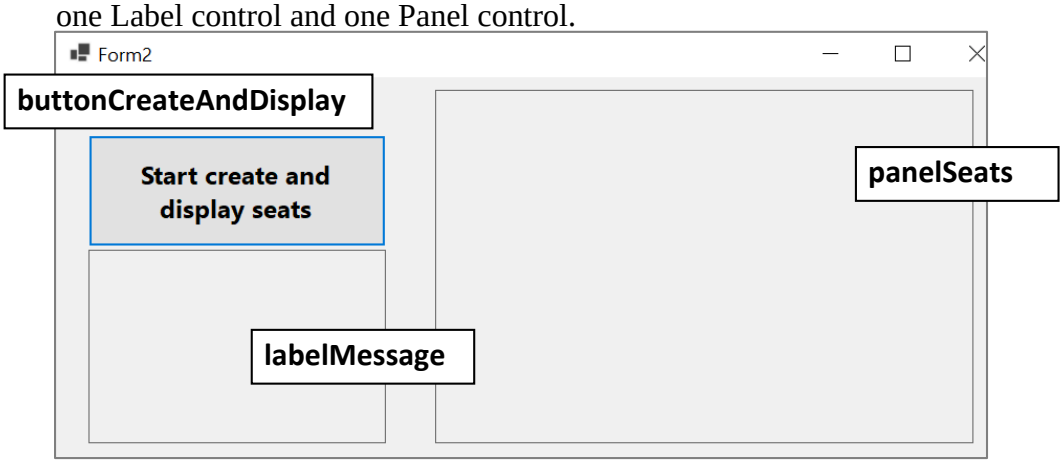
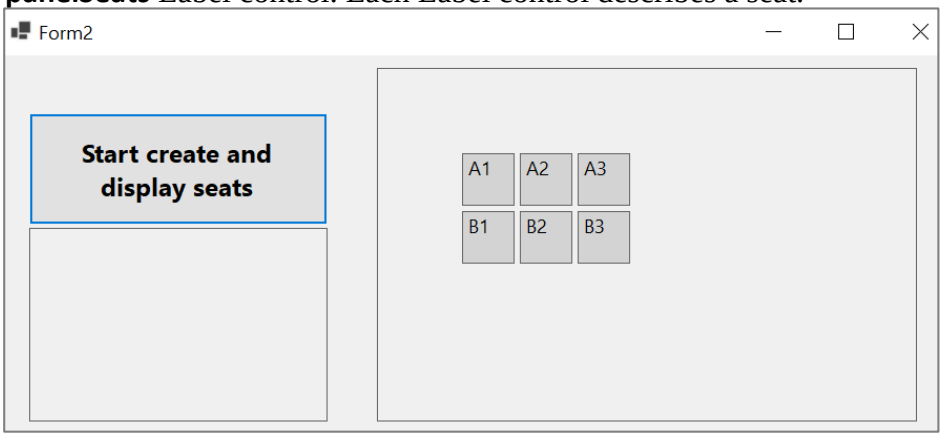
Assignment Preparation Exercise 1 Objective

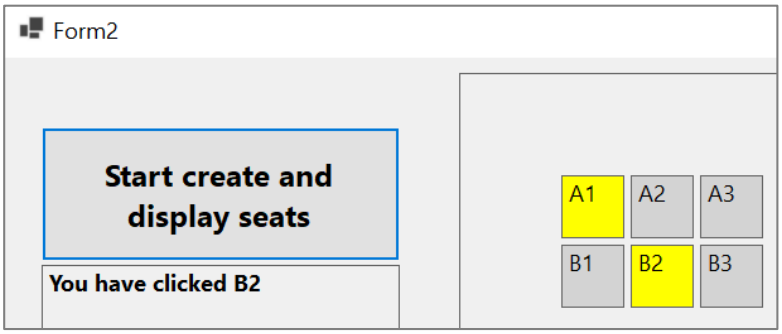
When the program starts, a form interface appears. The form interface displays one Button control, one Label control and one Panel control.



When the user clicks the button, the program dynamically creates 6 Label controls inside the **panelSeats** Label control. Each Label control describes a seat.

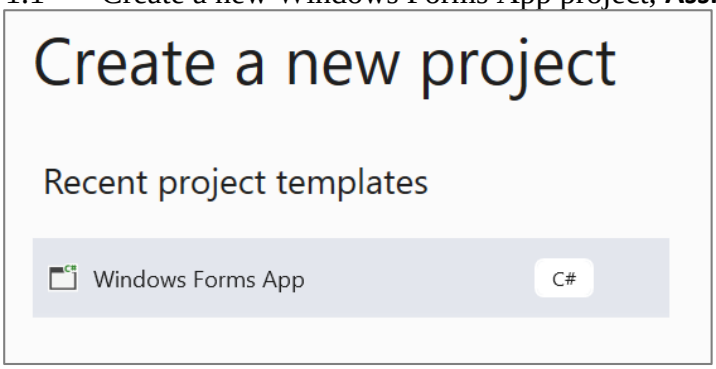


When the user clicks on the *same* seat several times, the program toggles the seat selection by highlighting a booked seat with yellow background or highlighting an un-booked seat with light gray background.

