

C#

Assignment - 1

1. Write a Simple console Application Calculator with the help of Visual Studio .NET IDE which will perform following operations on two numbers:

- Addition.
- Subtraction.
- Multiplication.
- Division

Accept input from the user and display results on the console. Make use of loops, switch case wherever required.

```
string operation = Console.ReadLine();
if (operation == "Addition")
{
    int n1 = Convert.ToInt32(Console.ReadLine());
    int n2 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("The sum of the numbers is "+Convert.ToString(n1+n2));
}
else if (operation == "Subtraction")
{
    int n1 = Convert.ToInt32(Console.ReadLine());
    int n2 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("The difference of two numbers is "+Convert.ToString(n1 - n2));
}
else if (operation == "Multiplication")
{
    int n1 = Convert.ToInt32(Console.ReadLine());
    int n2 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("The product of two numbers is "+Convert.ToString(n1 * n2));
}
else
```

```

int n1 = Convert.ToInt32(Console.ReadLine());
int n2 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("The division of the two numbers is "+Convert.ToString(n1 / n2));
}

```

```

1 string operation = Console.ReadLine();
2 if(operation == "Addition")
3 {
4     int n1 = Convert.ToInt32(Console.ReadLine());
5     int n2 = Convert.ToInt32(Console.ReadLine());
6     Console.WriteLine("The sum of the numbers is "+Convert.ToString(n1+n2));
7 }
8 else if (operation == "Subtraction")
9 {
10    int n1 = Convert.ToInt32(Console.ReadLine());
11    int n2 = Convert.ToInt32(Console.ReadLine());
12    Console.WriteLine("The difference of two numbers is "+Convert.ToString(n1 - n2));
13 }
14 else if (operation == "Multiplication")
15 {
16    int n1 = Convert.ToInt32(Console.ReadLine());
17    int n2 = Convert.ToInt32(Console.ReadLine());
18    Console.WriteLine("The product of two numbers is "+Convert.ToString(n1 * n2));
19 }
20 else
21 {
22    int n1 = Convert.ToInt32(Console.ReadLine());
23    int n2 = Convert.ToInt32(Console.ReadLine());
24    Console.WriteLine("The division of the two numbers is "+Convert.ToString(n1 / n2));
25 }
26

```

```

Terminal - A1
Subtraction
2
3
The difference of two numbers is -1

```

Build: 0 errors, 1 warning

```

Terminal - A1
Addition
1
3
The sum of the numbers is 4

```

Build: 0 errors, 1 warning

```

Terminal - A1
Multiplication
2
3
The product of two numbers is 6

```

Build: 0 errors, 1 warning

```

Terminal - A1
Division
6
3
The division of the two numbers is 2

```

Build: 0 errors, 1 warning

2. Accept average marks of five students. Display the highest marks obtained.

```
Console.WriteLine("Enter the 5 Marks of the Student");
```

```
int highest_Mark = 0;
```

```
for(int i = 0; i < 5; i++)
```

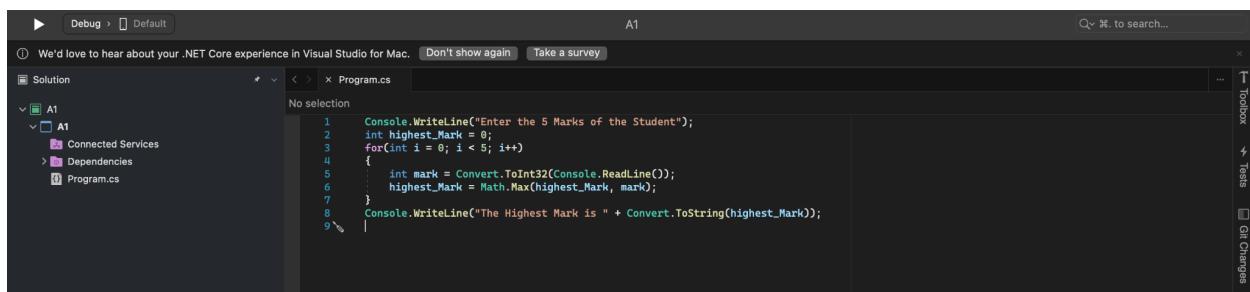
```
{
```

```
    int mark = Convert.ToInt32(Console.ReadLine());
```

```
    highest_Mark = Math.Max(highest_Mark, mark);
```

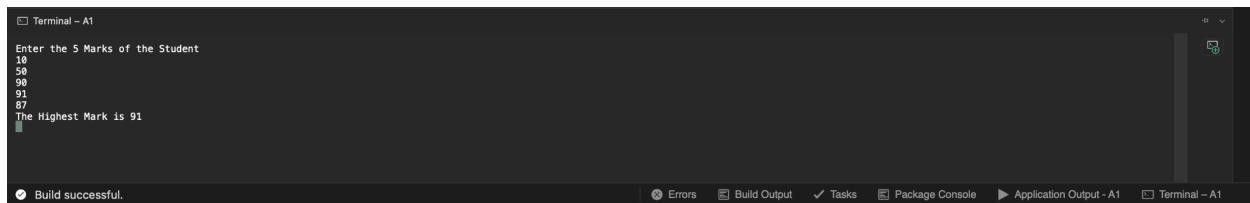
```
}
```

```
Console.WriteLine("The Highest Mark is " + Convert.ToString(highest_Mark));
```



A screenshot of the Visual Studio IDE. The top bar shows 'Debug' and 'Default'. The title bar says 'A1'. The left sidebar shows a solution named 'A1' with a file 'Program.cs'. The main code editor window contains the following C# code:

```
1 Console.WriteLine("Enter the 5 Marks of the Student");
2 int highest_Mark = 0;
3 for(int i = 0; i < 5; i++)
4 {
5     int mark = Convert.ToInt32(Console.ReadLine());
6     highest_Mark = Math.Max(highest_Mark, mark);
7 }
8 Console.WriteLine("The Highest Mark is " + Convert.ToString(highest_Mark));
9
```



The bottom part of the IDE shows a terminal window titled 'Terminal - A1'. It displays the following text:

```
Enter the 5 Marks of the Student
10
50
90
91
87
The Highest Mark is 91
```

At the bottom of the terminal window, it says 'Build successful.'

3. Write a static method to accept a param array of integers. The method should find the sum of all the integers passed and display the result. Write a client program to call the method.

```
class A1
```

```
{
```

```
public static int ArrSum(int[] arr)
```

```
{
```

```
    return arr.Sum();
```

```

}

public static void Main(string[] args)
{

    Console.WriteLine("Enter the number of elements in the Array");

    int n = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("Enter the elements in the Array");

    int[] k = new int[n];

    for (int i = 0; i < n; i++)
    {

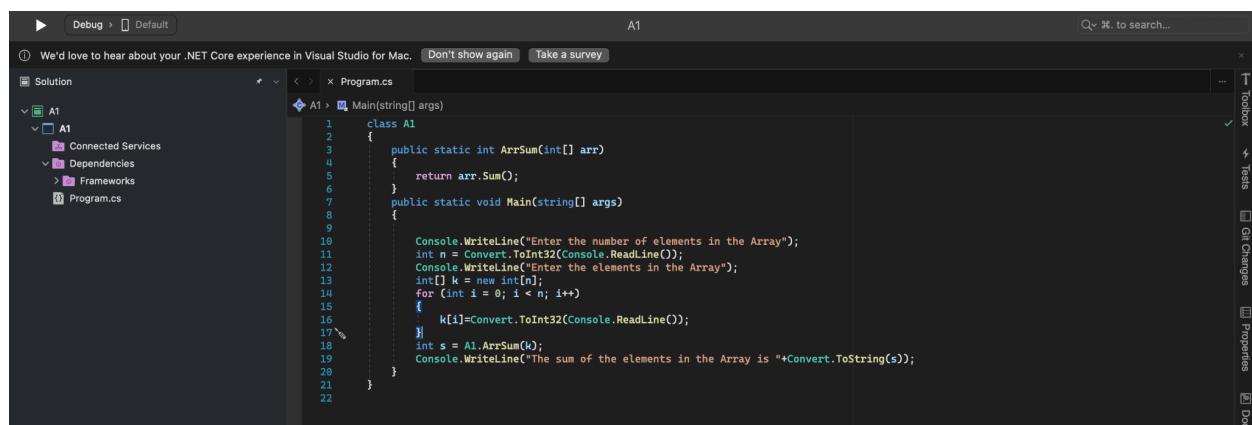
        k[i] = Convert.ToInt32(Console.ReadLine());
    }

    int s = A1.ArrSum(k);

    Console.WriteLine("The sum of the elements in the Array is " + Convert.ToString(s));
}

}

```



The screenshot shows the Visual Studio for Mac interface with the following details:

