| Ex. No. 3 | Delegates and Events | | |
|---|---|---|---|
| Date of Exercise | 03.08.2016 | Date of Upload | 19.10.2016 |

## Aim

To develop **Subject Registration System** using C# by including the concept of Delegates and Events in order to perform various functions such as  Subject selection, staff selection and time table generation.

## Description

**Delegates** are nothing but the function pointers so as to:

- to pass methods around to other methods
- a delegate contains the address of a method

There **are 4 methods** associated with it:

1. **Delegate declaration**

- Delegate derived from System.Delegate

2. **Delegate methods definition**

- Any function whose signature matches the delegate signature

3. **Delegate instance creation**

- Hold reference to delegate method

4. **Delegate invocation**

- Invoke the method indirectly

**Declaring and using delegates:**

**Syntax**

```
[access specifier] delegate returntype delegatename(paramaters);
```

**Example**

```
delegate void SimpleDelegate();

public delegate void MathOperation(int x, inty);
```

**Multicast delegates:**

- Each method wrap just one single method call
- To call more methods, create more delegates explicitly
- It is possible for a delegate to wrap more than one method: multicast delegate
- Calling multicast delegate call successive methods wrapped on it
- Delegate signature is void or only get result of last method invoked
- A multicast delegate is a class derived from System.MulticastDelegate , which in turn is derived from System.Delegate

**Example**

```
class MathOperations
{
        public static void MultiplyByTwo(double value)
        {
        double result = value * 2;
        Console.WriteLine(“Multiplying by 2: {0} gives {1}”, value, result);
        }
        public static void Square(double value)
        {
        double result = value * value;
        Console.WriteLine(“Squaring: {0} gives {1}”, value, result);
        }
}
delegate void DoubleOp(double value);
class MainEntryPoint
{
static void Main()
{
DoubleOp operations = MathOperations.MultiplyByTwo;
operations += MathOperations.Square;
DoubleOp operation1 = MathOperations.MultiplyByTwo;
DoubleOp operation2 = MathOperations.Square;
DoubleOp operations = operation1 + operation2; //another way
operations(2.0); //inturn call MultiplyByTwo then Square method
operations(7.94);
operations(1.414);
}
}
```

**Array of delegates:**

The Delegate class defines the method **GetInvocationList()** that returns an array of Delegate objects. Using this we can invoke the methods associated with them directly, catch exceptions, and continue with the next iteration.

**Example**

```
static void Main()
{
DemoDelegate d1 = Program.One;
d1 += Two;
Delegate[] delegates = d1.GetInvocationList();
foreach(DemoDelegate d in delegates){
try{d();}catch (Exception){Console.WriteLine("Error in one");}
}
}
```

**Events** are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. For example, interrupts. Events are used for inter-process communication.
**Steps in event:**

- Event handler method (delegate method definition)
- Delegate declaration
- Event declaration
- Event instance creation (event binding)
- Event invocation

Event is a delegate type class member that is used by an object or a class to provide a notification to other objects that an event has occurred

```
[access modifier] event delegatetype event-name;
```

Define a method to handle the event and bind this to the event using += operator

```
event-name +=new delegatetype(method-name);
```

To remove a source of events, use -= operator

```
event-name -=new delegatetype(method-name);
```