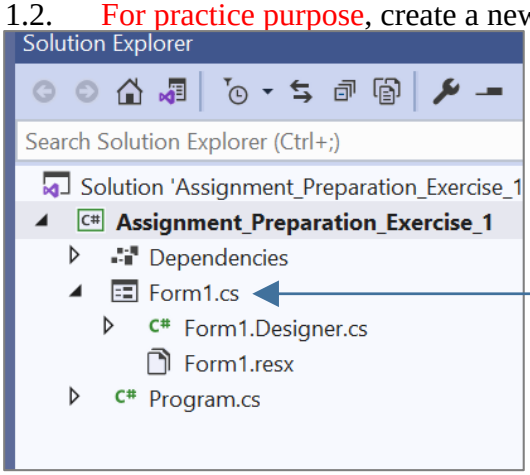


Prepare the Graphical User Interface layer (Form2)

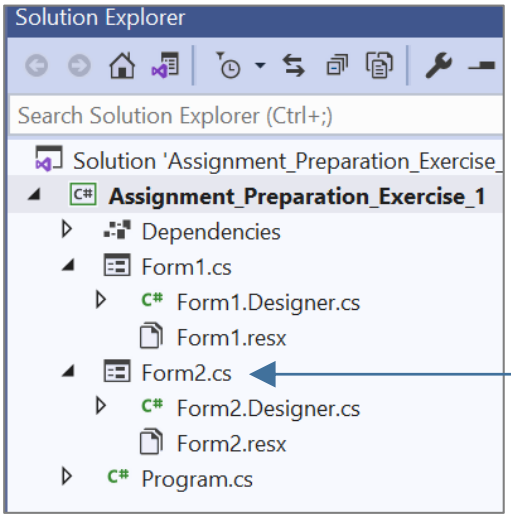
1.1 Create a new Windows Forms App project, **Assignment_Preparation_Exercise_1**.



1.2. For practice purpose, create a new **Form2** class inside the project.



Usually, a default **Form1** form class is provided after the project is created.
Create a new **Form2** class inside the project. You will provide code and some controls inside Form2.

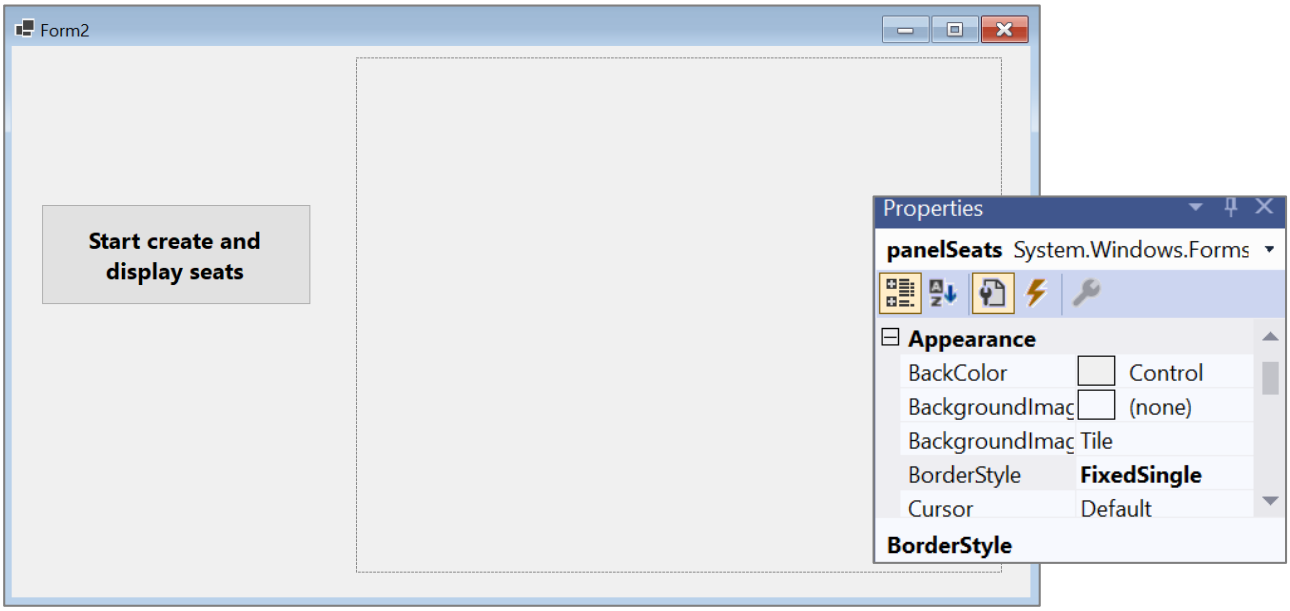


1.3. Setup the project, so that the Form2 will be displayed when you execute the project. Modify the code inside the **Program.cs** file.

```
11      /// <summary>
12      ///  The main entry point for the application.
13      /// </summary>
14      [STAThread]
15      static void Main()
16      {
17          Application.SetHighDpiMode(HighDpiMode.SystemAware);
18          Application.EnableVisualStyles();
19          Application.SetCompatibleTextRenderingDefault(false);
20          Application.Run(new Form2());
21      }
22  }
```

Modify the statement at line 20.