- Solution Explorer:** Shows a project named "A1" with a single file "Program.cs".
- Code Editor:** Displays the C# code for the "Main" method of "Program.cs".
- Status Bar:** Shows "A1" and a search bar.
- Toolbars and Menus:** Standard Visual Studio for Mac menus like "File", "Edit", "View", "Project", "Tools", "Help", and "Visual Studio".
- Right Sidebar:** Includes "Toolbox", "Test", "Git Changes", "Properties", and "Doc" tabs.

```

Terminal - A1
Enter the number of elements in the Array
5
Enter the elements in the Array
1
2
3
4
5
The sum of the elements in the Array is 15
Build successful.

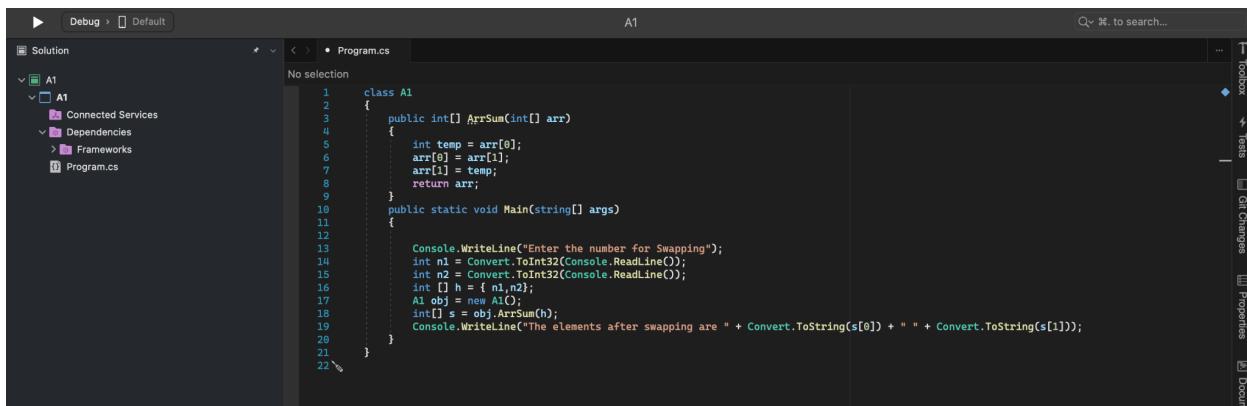
```

4. Write a method to swap two integers. The client code should call the method and print the swapped value.

```

class A1
{
    public int[] ArrSum(int[] arr)
    {
        int temp = arr[0];
        arr[0] = arr[1];
        arr[1] = temp;
        return arr;
    }
    public static void Main(string[] args)
    {
        Console.WriteLine("Enter the number for Swapping");
        int n1 = Convert.ToInt32(Console.ReadLine());
        int n2 = Convert.ToInt32(Console.ReadLine());
        int [] h = { n1,n2 };
        A1 obj = new A1();
        int[] s = obj.ArrSum(h);
        Console.WriteLine("The elements after swapping are " + Convert.ToString(s[0]) + " " +
Convert.ToString(s[1]));
    }
}

```



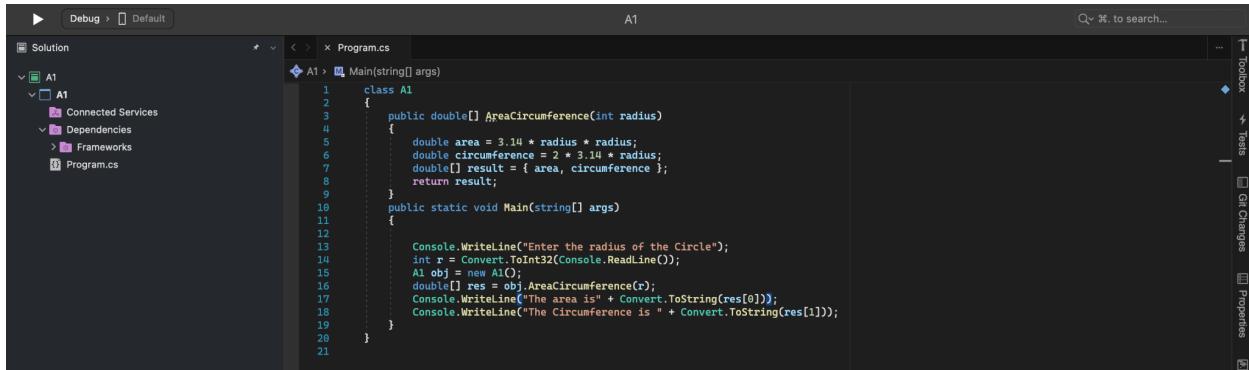
```
Terminal - A1
Enter the number for Swapping
23
42
The elements after swapping are 42 23

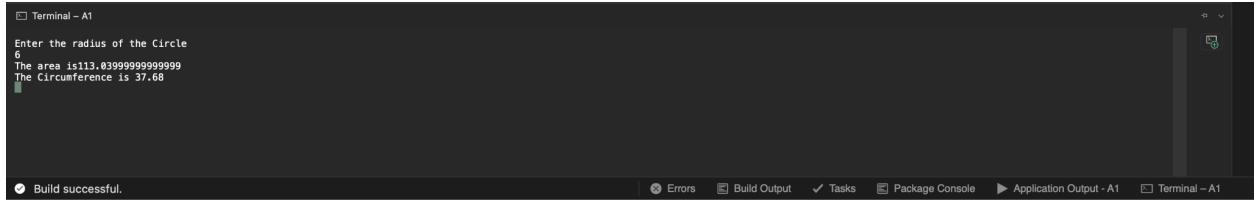
Build successful.
```

5. Write a single method that calculates the area and circumference of the circle. The area and circumference should be displayed through the client code.

```
class A1
{
    public double[] AreaCircumference(int radius)
    {
        double area = 3.14 * radius * radius;
        double circumference = 2 * 3.14 * radius;
        double[] result = { area, circumference };
        return result;
    }
    public static void Main(string[] args)
    {

        Console.WriteLine("Enter the radius of the Circle");
        int r = Convert.ToInt32(Console.ReadLine());
        A1 obj = new A1();
        double[] res = obj.AreaCircumference(r);
        Console.WriteLine("The area is" + Convert.ToString(res[0]));
        Console.WriteLine("The Circumference is " + Convert.ToString(res[1]));
    }
}
```





```
Terminal - A1
Enter the radius of the Circle
6
The area is113.03999999999999
The Circumference is 37.68

Build successful.
```

6. Create a structure Book which contains the following members:

```
bookId, title, price, bookType
```

Type of the book should be an enumerated data type with values as Magazine, Novel, ReferenceBook, Miscellaneous. Write a console based application to do the following tasks.

Accept the details of the book

Display the details of the book. The type of book should be displayed as a string e.g.:Magazine

Note: Use methods for accepting and displaying details.

```
class Book
```

```
{
```

```
    string bookId;
```

```
    string title;
```

```
    double price;
```

```
    string bookType;
```

```
    public Book(string a, string b, double c, string d)
```

```
{
```

```
        this.bookId = a;
```

```
        this.title = b;
```

```
        this.price = c;
```

```
        if (d == "ReferenceBook" || d == "Magazine" || d == "Miscellaneous" || d == "Novel")
```

```
{  
    this.bookType = d;  
}  
}  
  
public Book()  
{  
}  
  
}  
  
public void Display()  
{  
    Console.WriteLine("The Book Id is " + this.bookId);  
    Console.WriteLine("The title of the book is " + this.title);  
    Console.WriteLine("The price of the book is " + Convert.ToString(this.price));  
    Console.WriteLine("The Book type is " + this.bookType);  
}  
}  
  
class A1  
{  
    public static void Main(string[] args)  
    {  
        Book B1 = new Book("B101", "The Book of Why", 350.5, "ReferenceBook");  
        B1.Display();  
    }  
}
```

```
}
```

```
Program.cs
```

```
A1 > No selection
```

```
1  class Book
2  {
3      string bookId;
4      string title;
5      double price;
6      string bookType;
7
8      public Book(string a, string b, double c, string d)
9      {
10         this.bookId = a;
11         this.title = b;
12         this.price = c;
13         if (d == "ReferenceBook" || d == "Magazine" || d == "Miscellaneous" || d == "Novel")
14         {
15             this.bookType = d;
16         }
17     }
18
19     public Book()
20     {
21     }
22
23     public void Display()
24     {
25         Console.WriteLine("The Book Id is " + this.bookId);
26         Console.WriteLine("The title of the book is " + this.title);
27         Console.WriteLine("The price of the book is " + Convert.ToString(this.price));
28         Console.WriteLine("The Book type is " + this.bookType);
29     }
30 }
31
32 class A1
33 {
34     public static void Main(string[] args)
35     {
36         Book B1 = new Book("B101", "The Book of Why", 350.5, "ReferenceBook");
37         B1.Display();
38     }
39 }
40 }
```

```
Terminal - A1
```

```
The Book Id is B101
The title of the book is The Book of Why
The price of the book is 350.5
The Book type is ReferenceBook
```

```
Errors Build Output Tasks Package Console Application Output - A1 Terminal - A1
```

Assignment - 2

1. Develop Employee Management System for Litware Organization. Write a Class Library project LitwareLib.

a. Add class Employee with following private members:

- EmpNo int
- EmpName string
- Salary double
- HRA double
- TA double
- DA double
- PF double
- TDS double
- NetSalary double
- GrossSalary double.

Write methods for accepting EmpNo, EmpName and Salary. HRA, TA, DA, PPF, TDS, NET, GROSS should be calculated automatically. Follow the table for calculations.

Salary	HRA % of Salary	TA % of Salary	DA % of Salary
<5000	10	5	15
<10000	15	10	20
<15000	20	15	25
<20000	25	20	30
>=20000	30	25	35

GrossSalary = Salary + HRA + TA + DA.

