University of Mohamed Boudiaf M'sila
Faculty of Mathematics and Computer Science
Department of Computer Science
Master 01 SIGL

Duration: 2h (120 Minutes) Instructor Prof. Hichem Debbi

Object Analysis and Design(ACO)
Final Exam
Saturday 10, 2026

True/False (03 points)

Write True if the statement is true, otherwise write False.

_____ meat-modeling can help to create Profiles

_____ Iterator can use both mediator and observer. .

_____ Objects are equal bu not identical

_____ Adornments are Extended modelling elements

_____ In XP methodology, user stories can be updated in the productionizing phase

_____ EJB has a profile .

Section 2. Short Answer(05 points)

1. Explain the difference between Sealed, Abstract and Static classes?
2. What is XP -Pair programming ?
3. Cite four iterative and incremental methodologies ?
4. What is the main purpose of the third micro process activity, their steps, and the UML diagrams to be used.

Section 3. OCL Specification(04 points)

5. Introduce in OCL the following constraint given the Class diagram presented below:
   - A patient has at least 18 Years old
   - A patient can only pay his bills, not the bills of others.
   - A doctor can be assigned to one department at a time. Considering the doctor as a context, provide two versions, one using sample relation supposing that we have a direct relationship between doctor and department, and the second by using **oclIsKindOf**.
   - Implement in C# the aggregation relation between Department and Employee?

Section 4. Pattern Implementation(08 points)

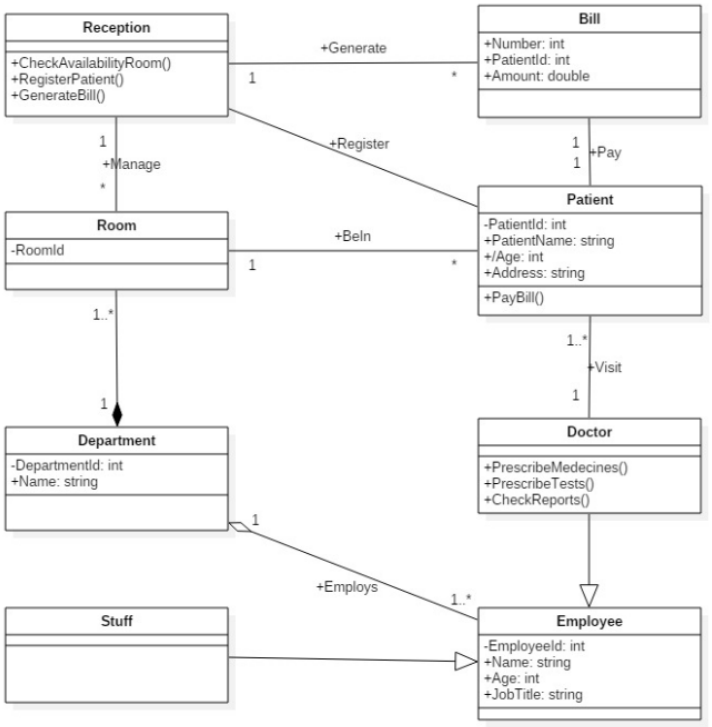

Figure 1: Class diagram of an Hospital Management System

6. In an airport, we need a reliable way for aircraft to communicate with the Air Traffic Control Tower and with each other. We want to develop a communication system that handles different types of aircraft (Airbus and Boeing) with their specific communication needs.
   We consider the following conditions:

   - The **Airport Tower** acts as the mediator for two distinct aircraft types: **Airbus A380** and **Boeing 787** .
   - The mediator supports both broadcast messages (to all aircraft) and direct messages (to specific aircraft)
   - Airbus A380 aircraft receive only Boeing aircraft, and Boeing 787 receives messages from both aircraft under the condition that the sender and receiver are not the same. In case of a message containing "emergency", the aircraft receives from itself.
   - The mediator keeps the log of activities as a list of strings
   - While **TuneToFrequency** is the SignOn method of all colleagues, **Transmit** is the send method.

   Given the example output depicted in Figure 3 for three (03) airplanes, and the class diagram presented below, provide a C# implementation for this problem.