1.4. Design the Form2 by adding one **Button** control (**buttonCreateAndDisplay**) and one **Panel** control (**panelSeats**). Set the Button control to have the name, **buttonCreateAndDisplay**. Name the Panel control as **panelSeats**. Use the Properties interface to setup the **panelSeats** control's border.



1.5. Double click on the **buttonCreateAndDisplay** Button control to:

(a) View the Form2's code

(b) Automatically generate an event handler method code which handles the Button control's click event.

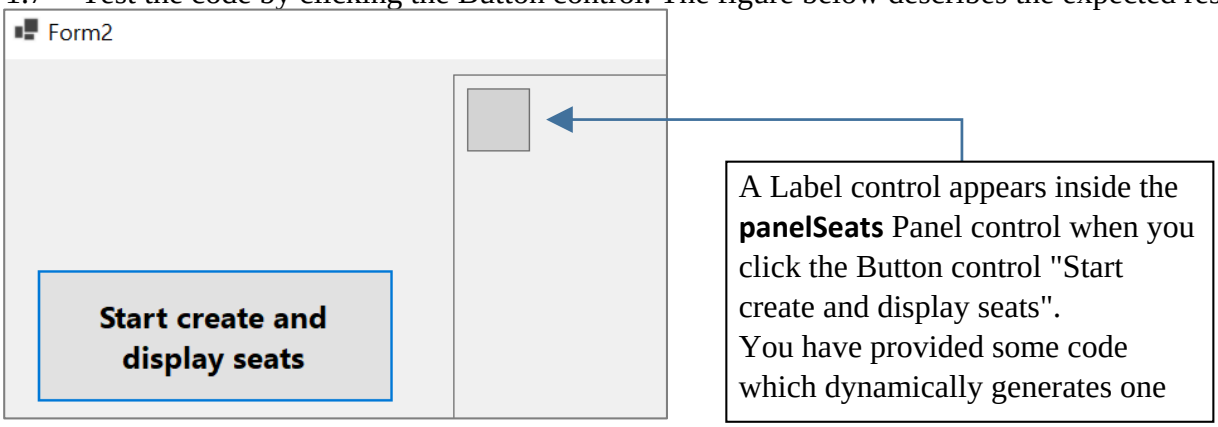
1.6 Refer to the following code.

Prepare the code within the body of the **buttonStartCreateAndDisplay_Click** event handler method.

Explanation: When the Button control is clicked, the code inside this event handler method will execute to dynamically create one Label control and insert the Label control inside the Panel control. The Panel control will be the "container".

```
18  private void buttonStartCreateAndDisplay_Click(object sender, EventArgs e)
19  {
20      //Create two variables which can be used to control a nested looping logic later.
21      int maxRow = 2;
22      int maxColumn = 3;
23      int seatWidth = 50;
24      int seatHeight = 50;
25      Label labelSeat = new Label();
26      labelSeat.Top = 10;
27      labelSeat.Left = 10;
28      labelSeat.Width = seatWidth;
29      labelSeat.Height = seatHeight;
30      labelSeat.BorderStyle = BorderStyle.FixedSingle;
31      labelSeat.BackColor = Color.LightGray;
32      //Insert the new Label object inside the Panel control, panelSeats.
33      panelSeats.Controls.Add(labelSeat);
34  }// End of buttonStartCreateAndDisplay_Click method
```

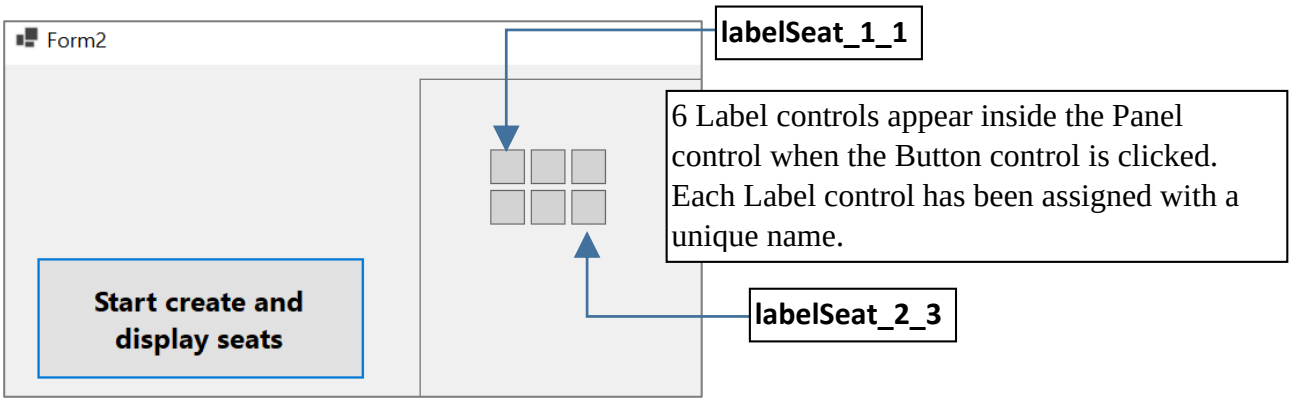
1.7 Test the code by clicking the Button control. The figure below describes the expected result.



