**ITELIMA FAITH IBIFUBARA HAPPINESS**

**18/SCI01/106**

**CSC 302**

Main Program

Module 1.1

Module 1

Module 2

Module 3

Module 1.2

Module 1.3

Module 3.1

Module 3.2

**Fig. 1.1: A schema illustrating modular programming paradigm**

**CLASS**

|  |
| --- |
| **OBJECTS** |
| Instance of the class |

|  |
| --- |
| **CLASS VARIABLES** |
| Belong to the class |

|  |
| --- |
| **INSTANCE** |
| Variables belong to the object |

|  |
| --- |
| **METHODS** |
| Functions of a class |

**Fig. 1.2: A schema illustrating object oriented programming paradigm**

Modular programming: The idea of modular programming is to sub-divide a program into smaller units that are independently testable and that can be integrated to accomplish the overall programming objective. 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!

Object oriented programming: The objectoriented approach is allegedly more flexible, by separating a program into a network of subsystems, with each controlling their own data, algorithms, or devices across the entire program, but only accessible by first specifying named access to the subsystem object-class, not just by accidentally coding a similar global variable name. Rather than relying on a structured-programming hierarchy chart, object oriented programming needs a call-reference index to trace which subsystems or classes are accessed from other locations. 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.

Airline Reservation Program

Ticket reservation

Reservation

Information

Administrator

View

Cancellation

Airbus

Fare discount

Flight schedule

New

Choose Flight

Search

Traveller Details

Confirmation

1.

**Fig. 2.1: Modular programming representation**

|  |
| --- |
| **Ticket series** |
| ticketseries: string**Fig. 2.2: Object Oriented programming representation** |
| bookdate: string |
|  |

|  |
| --- |
| **Ticket unit** |
| id: int |
|  |

|  |
| --- |
| **Passenger** |
| id: int |
| Creditinfo: string |
|  |

|  |
| --- |
| **Ticket route** |
| type:{one-way, round-trip} |
| trip\_source: string |
| trip\_destination:string |
|  |

|  |
| --- |
| **Span** |
| origin: string |
| destination: string |
|  |

|  |
| --- |
| **Flight** |
| flight: string |
| schedule: list<departure\_time, arrival\_time, stop> |
|  |

|  |
| --- |
| **seat** |
| seat: string |
|  |

|  |
| --- |
| **Plane** |
| type: string |
| number:string |
| seatCapacity:infoList |
| classInfo:infoList |
|  |

|  |
| --- |
| **Airport** |
| name: string |
| city: string |
|  |

|  |
| --- |
| **Route** |
| origin: string |
| destination: string |
|  |

|  |
| --- |
| **Airlink** |
| airline\_link: string |
|  |

|  |
| --- |
| **AirportRole** |
| roleType:{route\_t, span\_t} |
|  |

|  |
| --- |
| **Span** |
| departure\_time: date |
| parkingLot: int |
|  |

|  |
| --- |
| **Destination** |
| arrival\_time: date |
| nearby\_hotel: string |
|  |

|  |
| --- |
| **Intermediate** |
| arrival\_time: date |
| departure\_time: date |
| hotels:string |
|  |