   Introduce a code that iterates over the airplanes and show their information.

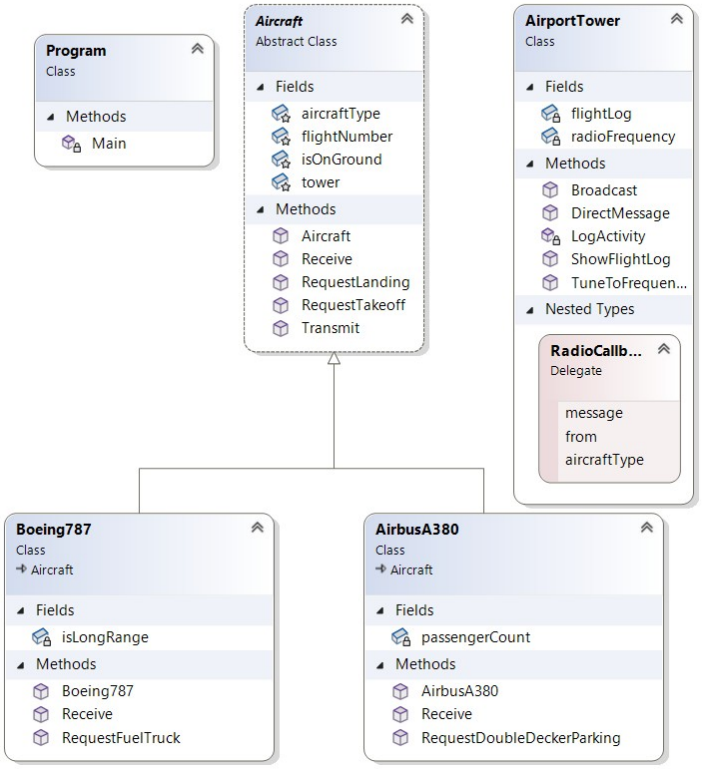

Figure 2: Mediator Class diagram

```
[Airbus A380 TK202 transmits]: Requesting landing clearance
[Boeing 787 AH101 received from ]: Airbus A380 TK202: Requesting landing clearance
[Boeing 787 QR789 received from ]: Airbus A380 TK202: Requesting landing clearance

[Tower to TK202]: Cleared for landing on runway 27L (from TOWER TOWER)

[Boeing 787 QR789 transmits]: Requesting takeoff clearance
[Airbus A380 TK202 receives Boeing traffic]: QR789: Requesting takeoff clearance
[Boeing 787 AH101 received from ]: Boeing 787 QR789: Requesting takeoff clearance

[Tower to QR789]: Cleared for takeoff from runway 27R (from TOWER TOWER)


--- Emergency Scenario ---
[Boeing 787 AH101 transmits]: MEDICAL EMERGENCY ON BOARD - requesting priority handling
[Airbus A380 TK202 receives Boeing traffic]: AH101: MEDICAL EMERGENCY ON BOARD - requesting priority handling
[Boeing 787 AH101 - received PRIORITY MESSAGE] from: Boeing 787 AH101: MEDICAL EMERGENCY ON BOARD - requesting priority
handling
[Boeing 787 QR789 - received PRIORITY MESSAGE] from: Boeing 787 AH101: MEDICAL EMERGENCY ON BOARD - requesting priority
handling
```

Figure 3: Mediator Output Example

Answer Key for Exam A

True/False (03 points)

Write True if the statement is true, otherwise write False.

False meat-modeling can help to create Profiles

False Iterator can use both mediator and observer. .

True Objects are equal bu not identical

False Adornments are Extended modelling elements

True In XP methodology, user stories can be updated in the productionizing phase

True EJB has a profile .

Section 2. Short Answer(05 points)

1. Explain the difference between Sealed, Abstract and Static classes?

   Answer: Their methods and attributes.