Calculate PF, TDS and Net salary in a function named “CalculateSalary()”

PF = 10 % of GrossSalary. TDS = 18 % of GrossSalary.

NetSalary = GrossSalary – (PF + TDS).

Write a console application Employee Management which allow HR staff member to register newly joined employee with EmpNo, EmpName and Salary. Display gross salary of employee on console. LitwareLib class Library will be used in Test console application for creating objects and invoking functionality of Employee class. Use Exception Handling mechanism wherever necessary.

```
class Employee
{
    int EmpNo { get; set; }
    string EmpName {get; set;}
    double Salary { get; set; }
    double HRA;
```

```
double TA;
double DA;
double PF;
double TDS;
double NetSalary;
double GrossSalary;

public Employee()
{
}

public Employee(int ENO, string ENAME, double SAL)
{
    this.EmpNo = ENO;
    this.EmpName = ENAME;
    this.Salary = SAL;
}

public double CalculateSalary()
{
    if (this.Salary < 5000)
    {
        this.HRA = 0.1 * this.Salary;
        this.TA = 0.05 * this.Salary;
        this.DA = 0.15 * this.Salary;
    }

    else if (this.Salary < 10000)
    {
        this.HRA = 0.15 * this.Salary;
        this.TA = 0.1 * this.Salary;
        this.DA = 0.2 * this.Salary;
    }

    else if (this.Salary < 15000)
    {
        this.HRA = 0.2 * this.Salary;
        this.TA = 0.15 * this.Salary;
        this.DA = 0.25 * this.Salary;
    }

    else if (this.Salary < 20000)
    {
```

```

        this.HRA = 0.25 * this.Salary;
        this.TA = 0.2 * this.Salary;
        this.DA = 0.3 * this.Salary;
    }

    else
    {
        this.HRA = 0.3 * this.Salary;
        this.TA = 0.25 * this.Salary;
        this.DA = 0.35 * this.Salary;
    }

    this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;

    this.PF = 0.1 * this.GrossSalary;
    this.TDS = 0.18 * this.GrossSalary;
    this.NetSalary = this.GrossSalary - (this.PF + this.TDS);

    return this.GrossSalary;
}

}

class Assignment2
{
    public static void Main(string[] args)
    {
        Employee E1 = new Employee(1304, "M Rajesh Kannan", 20000);
        double grossSalary = E1.CalculateSalary();
        Console.WriteLine("The Gross Salary is "+Convert.ToString(grossSalary));
    }
}

```

The Gross Salary is 38000

Build successful.

```
Assignment2
Assignment2 > Program.cs
  57     {
  58         this.HRA = 0.3 * this.Salary;
  59         this.TA = 0.25 * this.Salary;
  60         this.DA = 0.35 * this.Salary;
  61     }
  62
  63     this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;
  64
  65     this.PF = 0.1 * this.GrossSalary;
  66     this.TDS = 0.18 * this.GrossSalary;
  67     this.NetSalary = this.GrossSalary - (this.PF + this.TDS);
  68
  69     return this.GrossSalary;
  70
  71 }
  72
  73
  74 class Assignment2
  75 {
  76     public static void Main(string[] args)
  77     {
  78         Employee E1 = new Employee(1304, "M Rajesh Kannan", 20000);
  79         double grossSalary = E1.CalculateSalary();
  80         Console.WriteLine("The Gross Salary is "+Convert.ToString(grossSalary));
  81     }
  82 }
  83
```

The Gross Salary is 13050

Build: 0 errors, 1 warning

```
Assignment2
Assignment2 > Program.cs
  57     {
  58         this.HRA = 0.3 * this.Salary;
  59         this.TA = 0.25 * this.Salary;
  60         this.DA = 0.35 * this.Salary;
  61     }
  62
  63     this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;
  64
  65     this.PF = 0.1 * this.GrossSalary;
  66     this.TDS = 0.18 * this.GrossSalary;
  67     this.NetSalary = this.GrossSalary - (this.PF + this.TDS);
  68
  69     return this.GrossSalary;
  70
  71 }
  72
  73
  74 class Assignment2
  75 {
  76     public static void Main(string[] args)
  77     {
  78         Employee E1 = new Employee(1304, "M Rajesh Kannan", 9000);
  79         double grossSalary = E1.CalculateSalary();
  80         Console.WriteLine("The Gross Salary is "+Convert.ToString(grossSalary));
  81     }
  82 }
  83
```

Assignment - 3

1. Create a hierarchy of Employee, Manager, MarketingExecutive in the Employee Management System. They should have the following functionality.

- a. Manager with following private members.
 - Petrol Allowance: 8 % of Salary.
 - Food Allowance : 13 % of Salary.
 - Other Allowances : 3% of Salary.

Calculate GrossSalary by adding above allowances. Override CalculateSalary() method to calculate Net Salary. NetSalary. PF calculation should not consider above allowances.

- a. MarketingExecutive with following private members.
 - Kilometer travel
 - Tour Allowances : Rs 5/- per Kilometer (Automatically generated).
 - Telephone Allowances : Rs.1000/-

Calculate GrossSalary by adding above allowances. Override CalculateSalary(). NetSalary,PF calculation should not consider above allowances.

Implement IPrintable interface for every Employee which will allow them to print details of Employee on console.

```
interface IPrintable
{
    public void Display();
}

class Employee : IPrintable
{
    int EmpNo { get; set; }
    string EmpName { get; set; }
    double Salary { get; set; }
    double HRA;
    double TA;
    double DA;
    double PF;
    double TDS;
    double NetSalary;
    double GrossSalary;
```

```
public Employee()
{
}

public Employee(int ENO, string ENAME, double SAL)
{
    this.EmpNo = ENO;
    this.EmpName = ENAME;
    this.Salary = SAL;
}

public double CalculateSalary()
{
    if (this.Salary < 5000)
    {
        this.HRA = 0.1 * this.Salary;
        this.TA = 0.05 * this.Salary;
        this.DA = 0.15 * this.Salary;
    }

    else if (this.Salary < 10000)
    {
        this.HRA = 0.15 * this.Salary;
        this.TA = 0.1 * this.Salary;
        this.DA = 0.2 * this.Salary;
    }

    else if (this.Salary < 15000)
    {
        this.HRA = 0.2 * this.Salary;
        this.TA = 0.15 * this.Salary;
        this.DA = 0.25 * this.Salary;
    }

    else if (this.Salary < 20000)
    {
        this.HRA = 0.25 * this.Salary;
        this.TA = 0.2 * this.Salary;
        this.DA = 0.3 * this.Salary;
    }

    else
    {

```

```

        this.HRA = 0.3 * this.Salary;
        this.TA = 0.25 * this.Salary;
        this.DA = 0.35 * this.Salary;
    }

    this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;

    this.PF = 0.1 * this.GrossSalary;
    this.TDS = 0.18 * this.GrossSalary;
    this.NetSalary = this.GrossSalary - (this.PF + this.TDS);

    return this.GrossSalary;

}

public void Display()
{
    Console.WriteLine("The Name of the Employee is " + this.EmpName);
    Console.WriteLine("The id of the Employee is " + this.EmpNo);
    Console.WriteLine("The Gross Salary of the employee is " +
Convert.ToString(this.GrossSalary));
}
}

class Manager : Employee
{
    double PetrolAllowance;
    double FoodAllowance;
    double OtherAllowances;

    public Manager():base()
    {

    }

    public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)
    {
        this.PetrolAllowance = 0.08 * SAL;
        this.FoodAllowance = 0.13 * SAL;
        this.OtherAllowances = 0.03 * SAL;
    }

    public double CalculateSalary()

```

```

    {
        double gross = base.CalculateSalary();
        double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance +
this.OtherAllowances;
        return grossAfterAllowances;
    }
}

class MarketExecutive : Employee
{
    double Kilometers;
    double TourAllowances;
    int TelephoneAllowances;

    public MarketExecutive() : base()
    {

    }

    public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,
ENAME, SAL)
    {
        this.TourAllowances = 5 * Km;
        this.TelephoneAllowances = 1000;
    }

    public double CalculateSalary()
    {
        double gross = base.CalculateSalary();
        double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
        return grossAfterAllowances;
    }
}

class Assignment2
{
    public static void Main(string[] args)
    {
        Manager M1 = new Manager(1183, "Anirudh", 9000);
        double grossSalary = M1.CalculateSalary();
        M1.Display();
    }
}

```

}

The screenshot shows a Visual Studio Code interface with the following details:

- Solution Explorer:** Shows a project named "Assignment3" containing files "Assignment3.cs", "Connected Services", "Dependencies", and "Program.cs".
- Editor:** Displays the content of "Program.cs". The code defines a class `MarketExecutive` which inherits from `Employee`. It includes methods for calculating salary based on kilometers traveled and tour allowances, and a `Display()` method. The `Main` method creates an instance of `MarketExecutive` and calls its `Display()` method.
- Terminal:** Shows the output of the program execution:

```
The Name of the Employee is M Rajesh Kannan
The id of the Employee is 1384
The Gross Salary of the employee is 38000
```
- Bottom Status Bar:** Shows build status (0 errors, 4 warnings), terminal tabs, and navigation icons.