1.8 Refer to the figure below. Modify the code so that:

(a) Two rows of seats can be created. Each row has three seats.

(b) Each Label control requires a unique name.



1.9 The logic which dynamically generates a Label and insert it into the Panel control is repetitive. Therefore, you modify the existing code by applying a looping control structure. Notice that there are 2 rows and 3 columns. Therefore, you need a nested looping control structure.

```
18 private void buttonStartCreateAndDisplay_Click(object sender, EventArgs e)
19 {
20   //Create two variables which can be used to control a nested looping logic later.
21   int maxRow = 2;
22   int maxColumn = 3;
23   int seatWidth = 50;
24   int seatHeight = 50;
25   int x, y = 0;
26       for (x = 1; x <= maxColumn; x++) {
27         for (y = 1; y <= maxRow; y++)
28         {
29           //The following code has been identified as repetitive
30           Label labelSeat = new Label();
31           labelSeat.Top = 10 + (y * (seatHeight + 5)) + 15; //Note that 5 is just to space out the boxes
32           labelSeat.Left = 10 + (x * (seatWidth + 5)) + 15;
33           labelSeat.Width = seatWidth;
34           labelSeat.Height = seatHeight;
35           labelSeat.BorderStyle = BorderStyle.FixedSingle;
36           labelSeat.BackColor = Color.LightGray;
37           labelSeat.Name = "labelSeat_" + y.ToString() + "_" + x.ToString(); //labelSeat_1_1 etc.
38
39           //Insert the new Label object inside the Panel control, panelSeats.
40           panelSeats.Controls.Add(labelSeat);
41         }// End of for (y=1; ....
42       }// End of for (x = 1; ....)
43   }// End of buttonStartCreateAndDisplay_Click method
```

Additional **x** and **y** variables are declared. They are used as counting variables inside the for loop control structure.

Note that, the values (e.g. 10, 5 and 15) were derived through trial and error.

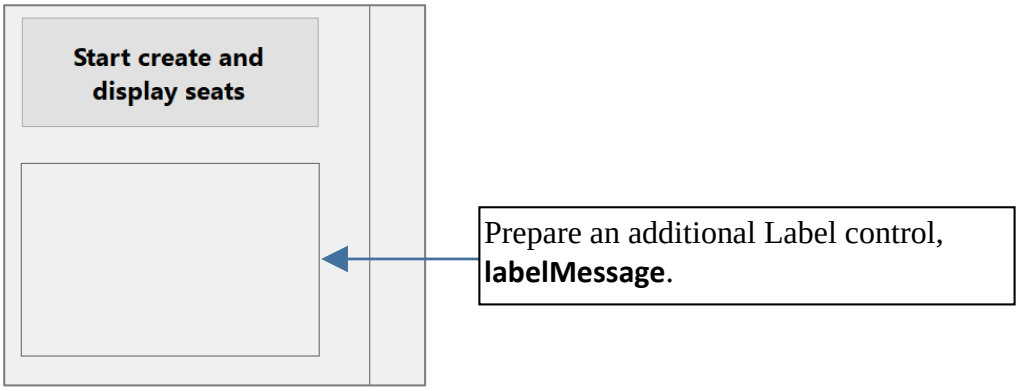
1.10 Refer to the article at https://www.codeproject.com/Questions/734230/How-Can-I-Write-Click-Event-For-A-Label-In-Csharp, a developer has shared how to define event handler to handle Label control's click event.
You have 6 Label controls. To make the program "remember" or "track" which Label control you have clicked, you need attach an event handler method to each dynamically generated Label control.

1.11 Add an additional Label control, **labelMessage** at Form2. Use the **Properties** interface to:

(a) Set the **AutoSize** property to false.

(b) Set the font to bold.

(c) Set the **BorderStyle** to FixedSingle.