2. What is XP -Pair programming ?

   Answer:

3. Cite four iterative and incremental methodologies ?

   Answer:

4. What is the main purpose of the third micro process activity, their steps, and the UML diagrams to be used.

   Answer:

Section 3. OCL Specification(04 points)

5. Introduce in OCL the following constraint given the Class diagram presented below:

   - A patient has at least 18 Years old
   - A patient can only pay his bills, not the bills of others.
   - A doctor can be assigned to one department at a time. Considering the doctor as a context, provide two versions, one using sample relation supposing that we have a direct relationship between doctor and department, and the second by using **oclIsKindOf**.
   - Implement in C# the aggregation relation between Department and Employee?

   Answer: • context Patient inv: self.Age ¿= 18

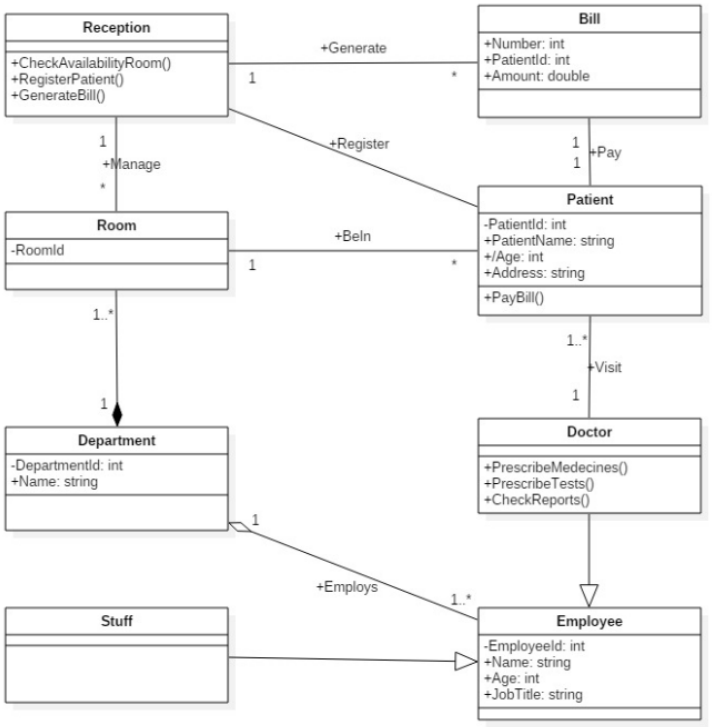

Figure 4: Class diagram of an Hospital Management System

- context Patient inv PayBills: self.bill-¿forAll(b — b.PatientId = self.PatientId)
  context Doctor
  inv OneDepartment:
  self.department-¿size() = 1

  context Doctor
  inv DoctorInOneDepartmentOnly:
  Department.allInstances()
  -¿select(d — d.employee-¿exists(e — e = self and e.oclIsKindOf(Doctor)))-¿size() = 1

Section 4. Pattern Implementation(08 points)

6. In an airport, we need a reliable way for aircraft to communicate with the Air Traffic Control Tower and with each other. We want to develop a communication system that handles different types of aircraft (Airbus and Boeing) with their specific communication needs.
   We consider the following conditions:

   - The **Airport Tower** acts as the mediator for two distinct aircraft types: **Airbus A380** and **Boeing 787** .
   - The mediator supports both broadcast messages (to all aircraft) and direct messages (to specific aircraft)
   - Airbus A380 aircraft receive only Boeing aircraft, and Boeing 787 receives messages from both aircraft under the condition that the sender and receiver are not the same. In case of a message containing "emergency", the aircraft receives from itself.

- The mediator keeps the log of activities as a list of strings
- While **TuneToFrequency** is the SignOn method of all colleagues, **Transmit** is the send method.

Given the example output depicted in Figure 3 for three (03) airplanes, and the class diagram presented below, provide a C# implementation for this problem.

Introduce a code that iterates over the airplanes and show their information.