The screenshot shows a Visual Studio Code interface with the following details:

- Solution Explorer:** Shows a project named "Assignment3" with files "Program.cs" and "Assignment2.cs".
- Editor:** The "Program.cs" file is open, displaying code for a class hierarchy. It includes a base class "Employee" and two derived classes, "MarketExecutive" and "Assignment2". The "MarketExecutive" class has properties for kilometers, tour allowances, and telephone allowances, and a method to calculate salary after allowances. The "Assignment2" class has a static Main method that creates a "Manager" object and prints its details.
- Terminal:** The output terminal shows the execution of the program:

```
The Name of the Employee is M Rokith Kumar
The id of the Employee is 1384
The Gross Salary of the employee is 22400
```
- Status Bar:** Shows build statistics: "Build: 0 errors, 4 warnings".

The screenshot shows a Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows a solution named "Assignment3" containing a project named "Assignment3".
- Code Editor:** Displays the file "Program.cs" with code for classes "MarketExecutive" and "Assignment2".
- Terminal - Assignment3:** Shows the following output:

```
The Name of the Employee is Anirudh
The Id of the Employee is 1183
The Gross Salary of the employee is 13050
```
- Status Bar:** Shows build statistics: 0 errors, 4 warnings.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus like Debug, View, Tools, etc.
- Right Sidebar:** Includes links to Toolbox, Tests, Git Changes, Properties, Document Outline, and other development tools.

```
113     class MarketExecutive : Employee
114     {
115         double Kilometers;
116         double TourAllowances;
117         int TelephoneAllowances;
118
119
120         public MarketExecutive() : base()
121     {
122     }
123
124         public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME, SAL)
125     {
126         this.TourAllowances = 5 * Km;
127         this.TelephoneAllowances = 1000;
128     }
129
130         public double CalculateSalary()
131     {
132         double gross = base.CalculateSalary();
133         double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
134         return grossAfterAllowances;
135     }
136
137     }
138
139
140     class Assignment2
141     {
142         public static void Main(string[] args)
143     {
144         Manager M1 = new Manager(1183, "Anirudh", 9000);
145         double grossSalary = M1.CalculateSalary();
146         M1.Display();
147     }
148
149     }
150
```

Assignment - 4

1. Write a class called MyStack with following members.

- a. integer array
- b. integer variable to store top position
- c. size of the array.

Implement Push() and Pop() operation. Implement ICloneable interface to perform cloning. Write a client application to perform cloning.

```
interface ICloneable
{
    public stack Clone();
}

public class StackException : Exception
{
    public string ErrorMessage;
    public StackException()
    {

    }

    public StackException(string message) : base(message)
    {
        this.ErrorMessage = message;
    }
}

public class stack : ICloneable
{
    int top { get; set; }
    int size { get; set; }
    int[] arr { get; set; }

    public stack()
    {
    }
}
```

```
}

public stack(int Size)
{
    this.size = Size;
    this.arr = new int[Size];
}

public void push(int data)
{
    if (this.top == this.size)
    {
        try
        {
            throw new StackException("Stack is Full");
        }

        catch (StackException SE)
        {
            Console.WriteLine(SE.ErrorMessage);
        }
    }
    else
    {
        arr[this.top] = data;
        this.top++;
    }
}

public void pop()
{
    if (this.top == 0)
    {
        try
        {
            throw new StackException("The Stack is Empty");
        }
    }
}
```

```
        catch(StackException SE)
        {
            Console.WriteLine(SE.ErrorMessage);
        }
    }
else
{
    this.top--;
}
}

public void Display()
{
    if (this.top == 0)
    {
        Console.WriteLine("The stack is Empty");
    }
else
{
    Console.WriteLine("The elements in the stack are");
    for (int i = 0; i < this.top; i++)
    {
        Console.Write(" " + Convert.ToString(this.arr[i]));
    }
}
}

public stack Clone()
{
    stack CS = new stack(this.size);
    CS.top = this.top;
    for(int i = 0; i < CS.top; i++)
    {
        CS.arr[i] = this.arr[i];
    }
    return CS;
}
```

```
}
```

```
class Assignment4
{
    public static void Main(string[] args)
    {
        stack S1 = new stack(5);

        S1.push(1);
        S1.push(2);
        S1.push(1);
        S1.pop();
        S1.push(1);
        S1.pop();
        S1.push(3);
        stack S2 = S1.Clone();
        // Console.WriteLine(S1.size);
        S2.Display();
    }
}
```

The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure with "Assignment4" selected.
- Program.cs Editor:** Displays the C# code for the `Assignment4` class, specifically the `Main` method which creates a stack `S1`, performs push and pop operations, creates a clone `S2`, and displays the elements.
- Terminal - Assignment4:** Shows the output of the program's execution, displaying the elements in the stack as "1 2 3".
- Status Bar:** Shows build statistics: "Build: 0 errors, 2 warnings".

2. Create a custom exception class named StackException. The Push() and Pop() method should throw object of StackException when the stack is full or empty respectively.

```
Assignment4 > Program.cs
1    using System;
2    using System.Collections.Generic;
3
4    public class Stack
5    {
6        private int[] arr;
7        private int top;
8
9        public Stack(int size)
10       {
11           arr = new int[size];
12           top = -1;
13       }
14
15       public void Push(int value)
16       {
17           if (top == arr.Length - 1)
18               Console.WriteLine("The stack is full");
19           else
20           {
21               arr[++top] = value;
22               Console.WriteLine("Pushed " + value);
23           }
24       }
25
26       public int Pop()
27       {
28           if (top == -1)
29               Console.WriteLine("The stack is empty");
30           else
31           {
32               int value = arr[top];
33               arr[top] = 0;
34               top--;
35               return value;
36           }
37       }
38
39       public void Print()
40       {
41           if (top == -1)
42               Console.WriteLine("The stack is empty");
43           else
44           {
45               Console.WriteLine("The elements in the stack are:");
46               for (int i = 0; i < top; i++)
47               {
48                   Console.Write(" " + Convert.ToString(arr[i]));
49               }
50           }
51       }
52
53       public Stack Clone()
54       {
55           Stack CS = new Stack(this.size);
56           CS.top = this.top;
57           for (int i = 0; i < CS.top; i++)
58           {
59               CS.arr[i] = this.arr[i];
60           }
61           return CS;
62       }
63   }
64
65   class Assignment4
66   {
67       public static void Main(string[] args)
68       {
69           Stack S1 = new Stack(5);
70
71           S1.push(1);
72           S1.push(2);
73           S1.push(3);
74           S1.push(4);
75           S1.push(5);
76           S1.push(6);
77       }
78   }
79
80   Terminal - Assignment4
81   Stack is Full
```

```
Assignment4 > Program.cs
1    using System;
2    using System.Collections.Generic;
3
4    public class Stack
5    {
6        private int[] arr;
7        private int top;
8
9        public Stack(int size)
10       {
11           arr = new int[size];
12           top = -1;
13       }
14
15       public void Push(int value)
16       {
17           if (top == arr.Length - 1)
18               Console.WriteLine("The stack is full");
19           else
20           {
21               arr[++top] = value;
22               Console.WriteLine("Pushed " + value);
23           }
24       }
25
26       public int Pop()
27       {
28           if (top == -1)
29               Console.WriteLine("The stack is empty");
30           else
31           {
32               int value = arr[top];
33               arr[top] = 0;
34               top--;
35               return value;
36           }
37       }
38
39       public void Print()
40       {
41           if (top == -1)
42               Console.WriteLine("The stack is empty");
43           else
44           {
45               Console.WriteLine("The elements in the stack are:");
46               for (int i = 0; i < top; i++)
47               {
48                   Console.Write(" " + Convert.ToString(arr[i]));
49               }
50           }
51       }
52
53       public Stack Clone()
54       {
55           Stack CS = new Stack(this.size);
56           CS.top = this.top;
57           for (int i = 0; i < CS.top; i++)
58           {
59               CS.arr[i] = this.arr[i];
60           }
61           return CS;
62       }
63   }
64
65   class Assignment4
66   {
67       public static void Main(string[] args)
68       {
69           Stack S1 = new Stack(5);
70
71           S1.push(1);
72           S1.push(2);
73           S1.push(1);
74           S1.push(5);
75           S1.push(3);
76           S1.push(3);
77           S1.pop();
78           S1.pop();
79           S1.pop();
80           S1.pop();
81           S1.pop();
82       }
83   }
84
85   Terminal - Assignment4
86   The Stack is Empty
```