Event invocation

```
event-name();
```

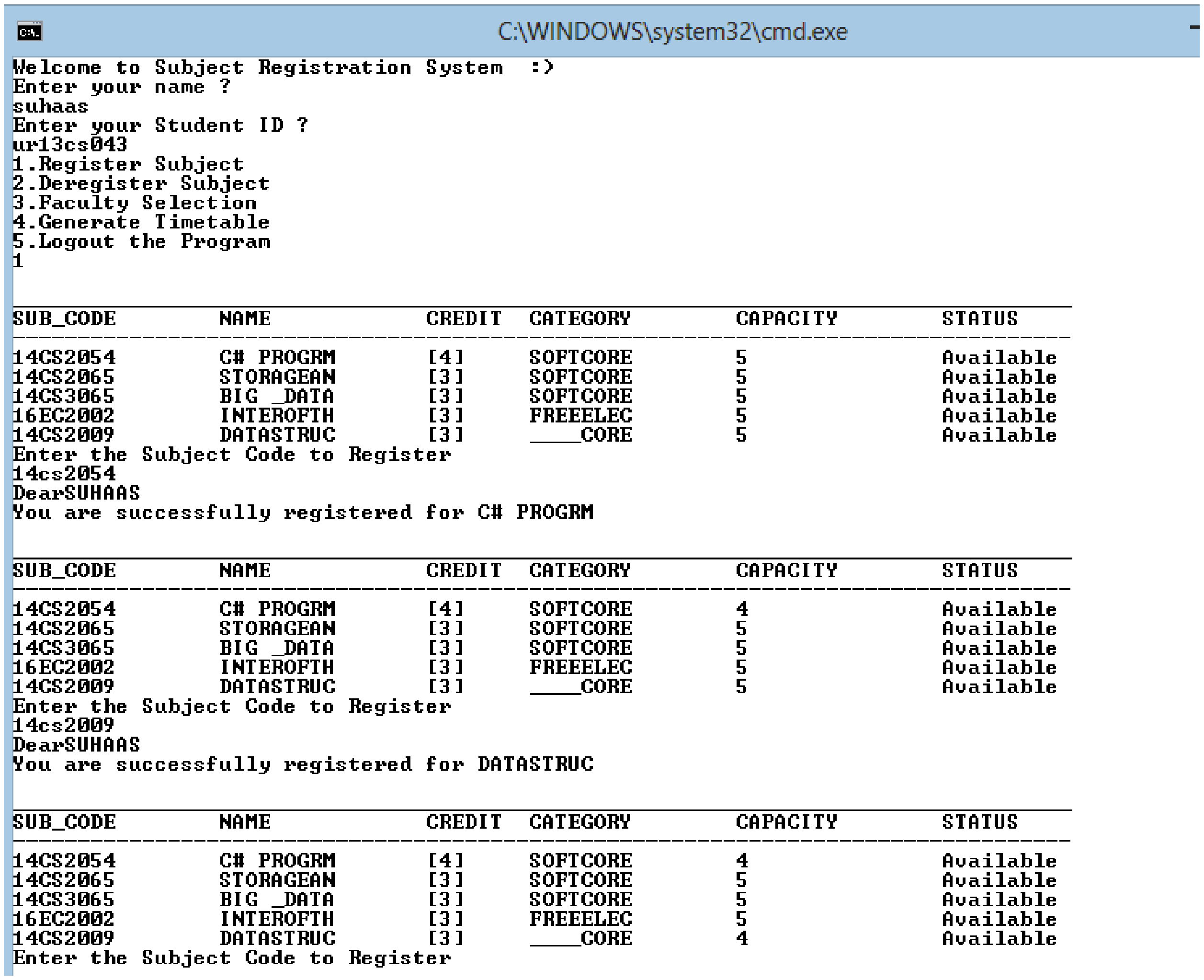
## Program

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

namespace SubjectRegistrationSystem
{

    class SubjectRegistration {
        public string SubjectCode;
        public int SubjectCapacity;
        public string SubjectCategory;
        public int SubjectCredit;
        public string SubjectName;
        public string SubjectStatus;
        public string GroupCode;
    }
    class UniversityStudents{
        public string StudentName;
        public string StudentId;
    }
    class RegisteredSubject {
        public string SubjectCode;
        public string StudentId;
    }

    //delegate for initialisation all meta lists
    delegate void delegatemethodforinit(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects);

    //delegate for function having subject list
    delegate void delegatemethodforsubjects(List<SubjectRegistration> subjects);


    //delegate for function having student list
    delegate int delegatemethodforstudents(List<UniversityStudents> student);
    //Event-Delegate declaration
    delegate int ValueChangedEventHandler(List<UniversityStudents> student);//delegate
declaration

    class Program
    {
        public event ValueChangedEventHandler Changed;//event declaration

            public  void Handle() {
            Console.WriteLine("");
        }

        static void Main(string[] args)
        {

            int index;
            List<SubjectRegistration> subjects = new List<SubjectRegistration>();
            List<UniversityStudents> student = new List<UniversityStudents>();
            List<RegisteredSubject> regsubjects = new List<RegisteredSubject>();

            Program pro = new Program();
            //defining method to handle the event
            pro.Changed += pro.initialisestudent;
            //Event Invocation
            index = pro.Changed(student);

            delegatemethodforinit menu = pro.initialisemenu;

            delegatemethodforinit regsub = pro.registersubject;

            delegatemethodforinit deregsub = pro.deregistersubject;

            delegatemethodforinit gentime = pro.generatetimetable;


            delegatemethodforsubjects initsubject = pro.initialisevalue;

            initsubject(subjects);

            menu(subjects, index, student, regsubjects);

        }

        public  void initialisemenu(List<SubjectRegistration> subjects, int studentindex,
List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {
            Console.WriteLine("1.Register Subject");
            Console.WriteLine("2.Deregister Subject");
            Console.WriteLine("3.Faculty Selection");
            Console.WriteLine("4.Generate Timetable");
            Console.WriteLine("5.Logout the Program");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice) {
                case 1:
                    registersubject(subjects, studentindex,student,regsubjects);
                    break;
                case 2:
                    deregistersubject(subjects, studentindex, student, regsubjects);
                    break;
                case 3:
                    break;
                case 4:
                    generatetimetable(subjects, studentindex, student, regsubjects);

                    break;
                case 5:
                    studentindex= initialisestudent(student);
                    initialisemenu(subjects, studentindex, student,regsubjects);
                    break;
                default:
                    Console.WriteLine("Invalid Choice");
                    break;
            }