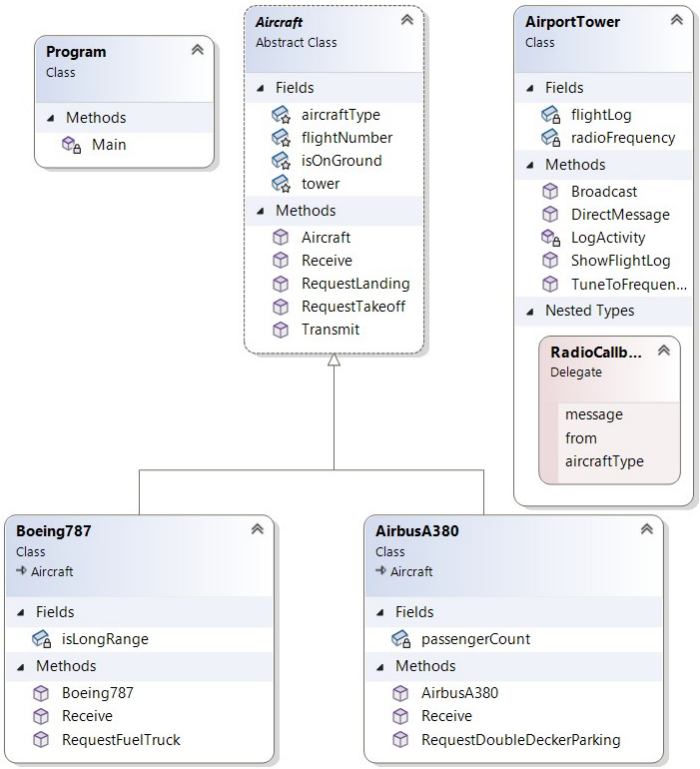

Figure 5: Mediator Class diagram

```
[Airbus A380 TK202 transmits]: Requesting landing clearance
[Boeing 787 AH101 received from ]: Airbus A380 TK202: Requesting landing clearance
[Boeing 787 QR789 received from ]: Airbus A380 TK202: Requesting landing clearance

[Tower to TK202]: Cleared for landing on runway 27L (from TOWER TOWER)

[Boeing 787 QR789 transmits]: Requesting takeoff clearance
[Airbus A380 TK202 receives Boeing traffic]: QR789: Requesting takeoff clearance
[Boeing 787 AH101 received from ]: Boeing 787 QR789: Requesting takeoff clearance

[Tower to QR789]: Cleared for takeoff from runway 27R (from TOWER TOWER)


--- Emergency Scenario ---
[Boeing 787 AH101 transmits]: MEDICAL EMERGENCY ON BOARD - requesting priority handling
[Airbus A380 TK202 receives Boeing traffic]: AH101: MEDICAL EMERGENCY ON BOARD - requesting priority handling
[Boeing 787 AH101 - received PRIORITY MESSAGE] from: Boeing 787 AH101: MEDICAL EMERGENCY ON BOARD - requesting priority
handling
[Boeing 787 QR789 - received PRIORITY MESSAGE] from: Boeing 787 AH101: MEDICAL EMERGENCY ON BOARD - requesting priority
handling
```