Assignment - 5

1. Create following types of arrays

- a. Integer
- b. String

Use System.Array class to perform following operations on them

Copy, Sort, Clear, Reverse

```
class Arrays
{
    public int[] int_array = { 3,2,5,4,7};
    public string[] string_array = {"Rajesh","Anirudh","Charan","Srinivas","Prateek"};

    public void add_int(int n1,int n2,int n3,int n4,int n5)
    {

    }

    public void copy(int[] arr1, int[] arr2)
    {
        System.Array.Copy(arr1, arr2,5);
    }
    public void copy(string[] arr1, string[] arr2)
    {
        System.Array.Copy(arr1, arr2,5);
    }

    public void sort(int[] arr1)
    {
        System.Array.Sort(arr1);
    }
    public void sort(string[] arr1)
    {
        System.Array.Sort(arr1);
    }
}
```

```
public void clear(int[] arr1)
{
    System.Array.Clear(arr1);
}
public void clear(string[] arr1)
{
    System.Array.Clear(arr1);
}

public void reverse(int[] arr1)
{
    System.Array.Reverse(arr1);
}
public void reverse(string[] arr1)
{
    System.Array.Reverse(arr1);
}

public void stringdisplay()
{
    if (this.string_array.Length == 0)
    {
        Console.Write("The array in empty");
    }
    else
    {
        Console.Write("The array elements are");
        for (int i = 0; i < 5; i++)
        {
            Console.Write(" " + Convert.ToString(this.string_array[i]));
        }
        Console.WriteLine("");
    }
}
public void display()
{
```

```
if (this.int_array.Length == 0)
{
    Console.WriteLine("The array is empty");
}
else
{
    Console.WriteLine("The array elements are");
    for (int i = 0; i < 5; i++)
    {
        Console.Write(" " + Convert.ToString(this.int_array[i]));
    }
    Console.WriteLine("");
}

}

class Assignment5
{
    public static void Main(string[] args)
    {
        Arrays a = new Arrays();
        a.stringdisplay();

        a.sort(a.string_array);
        a.stringdisplay();

        a.reverse(a.int_array);
        a.stringdisplay();

        Arrays s = new Arrays();
        a.copy(a.string_array, s.string_array);
        s.stringdisplay();

        s.clear(s.string_array);
        s.stringdisplay();
    }
}
```

The screenshot shows the Visual Studio IDE interface. The solution 'Assignment5' contains a single project 'Assignment5' with a file 'Program.cs'. The code in 'Program.cs' creates an array, sorts it, reverses it, and displays its elements. The output window shows the array elements being printed five times, each time with different values. The status bar at the bottom indicates a build with 0 errors and 2 warnings.

```
public void display()
{
    if (this.int_array.Length == 0)
    {
        Console.WriteLine("The array is empty");
    }
    else
    {
        Console.Write("The array elements are");
        for (int i = 0; i < 5; i++)
        {
            Console.Write(" " + Convert.ToString(this.int_array[i]));
        }
        Console.WriteLine("");
    }
}

class Assignment5
{
    public static void Main(string[] args)
    {
        Arrays a = new Arrays();
        a.display();
        a.sort(a.int_array);
        a.display();
        a.reverse(a.int_array);
        a.display();

        Arrays s = new Arrays();
        a.copy(a.int_array, s.int_array);
        s.display();

        s.clear(s.int_array);
        s.display();
    }
}
```

The array elements are 3 2 5 4 7
The array elements are 2 3 5 7
The array elements are 7 5 4 3 2
The array elements are 0 5 4 3 2
The array elements are 0 0 0 0 0

0 2 | Build: 0 errors, 2 warnings

The screenshot shows the Visual Studio IDE interface. The solution 'Assignment5' contains a single project 'Assignment5' with a file 'Program.cs'. The code in 'Program.cs' creates two string arrays, copies the first array to the second, sorts the second array, and then displays both arrays. The output window shows the array elements being printed five times, each time with different names. The status bar at the bottom indicates a successful build.

```
public void display()
{
    if (this.int_array.Length == 0)
    {
        Console.WriteLine("The array is empty");
    }
    else
    {
        Console.Write("The array elements are");
        for (int i = 0; i < 5; i++)
        {
            Console.Write(" " + Convert.ToString(this.int_array[i]));
        }
        Console.WriteLine("");
    }
}

class Assignment5
{
    public static void Main(string[] args)
    {
        Arrays a = new Arrays();
        a.stringdisplay();
        a.sort(a.string_array);
        a.stringdisplay();
        a.reverse(a.int_array);
        a.stringdisplay();

        Arrays s = new Arrays();
        a.copy(a.string_array, s.string_array);
        s.stringdisplay();

        s.clear(s.string_array);
        s.stringdisplay();
    }
}
```

The array elements are Rajesh Anirudh Charan Srinivas Prateek
The array elements are Anirudh Charan Prateek Rajesh Srinivas
The array elements are Anirudh Charan Prateek Rajesh Srinivas
The array elements are Anirudh Charan Prateek Rajesh Srinivas
The array elements are

Build successful.

2. Use collection class such as **ArrayList** to hold more than one employee objects in Employee Management application. Display all Employee details which are stored in collection.

```
interface IPrintable
```

```
{
```

```
    public void Display();
```

```
}
```

```
class Employee : IPrintable
```

```
{
```

```
    int EmpNo { get; set; }
```

```
    string EmpName { get; set; }
```

```
    double Salary { get; set; }
```

```
    double HRA;
```

```
    double TA;
```

```
    double DA;
```

```
    double PF;
```

```
    double TDS;
```

```
    double NetSalary;
```

```
    double GrossSalary;
```

```
    public Employee()
```

```
{
```

```
}
```

```
public Employee(int ENO, string ENAME, double SAL)
{
    this.EmpNo = ENO;
    this.EmpName = ENAME;
    this.Salary = SAL;
}
```

```
public double CalculateSalary()
```

```
{
    if (this.Salary < 5000)
    {
        this.HRA = 0.1 * this.Salary;
        this.TA = 0.05 * this.Salary;
        this.DA = 0.15 * this.Salary;
    }
```

```
else if (this.Salary < 10000)
```

```
{
    this.HRA = 0.15 * this.Salary;
    this.TA = 0.1 * this.Salary;
    this.DA = 0.2 * this.Salary;
}
```

```
else if (this.Salary < 15000)
```

```
{  
    this.HRA = 0.2 * this.Salary;  
    this.TA = 0.15 * this.Salary;  
    this.DA = 0.25 * this.Salary;  
}  
  
  
else if (this.Salary < 20000)  
{  
    this.HRA = 0.25 * this.Salary;  
    this.TA = 0.2 * this.Salary;  
    this.DA = 0.3 * this.Salary;  
}  
  
  
else  
{  
    this.HRA = 0.3 * this.Salary;  
    this.TA = 0.25 * this.Salary;  
    this.DA = 0.35 * this.Salary;  
}  
  
  
this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;  
  
  
this.PF = 0.1 * this.GrossSalary;  
this.TDS = 0.18 * this.GrossSalary;
```

```
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);

return this.GrossSalary;

}

public void Display()

{

    Console.WriteLine("The Name of the Employee is " + this.EmpName);

    Console.WriteLine("The id of the Employee is " + this.EmpNo);

    Console.WriteLine("The Gross Salary of the employee is " +

Convert.ToString(this.GrossSalary));

    Console.WriteLine("The NET Salary of the employee is " +

Convert.ToString(this.NetSalary));

}

class Manager : Employee

{

    double PetrolAllowance;

    double FoodAllowance;

    double OtherAllowances;

    public Manager() : base()
```

```
{  
}  
  
public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)  
{  
    this.PetrolAllowance = 0.08 * SAL;  
    this.FoodAllowance = 0.13 * SAL;  
    this.OtherAllowances = 0.03 * SAL;  
}  
  
public double CalculateSalary()  
{  
    double gross = base.CalculateSalary();  
    double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance +  
        this.OtherAllowances;  
    return grossAfterAllowances;  
}  
  
}  
  
class MarketExecutive : Employee  
{  
    double Kilometers;  
    double TourAllowances;  
    int TelephoneAllowances;
```

```
public MarketExecutive() : base()

{

}

public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,
ENAME, SAL)

{
    this.TourAllowances = 5 * Km;
    this.TelephoneAllowances = 1000;
}

public double CalculateSalary()

{
    double gross = base.CalculateSalary();

    double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;

    return grossAfterAllowances;
}

class Assignment5
```

```
public static void Main(string[] args)
{
    System.Collections.ArrayList EmployeeList = new System.Collections.ArrayList();

    MarketExecutive M1 = new MarketExecutive(1183, "Anirudh", 4000, 5);
    double grossSalary1 = M1.CalculateSalary();

    Employee M2 = new Employee(1304, "Rajesh", 25000);
    double grossSalary2 = M2.CalculateSalary();

    Employee M3 = new Employee(1194, "Aditya", 12000);
    double grossSalary3 = M3.CalculateSalary();

    EmployeeList.Add(M1);
    EmployeeList.Add(M2);
    EmployeeList.Add(M3);

    for(int i=0;i<3;i++)
    {
        EmployeeList[i].Display();
    }
}
```

```

Assignment5
Solution Explorer
Assignment5
Program.cs
Assignment5 > Main(string[] args)
121     public MarketExecutive() : base()
122     {
123     }
124 }
125
126     public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME, SAL)
127     {
128         this.TourAllowances = 5 * Km;
129         this.TelephoneAllowances = 1000;
130     }
131
132     public double CalculateSalary()
133     {
134         double gross = base.CalculateSalary();
135         double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
136         return grossAfterAllowances;
137     }
138
139
140
141 class Assignment5
142 {
143     public static void Main(string[] args)
144     {
145         System.Collections.ArrayList EmployeeList = new System.Collections.ArrayList();
146
147         MarketExecutive M1 = new MarketExecutive(1183, "Anirudh", 4000, 5);
148         double grossSalary1 = M1.CalculateSalary();
149
150         Employee M2 = new Employee(1304, "Rajesh", 25000);
151         double grossSalary2 = M2.CalculateSalary();
152
153         Employee M3 = new Employee(1194, "Aditya", 12000);
154         double grossSalary3 = M3.CalculateSalary();
155
156         EmployeeList.Add(M1);
157         EmployeeList.Add(M2);
158         EmployeeList.Add(M3);
}