        }

        public  void registersubject(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student,List<RegisteredSubject> regsubjects) {
            displaysubject(subjects);
            Console.WriteLine("Enter the Subject Code to Register");
            string Subcode = Console.ReadLine();
            if (Subcode.Equals("N")) { initialisemenu(subjects, studentindex, student,
regsubjects); };
            Subcode = Subcode.ToUpper();

            string sid = student[studentindex].StudentId;
            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode==Subcode &&
s.StudentId==sid);
            if (checkval == -1) {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].StudentName);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity--;
                        Console.WriteLine("You are successfully registered for " +
subjects[index].SubjectName);
                        regsubjects.Add(new RegisteredSubject() { SubjectCode = Subcode,
StudentId = sid });
                        registersubject(subjects, studentindex, student, regsubjects);

                    }
                    else
                    {
                        Console.WriteLine("No seats are further available :( ");
                        Console.WriteLine("Please try again :) ");
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subcode is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Already registered");
                Console.WriteLine("Please try again :) ");
                initialisemenu(subjects, studentindex, student, regsubjects);
            }



        }

        public  void deregistersubject(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {

            string sid = student[studentindex].StudentId;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("You have registered for the following subjects");

            foreach (RegisteredSubject subval in subs)
            {
                String subcodeval = subval.SubjectCode;
                List<SubjectRegistration> subject1 =
subjects.FindAll(a=>a.SubjectCode==subcodeval);
                displaysubject(subject1);
            }

            Console.WriteLine("Enter the Subject Code to Deregister");
            string Subcode = Console.ReadLine();
            Subcode = Subcode.ToUpper();

            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval != -1)
            {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].StudentName);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity++;
                        Console.WriteLine("You are successfully \"DEREGISTERED\" for " +
subjects[index].SubjectName);
                        regsubjects.RemoveAll(a => a.SubjectCode == Subcode &&
a.StudentId == sid);
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subject code is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Not Found"); }

        }

        public  void generatetimetable(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {

            string sid = student[studentindex].StudentId;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("__________________Time Table_________________________");

Console.WriteLine("Hour|||\tMonday|||\tTuesday|||\tWednesday|||\tThursday|||\tFriday|||")
;
            foreach (RegisteredSubject subval in subs) {
                String subcodeval = subval.SubjectCode;
                int indexvalof = subjects.FindIndex(a => a.SubjectCode == subcodeval);
                String groupval=subjects[indexvalof].GroupCode;
                //Console.WriteLine("Group:" + groupval);
                if (groupval.Equals("A"))
                {
                    Console.WriteLine();
                    Console.Write("1\t");
                    for (int i = 0; i < 4; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }
                    Console.WriteLine();
                }
                else if (groupval.Equals("B")) {
                    Console.WriteLine();
                    Console.Write("2\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else if (groupval.Equals("C"))
                {
                    Console.WriteLine();
                    Console.Write("3\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else if (groupval.Equals("D"))
                {
                    Console.WriteLine();
                    Console.Write("4\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else if (groupval.Equals("E"))
                {
                    Console.WriteLine();
                    Console.Write("5\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else
                {
                    Console.WriteLine("_____________You are free___________");
                }

            }
            initialisemenu(subjects, studentindex, student, regsubjects);
        }


        public  int initialisestudent(List<UniversityStudents> student) {
            Console.WriteLine("Welcome to Subject Registration System  :) ");
            Console.WriteLine("Enter your name ?");
            string Name = Console.ReadLine().ToUpper();
            Console.WriteLine("Enter your Student ID ?");
            string Id = Console.ReadLine().ToUpper();
            int index = -1;
            index = student.FindIndex(a => a.StudentId == Id);
            if (index < 0)
            {
                student.Add(new UniversityStudents() { StudentName = Name, StudentId = Id
});
            }
            index = student.FindIndex(a => a.StudentId == Id);
            return index;

        }

        public  void initialisevalue(List<SubjectRegistration> subjects)
        {
            subjects.Add(new SubjectRegistration() { SubjectName = "C# PROGRM",
SubjectCode = "14CS2054", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 4, SubjectStatus = "Available", GroupCode = "A"});
```

