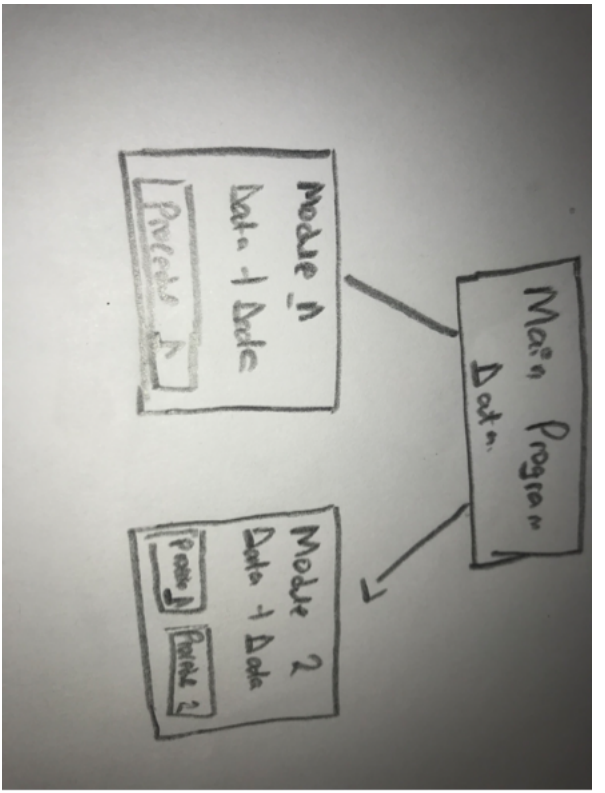
equality, facilitating incremental programming.

Modular programming allows one to restrict parametric polymorphism and use immediate binding, which can be useful for conserving efficiency of execution.

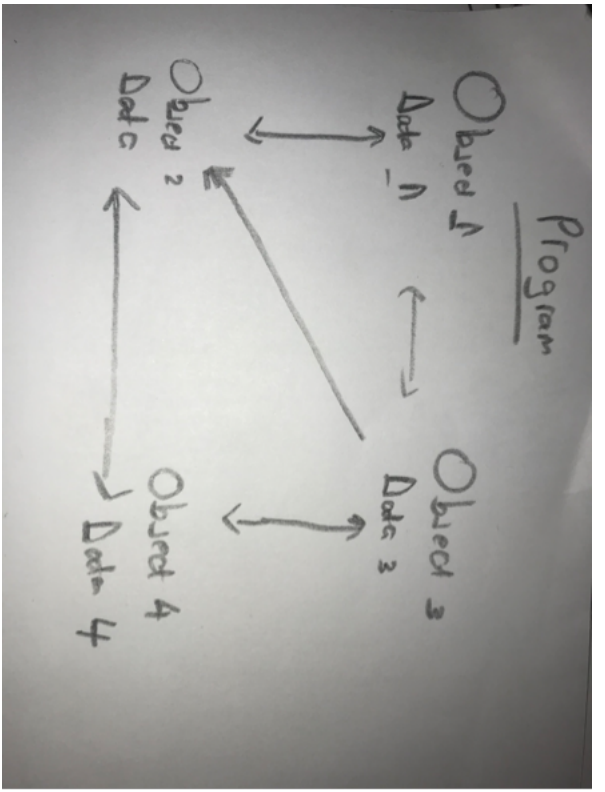
The main difference between modular programming and object programming in Objective CAML comes from the type system.The modular programming model permits the easy extension of functions on non-extensible recursive data types. If one wishes to add a case in a variant type, it will be necessary to modify a large part of the sources.

The object model of programming defines a set of recursive data types using classes. One interprets a class as a case of the data type.