```

The Name of the Employee is Anirudh
The Id of the Employee is 1183
The Gross Salary of the employee is 5200
The NET Salary of the employee is 3744
The Name of the Employee is Rajesh
The Id of the Employee is 1304
The Gross Salary of the employee is 47500
The NET Salary of the employee is 34200
The Name of the Employee is Aditya
The Id of the Employee is 1194
The Gross Salary of the employee is 19200

3. Write a console based program to create a linked list of Employee objects using the generic class `List<Employee>`. Perform following operations on the list:

- Add a new employee
- Display the list of employees.
- Total number of employees in the list

5. In the assignment 3 above, add a functionality to search an employee on name in the `List<>`.

using `System.Collections`;

interface `IPrintable`

{

 public void `Display()`;

}

```
class Employee : IPrintable

{
    int EmpNo { get; set; }

    public string EmpName { get; set; }

    double Salary { get; set; }

    double HRA;

    double TA;

    double DA;

    double PF;

    double TDS;

    double NetSalary;

    double GrossSalary;

    public Employee()

    {

    }

    public Employee(int ENO, string ENAME, double SAL)

    {
        this.EmpNo = ENO;
        this.EmpName = ENAME;
    }
}
```

```
this.Salary = SAL;  
}  
  
public double CalculateSalary()  
{  
    if (this.Salary < 5000)  
    {  
        this.HRA = 0.1 * this.Salary;  
        this.TA = 0.05 * this.Salary;  
        this.DA = 0.15 * this.Salary;  
    }  
  
    else if (this.Salary < 10000)  
    {  
        this.HRA = 0.15 * this.Salary;  
        this.TA = 0.1 * this.Salary;  
        this.DA = 0.2 * this.Salary;  
    }  
  
    else if (this.Salary < 15000)  
    {  
        this.HRA = 0.2 * this.Salary;  
        this.TA = 0.15 * this.Salary;  
    }  
}
```

```
this.DA = 0.25 * this.Salary;  
}  
  
else if (this.Salary < 20000)  
{  
    this.HRA = 0.25 * this.Salary;  
    this.TA = 0.2 * this.Salary;  
    this.DA = 0.3 * this.Salary;  
}  
  
else  
{  
    this.HRA = 0.3 * this.Salary;  
    this.TA = 0.25 * this.Salary;  
    this.DA = 0.35 * this.Salary;  
}  
  
this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;  
  
this.PF = 0.1 * this.GrossSalary;  
this.TDS = 0.18 * this.GrossSalary;  
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);
```

```
        return this.GrossSalary;

    }

    public void Display()
    {
        Console.WriteLine("The Name of the Employee is " + this.EmpName);
        Console.WriteLine("The id of the Employee is " + this.EmpNo);
        Console.WriteLine("The Gross Salary of the employee is " +
Convert.ToString(this.GrossSalary));
        Console.WriteLine("The NET Salary of the employee is " +
Convert.ToString(this.NetSalary));
    }

}

class EL:Employee,IEnumerable
{
    public System.Collections.ArrayList EList = new ArrayList();

    public IEnumerator GetEnumerator()
    {
        return ((IEnumerable)EList).GetEnumerator();
    }
}
```

```
public Employee Search(string name)

{
    foreach(Employee Emp in this.EList)

    {
        if (Emp.EmpName == name)

        {
            return Emp;

        }

        return new Employee(0, "NULL", 0);

    }

}
```

```
class Manager : Employee

{
    double PetrolAllowance;

    double FoodAllowance;

    double OtherAllowances;

    public Manager() : base()

{
```

```
}

public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)

{

    this.PetrolAllowance = 0.08 * SAL;

    this.FoodAllowance = 0.13 * SAL;

    this.OtherAllowances = 0.03 * SAL;

}

public double CalculateSalary()

{

    double gross = base.CalculateSalary();

    double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance + this.OtherAllowances;

    return grossAfterAllowances;

}

class MarketExecutive : Employee

{

    double Kilometers;

    double TourAllowances;
```

```
int TelephoneAllowances;

public MarketExecutive() : base()

{

}

public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,
ENAME, SAL)

{

    this.TourAllowances = 5 * Km;

    this.TelephoneAllowances = 1000;

}

public double CalculateSalary()

{

    double gross = base.CalculateSalary();

    double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;

    return grossAfterAllowances;

}
```

```
class Assignment5

{
    public static void Main(string[] args)
    {
        System.Collections.ArrayList EmployeeList = new System.Collections.ArrayList();

        MarketExecutive M1 = new MarketExecutive(1183, "Anirudh", 4000, 5);
        double grossSalary1 = M1.CalculateSalary();

        Employee M2 = new Employee(1304, "Rajesh", 25000);
        double grossSalary2 = M2.CalculateSalary();

        Employee M3 = new Employee(1194, "Aditya", 12000);
        double grossSalary3 = M3.CalculateSalary();

        EmployeeList.Add(M1);
        EmployeeList.Add(M2);
        EmployeeList.Add(M3);

        EL list = new EL();
        list.EList = EmployeeList;
        Employee FoundEmployee = list.Search("Rajesh");
```

```

        FoundEmployee.Display();
    }

}

```

The Name of the Employee is Rajesh
The id of the Employee is 1304
The Gross Salary of the employee is 47500
The NET Salary of the employee is 34200

4. Write Custom Generic class MyStack based on assignment of previous session, with Push() and Pop() methods to store any kind of .NET Type.

```
public class StackException : Exception
{
    public string ErrorMessage;
    public StackException()
    {

    }
    public StackException(string message) : base(message)
    {
        this.ErrorMessage = message;
    }
}

public class Mystack <T>
{
    int top { get; set; }
    int size { get; set; }
    T[] arr { get; set; }

    public Mystack()
    {

    }

    public Mystack(int Size)
    {
        this.size = Size;
        this.arr = new T[Size];
    }

    public void push(T data)
    {
        if (this.top == this.size)
        {
            try
            {
                throw new StackException("Stack is Full");
            }
        }
    }
}
```

```
        catch (StackException SE)
        {
            Console.WriteLine(SE.ErrorMessage);
        }
    }
else
{
    arr[this.top] = data;
    this.top++;
}
}

public void pop()
{
    if (this.top == 0)
    {
        try
        {
            throw new StackException("The Stack is Empty");
        }
    }
    catch (StackException SE)
    {
        Console.WriteLine(SE.ErrorMessage);
    }
}
else
{
    this.top--;
}
}

public void Display()
{
    if (this.top == 0)
    {
        Console.WriteLine("The stack is Empty");
    }
    else
    {
        Console.WriteLine("The elements in the stack are");
        for (int i = 0; i < this.top; i++)
        {
            Console.Write(" " + Convert.ToString(this.arr[i]));
        }
    }
}
```

```
        }
    }

public Mystack<T> Clone()
{
    Mystack<T> CS = new Mystack<T>(this.size);
    CS.top = this.top;
    for (int i = 0; i < CS.top; i++)
    {
        CS.arr[i] = this.arr[i];
    }
    return CS;
}

class Assignment4
{
    public static void Main(string[] args)
    {
        Mystack<string> S1 = new Mystack<string>(5);

        S1.push("hello");
        S1.push("rajesh");
        S1.push("Hola");
        S1.pop();
        S1.push("anirudh");
        S1.push("hi");
        S1.pop();
        Mystack<string> S2 = S1.Clone();
        S2.Display();
    }
}
```

Solution Explorer

Assignment5

Program.cs

```
Assignment5
Assignment4 > Main(string[] args)
83     Console.WriteLine(" " + Convert.ToString(this.arr[i]));
84 }
85 }
86 }
87 }
88 public MyStack<T> Clone()
89 {
90     MyStack<T> CS = new MyStack<T>(this.size);
91     CS.top = this.top;
92     for (int i = 0; i < CS.top; i++)
93     {
94         CS.arr[i] = this.arr[i];
95     }
96     return CS;
97 }
98 }
99 class Assignment4
100 {
101     public static void Main(string[] args)
102     {
103         MyStack<int> S1 = new MyStack<int>(5);
104         S1.push(1);
105         S1.push(6);
106         S1.push(5);
107         S1.pop();
108         S1.push(2);
109         S1.push(1);
110         S1.pop();
111         MyStack<int> S2 = S1.Clone();
112         S2.Display();
113     }
114 }
115 }
```

Terminal – Assignment5

```
The elements in the stack are
1 6 2
```

Build: 0 errors, 2 warnings

Solution Explorer

Assignment5

Program.cs

```
Assignment5
Assignment4 > Main(string[] args)
83     Console.WriteLine(" " + Convert.ToString(this.arr[i]));
84 }
85 }
86 }
87 }
88 public MyStack<T> Clone()
89 {
90     MyStack<T> CS = new MyStack<T>(this.size);
91     CS.top = this.top;
92     for (int i = 0; i < CS.top; i++)
93     {
94         CS.arr[i] = this.arr[i];
95     }
96     return CS;
97 }
98 }
99 class Assignment4
100 {
101     public static void Main(string[] args)
102     {
103         MyStack<string> S1 = new MyStack<string>(5);
104         S1.push("Hello");
105         S1.push("rajes");
106         S1.push("Hello");
107         S1.pop();
108         S1.push("anirudh");
109         S1.push("hi");
110         S1.pop();
111         MyStack<string> S2 = S1.Clone();
112         S2.Display();
113     }
114 }
115 }
```

Terminal – Assignment5

```
The elements in the stack are
hello rajesh anirudh
```

Build: 0 errors, 2 warnings

6. Create a class named Player that contains Player name, runs scored as data members. Create a class named Team that contains an array of Player. Implement IEnumerable interface for class Team.