1.12 Refer to the code below.

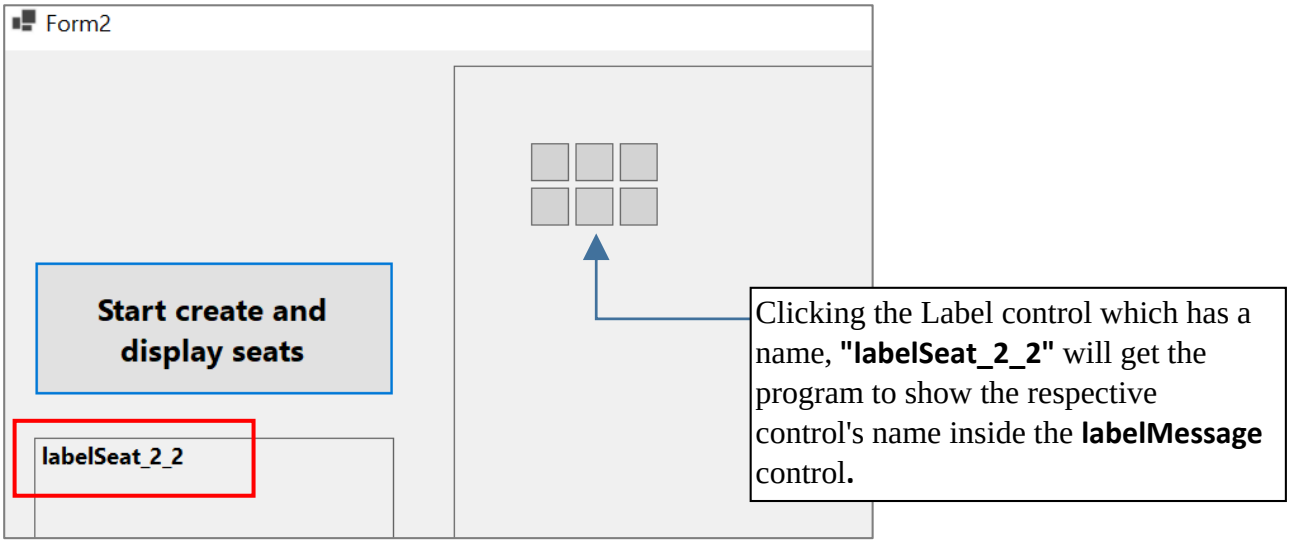
(a) Use the code from line 46 to line 55 to create a new method, **HandleLabelClick**. The **HandleLabelClick** requires two input parameters. The code was influenced by the article content at https://www.codeproject.com/Questions/734230/How-Can-I-Write-Click-Event-For-A-Label-In-Csharp.

(b) Use the code from line 38 to line 39 to attach the **HandleLabelClick** method to the **Click** property of the dynamically generated Label control.

```
36            labelSeat.BackColor = Color.LightGray;
37            labelSeat.Name = "labelSeat_" + y.ToString() + "_" + x.ToString();
38            //Attach a method to handle the Label control's Click event
39            labelSeat.Click += new EventHandler(HandleLabelClick);
40            //Insert the new Label object inside the Panel control, panelSeats.
41            panelSeats.Controls.Add(labelSeat);
42          }// End of for (y=1; ....
43        }// End of for (x = 1; ....)
44      }// End of buttonStartCreateAndDisplay_Click method
45
46      private void HandleLabelClick(object sender, EventArgs e)
47      {
48        string labelName = "";
49        // The sender references the object which raised the Click event signal.
50        // To make the Label datatype labelSeat variable reference the sender,
51        // a simple casting technique is required.
52        Label labelSeat = (Label)sender;
53        labelMessage.Text = labelSeat.Name; //Display the name of the control which is clicked.
54
55      }// End of HandleLabelClick
56
57
58
59    }// End of Form2 class
60  }// End of namespace
```

1.13 Test the project to ensure that the event handler method is working.

1.14 Refer to the figure below. The **labelMessage** control shows "**labelSeat_2_2**" because the Label control which has the name "**labelSeat_2_2**" is clicked.



Prepare Seat class, Node class, DoubleLinkedList class and SeatManager class to ①model seat and seat arrangement, ②structure the data using double linked list technique and ③apply simple algorithms to manage the data.

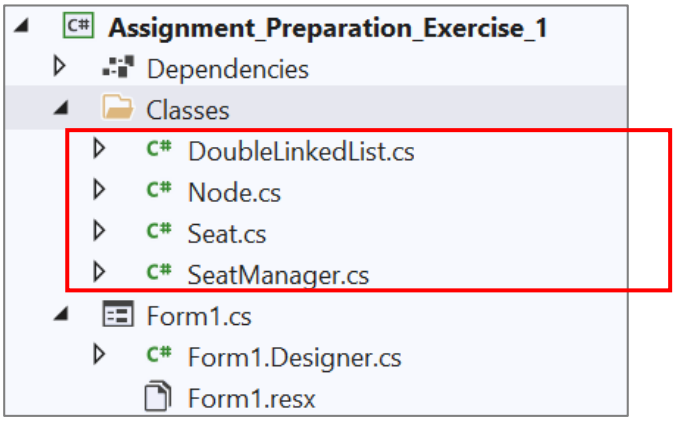
At this stage of the exercise, you have prepared a form interface which has the basic code capable of allowing the user to interact with the form by clicking on the dynamically generated Label controls.

You need to provide more code inside the appropriate classes to structure the data so that the program can **track and remember** the user actions.

2.1. Use the Solution Explorer interface to:

(a) Create a **Classes** folder.

(b) Create four class files inside the **Classes** folder. The class file names are **Node.cs**, **Seat.cs**, **DoubleLinkedList.cs** and **SeatManager.cs** **(SeatManager class is the business logic class)**.