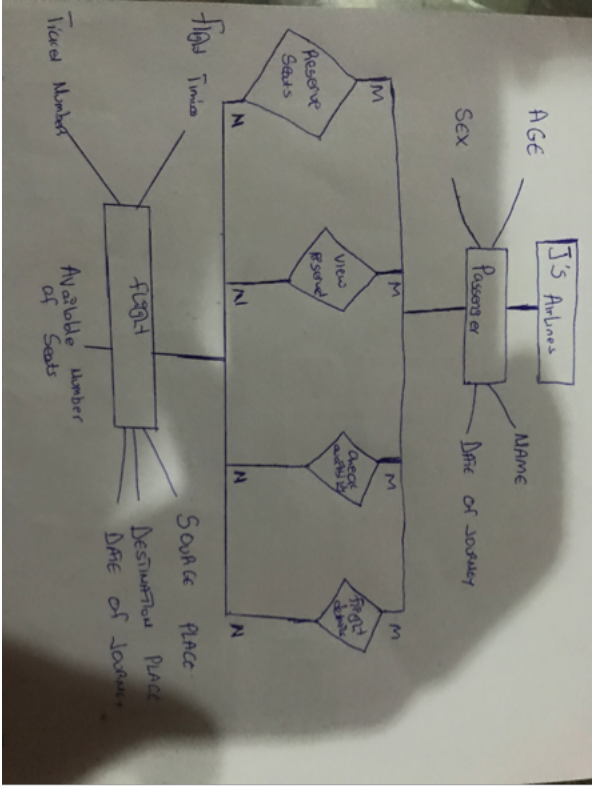
Below is a scenario of a modular programming paradigm



Below is also a schema of OOP paradigm:

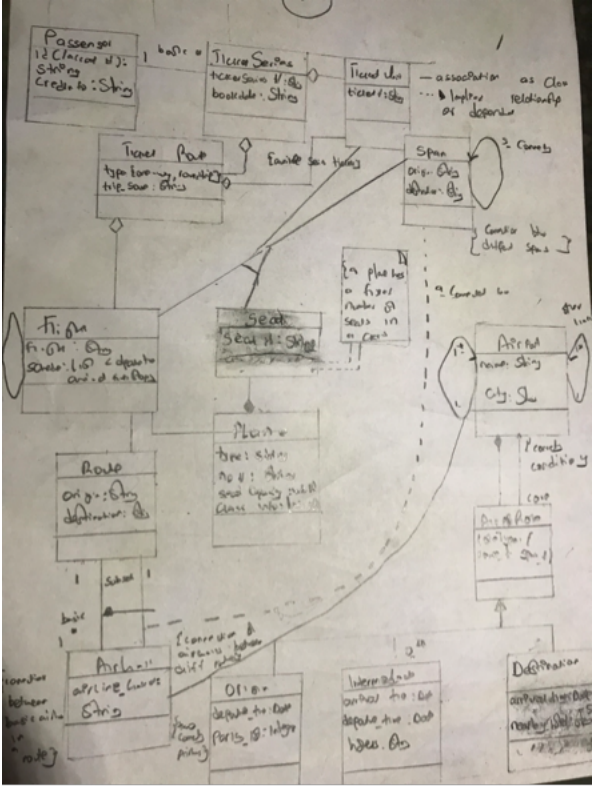


2. Below is a designed schema for a modular design which will further be converted into the object oriented programming design.



Below is the converted airlines schema from modular to object oriented design

Below is Ann OOP schema approach for an airline



In conclusion,
The above approach involves the use of object-oriented methods and Semantic Analysis Patterns. By solving this type of problems using object-oriented methods we reap the general benefits of this approach, i.e., reusability, extensibility, and conceptual abstraction. It is recognized

by researchers and practitioners that object-oriented methods are superior to procedural approaches for handling complex systems.

This advantage extends to our approach. The general use of patterns is considered an advance in object-oriented methods because patterns distill the knowledge and experience of many developers and are highly reusable. Patterns also improve software quality because they have been scrutinized by many. Our Semantic Analysis Patterns have been shown to ease the task of building conceptual models by directly translating functional aspects of an

application [Fer00a] and can also be used to define Secure SAPs, where the functionality is complemented with authorization and authentication aspects [Fer07]. In this paper we have shown, through a case study, the ability of SAPs to compose patterns to build complex patterns or complex models in general.

The component patterns realize the specifications of the system. While experiments with actual projects are necessary to prove the practicality of this approach, we can say that this methodology is a better way to build complex systems than procedural

programming or ad-hoc object-oriented methods. We have also shown our approach to be convenient to improve practical approaches such as XP [Fer03], which is another proof of its possible value. There are other object-oriented approaches based on patterns, e.g., several approaches are discussed in [Sia01], and we don‟t claim that our approach is better than any of these methods, this would require a detailed and lengthy study. We do claim that our approach allows us to build complex models in a convenient and error-free way.