Figure 6: Mediator Output Example

```
using System;
using System.Collections.Generic;
// Mediator Pattern - Airport Tower Scenario
// Airport Tower as Mediator for Airbus and Boeing airplanes
class AirportTower
{
// Delegate for communication between airplanes
public delegate void RadioCallback(string message, string from, string
aircraftType);
private RadioCallback radioFrequency;
private List<string> flightLog = new List<string>();
// Airplanes sign on to the tower's frequency
public void TuneToFrequency(RadioCallback method, string flightNumber)
{
radioFrequency += method;
LogActivity($"New aircraft tuned to frequency: {flightNumber} ");
}
// Tower broadcasts message to all aircraft
public void Broadcast(string message, string from, string
aircraftType)
{
LogActivity($"{aircraftType} {from}: {message}");
radioFrequency?.Invoke(message, from, aircraftType);
Console.WriteLine();
}
// Tower sends direct message to specific aircraft
public void DirectMessage(string message, string from, string to,
string aircraftType)
{
LogActivity($"DIRECT: {aircraftType} {from} → {to}: {message}");
Console.WriteLine($"[Tower to {to}]: {message} (from
{aircraftType} {from})");
Console.WriteLine();
}
private void LogActivity(string activity)
{
flightLog.Add($"[{DateTime.Now:HH:mm:ss}] {activity}");
}
public void ShowFlightLog()
{

foreach (var entry in flightLog)
{
Console.WriteLine(entry);
}
}
}
abstract class Aircraft
{
protected AirportTower tower;
protected string flightNumber;
protected string aircraftType;
protected bool isOnGround = true;
public Aircraft(AirportTower tower, string flightNumber, string
aircraftType)
```

```
{
this.tower = tower;
this.flightNumber = flightNumber;
this.aircraftType = aircraftType;
tower.TuneToFrequency(Receive, flightNumber);
}
public virtual void Receive(string message, string from, string
fromAircraftType)
{ }
public void Transmit(string message)
{
Console.WriteLine($"[{aircraftType} {flightNumber} transmits]:
{message}");
tower.Broadcast(message, flightNumber, aircraftType);
}
public void RequestLanding()
{
isOnGround = false;
Transmit("Requesting landing clearance");
tower.DirectMessage($"Cleared for landing on runway 27L", "TOWER", flightNumber, "TOWER");
}
public void RequestTakeoff()
{
Transmit($"Requesting takeoff clearance");
tower.DirectMessage($"Cleared for takeoff from runway 27R", "TOWER", flightNumber, "TOWER");
isOnGround = false;
}
}
class AirbusA380 : Aircraft
{
private int ;
public AirbusA380(AirportTower tower, string flightNumber, int
passengerCount)
: base(tower, flightNumber, "Airbus A380")
{
this.passengerCount = passengerCount;
}
public override void Receive(string message, string from, string
fromAircraftType)
{
// Airbus filters Boeing messages differently
if (fromAircraftType.Contains("Boeing"))
{
Console.WriteLine($"[{aircraftType} {flightNumber} receives
Boeing traffic]: {from}: {message}");
}
else if (!from.Equals(flightNumber))
{
Console.WriteLine($"[{aircraftType} {flightNumber} received
from ]: {fromAircraftType} {from}: {message}");
}
}
public void RequestDoubleDeckerParking()
{
Transmit($"Requesting double-decker parking gate");
```

```
tower.DirectMessage($"Proceed to gate A1 - double-decker parking
available", "TOWER", flightNumber, "TOWER");
}
}
class Boeing787 : Aircraft
{
private bool ;
public Boeing787(AirportTower tower, string flightNumber, bool
isLongRange)
: base(tower, flightNumber, "Boeing 787")
{
this.isLongRange = isLongRange;
}
public override void Receive(string message, string from, string
fromAircraftType)
{
// Boeing prioritizes different types of messages
if (message.Contains("emergency") || message.Contains("priority"))
{
Console.WriteLine($"[{aircraftType} {flightNumber} - received
PRIORITY MESSAGE] from: {fromAircraftType} {from}:
{message}");
}
else if (!from.Equals(flightNumber))
{
Console.WriteLine($"[{aircraftType} {flightNumber} received
from ]: {fromAircraftType} {from}: {message}");
}
}
public void RequestFuelTruck(int fuelAmount)
{
Transmit($"Requesting fuel truck - need {fuelAmount} liters");
tower.DirectMessage($"Fuel truck dispatched to your position",
"TOWER", flightNumber, "TOWER");
}
}
class Program
{
static void Main(string[] )
{
Console.WriteLine("=== AIRPORT TOWER COMMUNICATION SIMULATION ===
\n");
AirportTower tower = new AirportTower();
AirbusA380 Turkih = new AirbusA380(tower, "TK202", 517);
Boeing787 AirAlgerie = new Boeing787(tower, "AH101", true);
Boeing787 QuatarArwais = new Boeing787(tower, "QR789", false);
Console.WriteLine("\n--- Morning Operations ---");
Turkih.RequestLanding();
QuatarArwais.RequestTakeoff();
Console.WriteLine("\n--- Emergency Scenario ---");
AirAlgerie.Transmit("MEDICAL EMERGENCY ON BOARD - requesting
priority handling");
// Show flight log
tower.ShowFlightLog();
Console.ReadKey();}}
```