```
            subjects.Add(new SubjectRegistration() { SubjectName = "STORAGEAN",
SubjectCode = "14CS2065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "B"});
            subjects.Add(new SubjectRegistration() { SubjectName = "BIG _DATA",
SubjectCode = "14CS3065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "C"});
            subjects.Add(new SubjectRegistration() { SubjectName = "INTEROFTH",
SubjectCode = "16EC2002", SubjectCategory = "FREEELEC", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "D"});
            subjects.Add(new SubjectRegistration() { SubjectName = "DATASTRUC",
SubjectCode = "14CS2009", SubjectCategory = "____CORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "E"});
        }

        public  void displaysubject(List<SubjectRegistration> subjects) {

Console.WriteLine("\n____________________________________________________________________
______________");
            Console.WriteLine("SUB_CODE\tNAME\t\tCREDIT\tCATEGORY\tCAPACITY\tSTATUS");
            Console.WriteLine("-------------------------------------------------------------
------------------------");
            foreach (SubjectRegistration subval in subjects)
            {
                Console.WriteLine("{0}\t{1}  \t[{2}]\t{3}\t{4}\t\t{5}",
subval.SubjectCode, subval.SubjectName, subval.SubjectCredit, subval.SubjectCategory,
subval.SubjectCapacity, subval.SubjectStatus);

            }
        }


    }
}
```

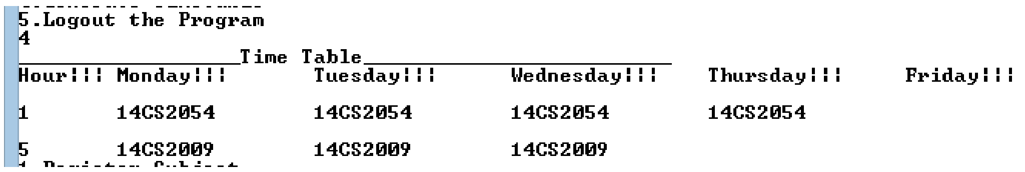
## Output

- **Register Subject**



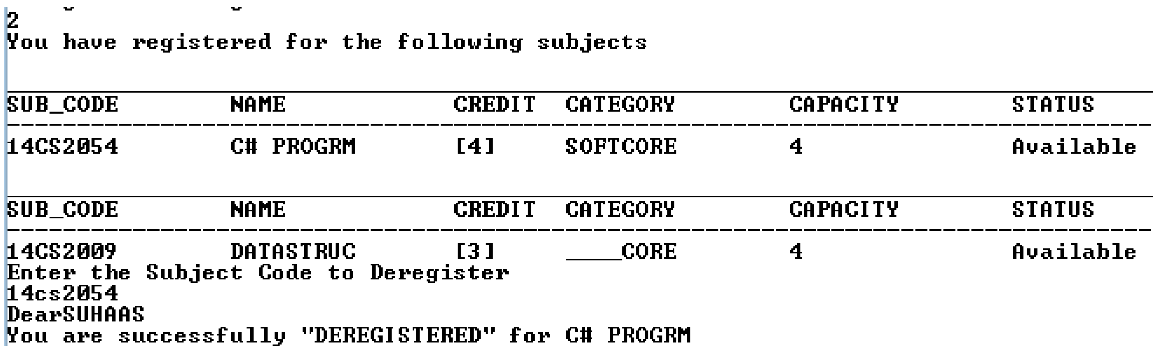
```
Welcome to Subject Registration System  :)
Enter your name ?
suhaas
Enter your Student ID ?
ur13cs043
1.Register Subject
2.Deregister Subject
3.Faculty Selection
4.Generate Timetable
5.Logout the Program
1

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        5               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        5               Available
Enter the Subject Code to Register
14cs2054
DearSUHAAS
You are successfully registered for C# PROGRM

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        5               Available
Enter the Subject Code to Register
14cs2009
DearSUHAAS
You are successfully registered for DATASTRUC

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        4               Available
Enter the Subject Code to Register
```

- **Timetable Generation**



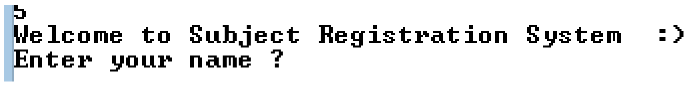
```
5.Logout the Program
4
________________Time Table_______________________
Hour!!! Monday!!!       Tuesday!!!      Wednesday!!!    Thursday!!!     Friday!!!

1       14CS2054        14CS2054        14CS2054        14CS2054

5       14CS2009        14CS2009        14CS2009
```

- **Deregister Subjects**



```
2
You have registered for the following subjects

SUB_CODE         NAME             CREDIT   CATEGORY         CAPACITY         STATUS
--------------------------------------------------------------------------------------
14CS2054         C# PROGRM        [4]      SOFTCORE         4                Available

SUB_CODE         NAME             CREDIT   CATEGORY         CAPACITY         STATUS
--------------------------------------------------------------------------------------
14CS2009         DATASTRUC        [3]      ____CORE         4                Available
Enter the Subject Code to Deregister
14cs2054
DearSUHAAS
You are successfully "DEREGISTERED" for C# PROGRM
```

- **Log Out**



```
5
Welcome to Subject Registration System  :)
Enter your name ?
```

## Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]