2.2. Refer to the code listing below. Provide the necessary code for the **Seat** class. The **Seat** class is used inside this project to model one seat.

① A seat has a row value (row is equivalent to y inside the code). Therefore, a **Row** property is defined. The **_row** variable is used to hold the value.

② A seat has a seat number value (seat number is equivalent to column or x inside the code). Therefore, a **SeatNumber** property is defined. The **_seatNumber** variable is used to hold the seat number (column) information.

③ Each seat has several statuses. In this workbook exercise, the program only needs to track whether the seat is booked or not booked. Therefore, **BookStatus** property is defined and **_bookStatus** variable is used to hold the value.

The number of required properties inside the **Seat** class is influenced by the program's specification and objective.

Seat class (You need this class to model a piece of information inside the Node object)

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Assignment_Preparation_Exercise_1.Classes
{
   class Seat
   {
      private bool _bookStatus = false;
```

```
        private bool _canBook = false;
        // The _row field is 2 if the object is modelling  a seat at row B.
        private int _row;
        // _seatNumber field is  3 if the object is modelling a seat at column 3.
        private int _seatNumber;
        public int Row   // property
        {
            get { return _row; }  // get method
            set
            {
                _row = value;
            } // set method
        }
        public int SeatNumber   // property
        {
            get { return _seatNumber; }  // get method
            set { _seatNumber = value; } // set method
        }
```

```
        public bool CanBook   // property
        {
            get { return _canBook; }  // get method
            set
            {
                _canBook = value;
            } // set method
        }
        public bool BookStatus   // property
        {
            get { return _bookStatus; }  // get method
            set
            {
                _bookStatus = value;
            } // set method
        }
        public string ComputeSeatLabel()
        {
            return ((char)(_row + 64)).ToString() + _seatNumber.ToString();
        }
```

The **ComputeSeatLabel** method's code was developed through trial and error by researching on other developers' sharing at online articles. The logic converts a row value of 1 to "A', 2 to "B" etc. Then the alphabet is concatenated with the seat number.

```
    }// End of Seat class
}// End of namespace
```

2.3 Refer to the code listing below. Provide the necessary code for the **Node** class.

Node class (You need this class for the double linked list logic inside the **DoubleLinkedList** class)

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Assignment_Preparation_Exercise_1.Classes
{
   class Node
   {
```

```
   public Node prev;
   public Seat seat;
   public Node next;

   public Node(Seat pSeat)
   {
   seat = pSeat;
   prev = null;
   next = null;
   }
```

```
   }// End of Node class
}// End of namespace
```

2.4 Refer to the code listing below. Prepare the code for the **DoubleLinkedList** class

DoubleLinkedList class (You need this class for the data manipulation logic inside the **SeatManager** class)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Assignment_Preparation_Exercise_1.Classes
{
   class DoubleLinkedList
   {
```

```
      //"Always make sure" that this "start" refers to the first node of the list.
      public Node Start { get; set; }
      public DoubleLinkedList()
      {
         this.Start = null;
      }//End of constructor

      public void InsertAtEnd(Seat seatData)
      {
         Node newNode = new Node(seatData);
         if (this.Start == null)
         {
            this.Start = newNode;
            return;
```

```
        }
        Node p = this.Start;
        // Traverse through the list until the p refers to
        // the last node.
        while (p.next != null)
        {
           p = p.next;
        }// End of while
        p.next = newNode;
        newNode.prev = p;
      }//End of InsertAtEnd

      public Seat SearchByRowAndColumn(int pRow, int pColumn)
      {

        Node p = this.Start;
        while (p != null)
        {
          if ((p.seat.SeatNumber == pColumn) && (p.seat.Row == pRow))
          {
            //If the node referenced by p satisfies the
            //search criteria, exit the loop
            //The p will reference the node which satisfies
            //the search criteria before exiting the while loop.
            break;//Exit the while loop
          }
          p = p.next; //Continue to the next node
        }//While loop

        if (p == null)
        {
          return null;
        }
        else
        {
          return p.seat;
        }//End of if..else block
      }//End of SearchByRowAndColumn
    }// End of class
}// End of namespace
```

2.5 Refer to the following code listing. Prepare the code so that the logic inside the SeatManager class, can manage/handle the double-linked-list structured seat data.

SeatManager class

Explanation: You need this class for the *main interaction logic* inside the **Form2** class). The code inside the Form2 **should not directly use** the DoubleLinkedList type object.