Write a console based application to create an object named India. Use foreach loop to iterate through the object India to display information about its players.

```
using System.Collections;
```

```
class Player
{
    public string name { get; set; }
    public int runs_scored { get; set; }

    public Player()
    {

    }

    public Player(string name, int runs)
    {
        this.name = name;
        this.runs_scored = runs;
    }
}
```

```
class Team : IEnumerable
{
    List<Player> Players = new List<Player>();

    public Team()
    {

    }

    public void Add(Player P)
    {
        Players.Add(P);
    }

    public IEnumerator GetEnumerator()
    {
        return Players.GetEnumerator();
    }
}
```

```

public void Display()
{
    foreach(Player P in Players)
    {
        Console.WriteLine("The Player name is "+P.name+" and the runs scored is "+Convert.ToString(P.runs_scored));
    }
}
}

class Assignment5
{
    public static void Main(string[] args)
    {
        Player p1 = new Player("Virat", 1000);
        Player p2 = new Player("Rohit", 900);
        Player p3 = new Player("Dhoni", 1100);
        Team India = new Team();
        India.Add(p1);
        India.Add(p2);
        India.Add(p3);
        India.Display();
    }
}

```

The screenshot shows the Visual Studio IDE interface. The code editor displays the C# program above. The terminal window at the bottom shows the execution results:

```

The Player name is Virat and the runs scored is 1000
The Player name is Rohit and the runs scored is 900
The Player name is Dhoni and the runs scored is 1100

```

The status bar at the bottom indicates "Build: 0 errors, 1 warning".

7. Use an iterator to iterate through the players in the above example.

```
using System.Collections;
```

```
class Player  
{  
    public string name { get; set; }  
    public int runs_scored { get; set; }
```

```
    public Player()
```

```
{
```

```
}
```

```
    public Player(string name, int runs)
```

```
{
```

```
    this.name = name;
```

```
    this.runs_scored = runs;
```

```
}
```

```
}
```

```
class Team : IEnumerable
```

```
{
```

```
    List<Player> Players = new List<Player>();
```

```
public Team()  
{  
  
}  
  
public void Add(Player P)  
{  
    Players.Add(P);  
  
}  
  
public IEnumerator GetEnumerator()  
{  
    return Players.GetEnumerator();  
  
}  
  
public void Display()  
{  
    foreach(Player P in Players)  
    {  
        Console.WriteLine("The Player name is "+P.name+" and the runs scored is  
        "+Convert.ToString(P.runs_scored));  
    }  
}  
  
public void DisplayUsingEnumerator()
```

```
{  
    var enumerator = Players.GetEnumerator();  
  
    while (enumerator.MoveNext())  
  
    {  
        Console.WriteLine("The Player name is " + enumerator.Current.name + " and the runs scored is "  
        + Convert.ToString(enumerator.Current.runs_scored));  
    }  
}  
  
}  
  
class Assignment5  
{  
    public static void Main(string[] args)  
    {  
        Player p1 = new Player("Virat", 1000);  
  
        Player p2 = new Player("Rohit", 900);  
  
        Player p3 = new Player("Dhoni", 1100);  
  
        Team India = new Team();  
  
        India.Add(p1);  
  
        India.Add(p2);  
  
        India.Add(p3);  
  
        India.DisplayUsingEnumerator();  
    }  
}
```

The screenshot shows a Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows a solution named "Assignment5" containing a project named "Assignment5" with files "Connected Services", "Dependencies", and "Program.cs".
- Code Editor:** Displays the "Program.cs" file with the following code:

```
33     public IEnumerator GetEnumerator()
34     {
35         return Players.GetEnumerator();
36     }
37
38     public void Display()
39     {
40         foreach(Player P in Players)
41         {
42             Console.WriteLine("The Player name is "+P.name+" and the runs scored is "+Convert.ToString(P.runs_scored));
43         }
44     }
45
46     public void DisplayUsingEnumerator()
47     {
48         var enumerator = Players.GetEnumerator();
49         while (enumerator.MoveNext())
50         {
51             Console.WriteLine("The Player name is " + enumerator.Current.name + " and the runs scored is " + Convert.ToString(enumerator.Current.runs_scored));
52         }
53     }
54
55 }
56
57 class Assignment5
58 {
59     public static void Main(string[] args)
60     {
61         Player p1 = new Player("Virat", 1000);
62         Player p2 = new Player("Rohit", 900);
63         Player p3 = new Player("Dhoni", 1100);
64         Team India = new Team();
65         India.Add(p1);
66         India.Add(p2);
67         India.Add(p3);
68         India.DisplayUsingEnumerator();
69     }
70 }
```
- Terminal - Assignment5:** Shows the output of the program:

```
The Player name is Virat and the runs scored is 1000
The Player name is Rohit and the runs scored is 900
The Player name is Dhoni and the runs scored is 1100
```
- Status Bar:** Shows build information: 0 errors, 1 warning, and Build: 0 errors, 1 warning.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus like "File", "Edit", "View", "Tools", etc., are visible along the top.

Assignment - 6

1. In Assignment 3 of previous session, print the details of Managers with the help of delegate EmployeeDelegate. (UniCast Delegate)

```
public delegate void EDisplay();
```

```
interface IPrintable
```

```
{
```

```
    public void Display();
```

```
}
```

```
class Employee : IPrintable
```

```
{
```

```
    int EmpNo { get; set; }
```

```
    string EmpName { get; set; }
```

```
    double Salary { get; set; }
```

```
    double HRA;
```

```
    double TA;
```

```
    double DA;
```

```
    double PF;
```

```
    double TDS;
```

```
    double NetSalary;
```

```
    double GrossSalary;
```

```
    public Employee()
```

```
{
```

```
}
```

```
public Employee(int ENO, string ENAME, double SAL)
```

```
{
```

```
    this.EmpNo = ENO;
```

```
    this.EmpName = ENAME;
```

```
    this.Salary = SAL;
```

```
}
```

```
public double CalculateSalary()
```

```
{
```

```
    if (this.Salary < 5000)
```

```
{
```

```
        this.HRA = 0.1 * this.Salary;
```

```
        this.TA = 0.05 * this.Salary;
```

```
        this.DA = 0.15 * this.Salary;
```

```
}
```

```
else if (this.Salary < 10000)
```

```
{
```

```
    this.HRA = 0.15 * this.Salary;
```

```
    this.TA = 0.1 * this.Salary;
```

```
    this.DA = 0.2 * this.Salary;
```

```
}
```

```
else if (this.Salary < 15000)
```

```
{
```

```
    this.HRA = 0.2 * this.Salary;
```

```
    this.TA = 0.15 * this.Salary;
```

```
    this.DA = 0.25 * this.Salary;
```

```
}
```

```
else if (this.Salary < 20000)
```

```
{
```

```
    this.HRA = 0.25 * this.Salary;
```

```
    this.TA = 0.2 * this.Salary;
```

```
    this.DA = 0.3 * this.Salary;
```

```
}
```

```
else
```

```
{
```

```
    this.HRA = 0.3 * this.Salary;
```

```
    this.TA = 0.25 * this.Salary;
```

```
    this.DA = 0.35 * this.Salary;
```

```
}
```

```
this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;
```

```
this.PF = 0.1 * this.GrossSalary;  
this.TDS = 0.18 * this.GrossSalary;  
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);  
  
return this.GrossSalary;  
}
```

```
public void Display()  
{  
    Console.WriteLine("The Name of the Employee is " + this.EmpName);  
    Console.WriteLine("The id of the Employee is " + this.EmpNo);  
    Console.WriteLine("The Gross Salary of the employee is " + Convert.ToString(this.GrossSalary));  
    Console.WriteLine("The NET Salary of the employee is " + Convert.ToString(this.NetSalary));  
}
```

```
}
```

```
class Manager : Employee  
{  
    double PetrolAllowance;  
    double FoodAllowance;
```

```
double OtherAllowances;

public Manager() : base()
{
}

public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)
{
    this.PetrolAllowance = 0.08 * SAL;
    this.FoodAllowance = 0.13 * SAL;
    this.OtherAllowances = 0.03 * SAL;
}

public double CalculateSalary()
{
    double gross = base.CalculateSalary();
    double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance +
    this.OtherAllowances;
    return grossAfterAllowances;
}

class MarketExecutive : Employee
```

```
{  
    double Kilometers;  
  
    double TourAllowances;  
  
    int TelephoneAllowances;  
  
  
  
  
    public MarketExecutive() : base()  
  
    {  
  
    }  
  
  
  
    public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME,  
SAL)  
  
    {  
        this.TourAllowances = 5 * Km;  
  
        this.TelephoneAllowances = 1000;  
  
    }  
  
  
  
    public double CalculateSalary()  
  
    {  
        double gross = base.CalculateSalary();  
  
        double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;  
  
        return grossAfterAllowances;  
  
    }  
}
```

```

class Assignment2

{
    public static void Main(string[] args)
    {
        Manager M1 = new Manager(1183, "Anirudh", 4000);

        double grossSalary = M1.CalculateSalary();

        EDisplay delegateddisplay = new EDisplay(M1.Display);

        delegateddisplay();

    }
}

