Section: Master 1 SIGL

Faculty of Mathematics and Computer Science
Department of Computer Science

Object Analysis and Design (OAD)
Lab Session (TP) 05

Section 1. Mediator Pattern

We want to implement a communication system, between the chairs of sections, and chairs of courses, in which the *Department* plays the role of *Mediator*, who is responsible for exchanging messages. To build this system, we suppose the following: We have the class *Teacher* that has as information (Name, Id and Course), and *TeacherChair* that extends *Teacher*, and he is the only one allowed to exchange messages through the *Department* (Mediator). We have also the class *Student* that has as information (Name, InscNumber and Section), and *StudentChair* that extends *Student*, and he is the only one allowed to exchange messages through the *Department*. In order to send and receive messages, both *TeacherChair* and *StudentChair* implement two essential methods: *Send* and *Receive*, which are defined on the level of an interface called *IColleague*.

1. Given the condition that every *StudentChair* receives only from Teachers (*TeacherChair*), and *TeacherChair* receives only from students (*StudentChair*), Implement the previous scenario that gives the output presented below using the **Mediator pattern**.

```
Send (From Teacher Debbi): A seesion of ACO is palanned on Tuesday at 08:00,
please ack SIGL students
Elbar received from Teacher Debbi: A seesion of ACO is palanned on Tuesday at 08:00,
please ack SIGL students

Send (From Teacher Amroaui): A seesion of Algo is palanned on Monday at 09:30,
please ack SIGL students

Elbar received from Teacher Amroaui: A seesion of Algo is palanned on Monday at 09:30,
please ack SIGL students


Send (From the Student Elbar): Well received,
All SIGL students are acknowledged
Debbi received from Student Elbar: Well received,
All SIGL students are acknowledged
Amroaui received from Student Elbar: Well received,
All SIGL students are acknowledged
```

Figure 1: Mediator Pattern output

Answer Key for Exam A

Section 1. Mediator Pattern

We want to implement a communication system, between the chairs of sections, and chairs of courses, in which the *Department* plays the role of *Mediator*, who is responsible for exchanging messages. To build this system, we suppose the following: We have the class *Teacher* that has as information (Name, Id and Course), and *TeacherChair* that extends *Teacher*, and he is the only one allowed to exchange messages through the *Department* (Mediator). We have also the class *Student* that has as information (Name, InscNumber and Section), and *StudentChair* that extends *Student*, and he is the only one allowed to exchange messages through the *Department*. In order to send and receive messages, both *TeacherChair* and *StudentChair* implement two essential methods: *Send* and *Receive*, which are defined on the level of an interface called *IColleague*.

1. Given the condition that every *StudentChair* receives only from Teachers (*TeacherChair*), and *TeacherChair* receives only from students (*StudentChair*), Implement the previous scenario that gives the output presented below using the **Mediator pattern**.

```
Send (From Teacher Debbi): A seesion of ACO is palanned on Tuesday at 08:00,
please ack SIGL students
Elbar received from Teacher Debbi: A seesion of ACO is palanned on Tuesday at 08:00,
please ack SIGL students

Send (From Teacher Amroaui): A seesion of Algo is palanned on Monday at 09:30,
please ack SIGL students

Elbar received from Teacher Amroaui: A seesion of Algo is palanned on Monday at 09:30,
please ack SIGL students


Send (From the Student Elbar): Well received,
All SIGL students are acknowledged
Debbi received from Student Elbar: Well received,
All SIGL students are acknowledged
Amroaui received from Student Elbar: Well received,
All SIGL students are acknowledged
```

Figure 2: Mediator Pattern output

Answer:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Mediatore_Exam
{
    class Program
    {

        class Department
        {

            public delegate void Callback(string message, string from, string
              Role);
            Callback respond;

            public void SignOn(Callback method)
            {
                respond += method;
            }
            // Send is implemented as a broadcast
            public void Send(string message, string from, string Role)
            {
                respond(message, from, Role);
                Console.WriteLine();
            }
        }

        public interface Icolleague
        {
            void Send(string message);
            void Receive(string message, string from, string Role);
        }

        public class Student
        {
            protected string Name;
            protected string InscNumber;
            protected string Section;
        }

        class StudentChair: Student, Icolleague
        {
            Department mediator;
            string Role = "Student";
            public StudentChair(Department mediator, string name, string
              inscNumber, string section)
```

```
48              {
49                  this.mediator = mediator;
50                  mediator.SignOn(Receive);
51                  this.Name = name;
52                  this.InscNumber = inscNumber;
53                  this.Section = section;
54              }
55
56              public void Receive(string message, string from, string role)
57              {
58                  if (Role!=role)
59                      Console.WriteLine(Name + " received from "   + role + " " + ↵
                        from + ": " + message);
60              }
61
62              public void Send(string message)
63              {
64                  Console.WriteLine("Send (From " +  "the " + Role + " " +  Name  ↵
                      + "): " + message);
65                  mediator.Send(message, Name, Role);
66              }
67          }
68
69          public class Teacher
70          {
71              protected string Name;
72              protected string Id;
73              public string Course;
74          }
75
76          class TeacherChair : Teacher, Icolleague
77          {
78              Department mediator;
79              string Role = "Teacher";
80              public TeacherChair(Department mediator, string name, string id,    ↵
                  string course)
81              {
82                  this.mediator = mediator;
83                  mediator.SignOn(Receive);
84                  this.Name = name;
85                  this.Id = id;
86                  this.Course = course;
87              }
88
89              public void Receive(string message, string from, string role)
90              {
91                  if (Role != role)
92                      Console.WriteLine(Name + " received from "   + role + " " + ↵
                        from + ": " + message);
```

```
93              }
94
95              public void Send(string message)
96              {
97                  Console.WriteLine("Send (From "   + Role +   " " + Name + "): " + message);
98                  mediator.Send(message, Name, Role);
99              }
100         }
101
102         static void Main()
103         {
104             Department m = new Department();
105             // Two from head office and one from a branch office
106             StudentChair stdnt = new StudentChair(m, "Elbar", "2022_1", "SIGL_1");
107             TeacherChair tchr1 = new TeacherChair(m, "Debbi", "0000", "ACO");
108             TeacherChair tchr2 = new TeacherChair(m, "Amroaui", "1111", "Algo");
109
110             tchr1.Send("A seesion of " + tchr1.Course + " is palanned on Tuesday at 08:00, " +
111                 "\nplease ack SIGL students");
112             tchr2.Send("A seesion of " + tchr2.Course + " is palanned on Monday at 09:30, " +
113             "\nplease ack SIGL students\n");
114             stdnt.Send("Well received,\nAll SIGL students are acknowledged");
115             Console.ReadKey();
116         }
117     }
118 }
119
```