```
using System;
using System.Collections.Generic;
using System.Text;
using Assignment_Preparation_Exercise_1.Classes;
namespace Assignment_Preparation_Exercise_1.Classes
{   // The SeatManager class has methods to manage the double linked list.
    // For example, InsertOneSeat, ResetAllSeatStatus,
    //FindOneSeatToBook, FindOneSeatToUnbook etc.
  class SeatManager
  {
     DoubleLinkedList _seats;
     public SeatManager()
     {
        _seats = new DoubleLinkedList();
     }
     public Seat InsertOneSeat(int row,int column)
     {
        Seat newSeat = new Seat();
        //Setup the Seat object
        newSeat.Row = row;
        newSeat.SeatNumber = column;
        newSeat.CanBook = true;
        //Insert the Seat object into the double linked list.
        _seats.InsertAtEnd(newSeat);
        return newSeat;
     } // End of InsertOneSeat
     public Seat FindOneSeatToBook (int row, int column)
     {
        Seat seat = _seats.SearchByRowAndColumn(row, column);
        seat.BookStatus = true;
        return seat;
     }// End of FindOneSeatToBook

     public Seat FindOneSeatToUnbook(int row, int column)
     {
        Seat seat = _seats.SearchByRowAndColumn(row, column);
        seat.BookStatus = false;
        return seat;
     }// End of FindOneSeatToUnbook

  }// End of SeatManager class
}// End of namespace
```

A private class scope DoubleLinkedList type variable, **seats** is declared here.

Observe that, the **InsertOneSeat**, **FindOneSeatToBook** and **FindOneSeatToUnbook** methods return a Seat type object to the calling program. Without having the Seat object returned to the code inside the Form2, the code inside the Form2 cannot make changes to the respective Label control's appearance to reflect the user actions.

Using the SeatManager class inside the Form2

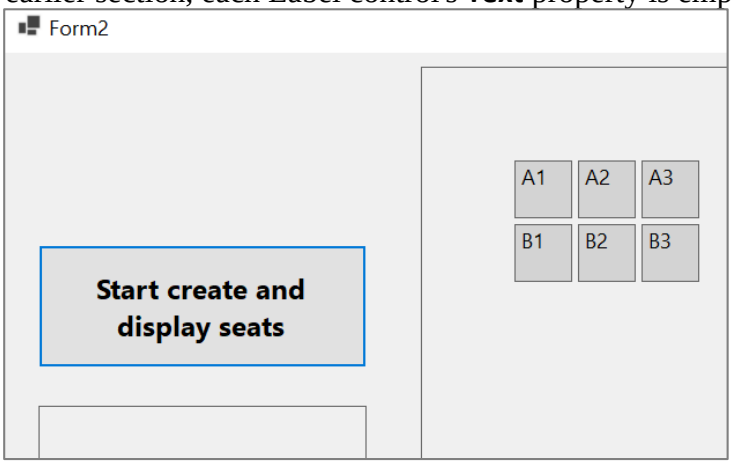
The previous section has prepared the classes starting from Seat class ➔ Node class ➔ DoubleLinkedList class ➔ SeatManager class.

Note: Due to author's constraint, the workbook exercise did not emphasize on testing the code which is done in the previous section. It is necessary to test the code before beginning this section.

3.1 Form2's code focus on handling the graphical user interface (what user should see, what user can do, what to show after user actions etc.). While the SeatManager class's code focus on structuring the data and data management. You need to declare one SeatManager type variable, **_seatManager** as seen at line 13 inside the code listing below.

```
4   using System.Data;
5   using System.Drawing;
6   using System.Text;
7   using System.Windows.Forms;
8   using Assignment_Preparation_Exercise_1.Classes;
9   namespace Assignment_Preparation_Exercise_1
10  {
11      public partial class Form2 : Form
12      {
13          SeatManager _seatManager = new SeatManager();
14          public Form2()
15          {
16              InitializeComponent();
17          }
18
19  private void buttonStartCreateAndDisplay_Click(object sender, EventArgs e)
20  {
21      //Create two variables which can be used to control a nested looping logic later.
22      int maxRow = 2;
```

Provide more code inside Form2 so that when the user clicks the "Start create and display seats" button, the program can display seat label information for each Label control. Recall that, in the earlier section, each Label control's **Text** property is empty string.



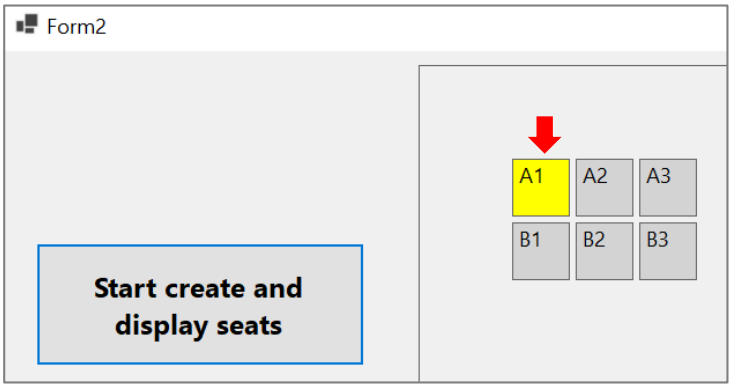
3.2 Call the **InsertOneSeat** method to tell the **_seatManager** object to *internally* create a new Seat object, managed the new object inside the double linked list and finally return the new Seat object back to the calling program.

```
38              labelSeat.Name = "labelSeat_" + y.ToString() + "_" + x.ToString();
39              //Attach a method to handle the Label control's Click event
40              labelSeat.Click += new EventHandler(HandleLabelClick);
41              //Use the seatManager object to create a new Seat object
42              Seat seat = _seatManager.InsertOneSeat(y, x);
43              //Set the Tag property of the new Label control to reference the new Seat object, seat.
44              labelSeat.Tag = seat;
45              labelSeat.Text = seat.ComputeSeatLabel();
46              //Insert the new Label object inside the Panel control, panelSeats.
47              panelSeats.Controls.Add(labelSeat);
48          }// End of for (y=1; ....
49      }// End of for (x = 1; ....)
50  }// End of buttonStartCreateAndDisplay_Click method
```

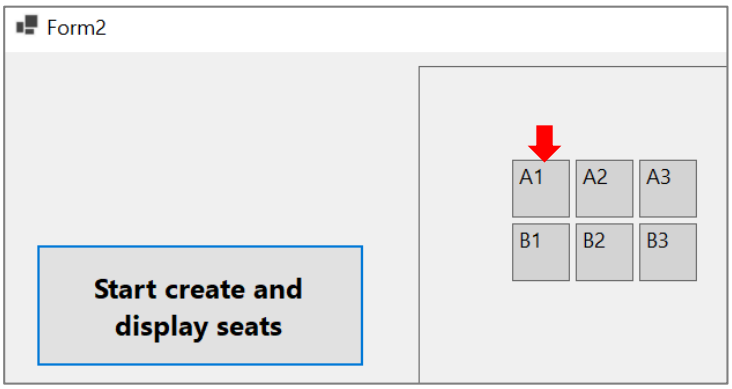
3.3 Set the new Label control's **Tag** property to reference the new Seat object. Set the **Text** property to display the seat label information to the user.

```
38            labelSeat.Name = "labelSeat_" + y.ToString() + "_" + x.ToString();
39            //Attach a method to handle the Label control's Click event
40            labelSeat.Click += new EventHandler(HandleLabelClick);
41            //Use the seatManager object to create a new Seat object
42            Seat seat = _seatManager.InsertOneSeat(y, x);
43            //Set the Tag property of the new Label control to reference the new Seat object, seat.
44            labelSeat.Tag = seat;
45            labelSeat.Text = seat.ComputeSeatLabel();
46            //Insert the new Label object inside the Panel control, panelSeats.
47            panelSeats.Controls.Add(labelSeat);
48          }// End of for (y=1; ....
49        }// End of for (x = 1; ....)
50    }// End of buttonStartCreateAndDisplay_Click method
```

3.4 Refer to the following figure. Refer to the next code listing (next page). Provide more code inside the **HandleLabelClick** method to manage the program behaviour and track the user selection.



Click the Label control which represents seat A1.
The code inside the event handler method will check the seat's *status*.
If the seat *has not been booked*, the respective Label control's background colour changes to yellow.



Click the Label control which represents seat A1 for the second time.
The code inside the event handler method will check the seat's *status*.
If the seat *has been booked*, the respective Label control's background colour changes from yellow to light grey.

```
52  private void HandleLabelClick(object sender, EventArgs e)
53  {  // The sender automatically references the object which raised the Click event signal.
54     // Refer to the statement below. To make the Label datatype labelSeat variable
55     // reference the sender,  a simple casting technique is required.
56     Label labelSeat = (Label)sender;
57     // Obtain the Seat object which is tagged to the Label which raised the Click event.
58     Seat seat = (Seat)labelSeat.Tag;
59     // Obtain the row and seat number info from the Seat object, so that can search for the seat
60     if (seat.BookStatus == false)
61     {
62        seat = _seatManager.FindOneSeatToBook(seat.Row, seat.SeatNumber);
63     }
64     else
65     {
66        seat = _seatManager.FindOneSeatToUnbook(seat.Row, seat.SeatNumber);
67     }
68     if (seat.BookStatus == false)
69     {
70        labelSeat.BackColor = Color.LightGray;
71     }
72     else
73     {
74        labelSeat.BackColor = Color.Yellow;
75     }
76     labelMessage.Text = String.Format("You have clicked {0}", labelSeat.Text);
77  }// End of HandleLabelClick
```

End

Sample code during the initial experimentation on create a dynamic control.

```
private void buttonStartCreateAndDisplay_Click(object sender, EventArgs e)
{
   //Create two variables which can be used to control a nested looping logic later.
   int maxRow = 2;
   int maxColumn = 3;
   int seatWidth = 30;
   int seatHeight = 30;
   Label labelSeat = new Label();
   labelSeat.Top = 10;
   labelSeat.Left = 10;
   labelSeat.Width = seatWidth;
   labelSeat.Height = seatHeight;
   labelSeat.BorderStyle = BorderStyle.FixedSingle;
   labelSeat.BackColor = Color.LightGray;
   //Insert the new Label object inside the Panel control, panelSeats.
   panelSeats.Controls.Add(labelSeat);
}// End of buttonStartCreateAndDisplay_Click method
```