```

The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure with files: Assignment6, Assignment6.cs, Connected Services, Dependencies, and Program.cs.
- Code Editor:** Displays the `Program.cs` file containing C# code for calculating employee salaries.
- Terminal - Assignment6:** Shows the command-line output of the application's execution.
- Status Bar:** Shows build statistics: 0 errors, 4 warnings.

```

Assignment2 > Program.cs
Assignment6
Assignment6.cs
Connected Services
Dependencies
Program.cs

117    class MarketExecutive : Employee
118    {
119        double Km;
120        double TourAllowances;
121        int TelephoneAllowances;
122
123
124
125        public MarketExecutive() : base()
126        {
127        }
128
129
130        public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME, SAL)
131        {
132            this.TourAllowances = 5 * Km;
133            this.TelephoneAllowances = 1000;
134        }
135
136        public double CalculateSalary()
137        {
138            double gross = base.CalculateSalary();
139            double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
140            return grossAfterAllowances;
141        }
142
143
144
145    class Assignment2
146    {
147        public static void Main(string[] Args)
148        {
149            Manager M1 = new Manager(1183, "Anirudh", 4000);
150            double grossSalary = M1.CalculateSalary();
151            EDisplay delegateddisplay = new EDisplay(M1.Display);
152            delegateddisplay();
153        }
154    }
}

Terminal – Assignment6
The Name of the Employee is Anirudh
The id of the Employee is 1183
The Gross Salary of the employee is 5200
The NET Salary of the employee is 3744

```

2. Create a MultiCast Delegate by adding address of another method to the above delegate to list the details of Marketing Executive.

```
public delegate void EDisplay();
```

```
interface IPrintable
```

```
{
```

```
    public void Display();
```

```
}
```

```
class Employee : IPrintable
```

```
{
```

```
    int EmpNo { get; set; }
```

```
    string EmpName { get; set; }
```

```
    double Salary { get; set; }
```

```
    double HRA;
```

```
    double TA;
```

```
    double DA;
```

```
    double PF;
```

```
    double TDS;
```

```
    double NetSalary;
```

```
    double GrossSalary;
```

```
    public Employee()
```

```
{
```

```
}
```

```
public Employee(int ENO, string ENAME, double SAL)
```

```
{
```

```
    this.EmpNo = ENO;
```

```
    this.EmpName = ENAME;
```

```
    this.Salary = SAL;
```

```
}
```

```
public double CalculateSalary()
```

```
{
```

```
    if (this.Salary < 5000)
```

```
{
```

```
        this.HRA = 0.1 * this.Salary;
```

```
        this.TA = 0.05 * this.Salary;
```

```
        this.DA = 0.15 * this.Salary;
```

```
}
```

```
else if (this.Salary < 10000)
```

```
{
```

```
    this.HRA = 0.15 * this.Salary;
```

```
    this.TA = 0.1 * this.Salary;
```

```
    this.DA = 0.2 * this.Salary;
```

```
}
```

```
else if (this.Salary < 15000)

{
    this.HRA = 0.2 * this.Salary;
    this.TA = 0.15 * this.Salary;
    this.DA = 0.25 * this.Salary;
}

else if (this.Salary < 20000)

{
    this.HRA = 0.25 * this.Salary;
    this.TA = 0.2 * this.Salary;
    this.DA = 0.3 * this.Salary;
}

else

{
    this.HRA = 0.3 * this.Salary;
    this.TA = 0.25 * this.Salary;
    this.DA = 0.35 * this.Salary;
}

this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;
```

```
this.PF = 0.1 * this.GrossSalary;  
this.TDS = 0.18 * this.GrossSalary;  
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);  
  
return this.GrossSalary;  
  
}  
  
  
public void Display()  
{  
    Console.WriteLine("The Name of the Employee is " + this.EmpName);  
    Console.WriteLine("The id of the Employee is " + this.EmpNo);  
    Console.WriteLine("The Gross Salary of the employee is " +  
Convert.ToString(this.GrossSalary));  
    Console.WriteLine("The NET Salary of the employee is " +  
Convert.ToString(this.NetSalary));  
}  
  
}
```

```
class Manager : Employee  
{  
    double PetrolAllowance;  
    double FoodAllowance;
```

```
double OtherAllowances;

public Manager() : base()

{

}

public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)

{

    this.PetrolAllowance = 0.08 * SAL;

    this.FoodAllowance = 0.13 * SAL;

    this.OtherAllowances = 0.03 * SAL;

}

public double CalculateSalary()

{

    double gross = base.CalculateSalary();

    double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance + this.OtherAllowances;

    return grossAfterAllowances;

}

class MarketExecutive : Employee
```

```
{  
    double Kilometers;  
  
    double TourAllowances;  
  
    int TelephoneAllowances;  
  
  
  
  
    public MarketExecutive() : base()  
    {  
  
    }  
  
  
  
    public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,  
ENAME, SAL)  
    {  
        this.TourAllowances = 5 * Km;  
  
        this.TelephoneAllowances = 1000;  
  
    }  
  
  
  
    public double CalculateSalary()  
    {  
        double gross = base.CalculateSalary();  
  
        double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;  
  
        return grossAfterAllowances;  
  
    }  
}
```

```
public void Display()
{
    base.Display();
    Console.WriteLine("The TourAllowances is " + Convert.ToString(this.TourAllowances));
}
}
```

```
class Assignment2
```

```
{
    public static void Main(string[] args)
    {
        Manager M1 = new Manager(1183, "Anirudh", 4000);
        double grossSalary1 = M1.CalculateSalary();
```

```
        MarketExecutive M2 = new MarketExecutive(1183, "Anirudh", 4000, 5);
        double grossSalary2 = M2.CalculateSalary();
```

```
        EDisplay delegateddisplay1 = new EDisplay(M1.Display);
        EDisplay delegateddisplay2 = new EDisplay(M2.Display);
```

```
        Console.WriteLine("The details of the Manager are:");
        delegateddisplay1();
```

```

Console.WriteLine("");

Console.WriteLine("The details of Marketing Executive are:");

delegatedisplay2();

}

}

```

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays `Program.cs` with the following C# code:

```

Assignment6
Assignment2 > Program.cs
Solution
Assignment6
Assignment2
Main(string[] args)
{
    double gross = base.CalculateSalary();
    double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
    return grossAfterAllowances;
}

public void Display()
{
    base.Display();
    Console.WriteLine("The TourAllowances is " + Convert.ToString(this.TourAllowances));
}

class Assignment2
{
    public static void Main(string[] args)
    {
        Manager M1 = new Manager(1183, "Anirudh", 4000);
        double grossSalary1 = M1.CalculateSalary();

        MarketExecutive M2 = new MarketExecutive(1183, "Anirudh", 4000, 5);
        double grossSalary2 = M2.CalculateSalary();

        EDisplay delegatedisplay1 = new EDisplay(M1.Display);
        EDisplay delegatedisplay2 = new EDisplay(M2.Display);

        Console.WriteLine("The details of the Manager are:");
        delegatedisplay1();
        Console.WriteLine("");
        Console.WriteLine("The details of Marketing Executive are:");
        delegatedisplay2();
    }
}

```

The terminal window at the bottom shows the execution results:

```

The details of the Manager are:
The Name of the Employee is Anirudh
The Id of the Employee is 1183
The Gross Salary of the employee is 5200
The NET Salary of the employee is 3744

The details of Marketing Executive are:
The Name of the Employee is Anirudh
The Id of the Employee is 1183
The Gross Salary of the employee is 5200
The NET Salary of the employee is 3744
The TourAllowances is 25

```

Build status: 0 errors, 5 warnings.

3. Write a console Application for banking domain, having Account class with data members as account number, customer name, balance . It should have Withdraw and Deposit methods for performing banking transaction. It should also define UnderBalance, BalanceZero events. These events would be raised when balance of account is less than certain amount and equal to zero respectively.

```
public delegate void balanceEvents();
```

```
class Account
```

```
{
```

```
    event balanceEvents Event1;
```

```
    event balanceEvents Event2;
```

```
    int account_number;
```

```
    double balance;
```

```
    string name;
```

```
    public Account()
```

```
{
```

```
}
```

```
    public Account(int ac, int b, string n)
```

```
{
```

```
        this.name = n;
```

```
        this.balance = b;
```

```
        this.account_number = ac;
```

```
}
```

```
    public void Withdraw(double amount)
```

```
{  
    if (this.balance - amount > 0)  
    {  
        balance -= amount;  
        Console.WriteLine("Withdrawal of " + Convert.ToString(amount) + " successful");  
        Console.WriteLine("The balance of the account is "+Convert.ToString(this.balance));  
    }  
  
    else if(this.balance - amount == 0)  
    {  
        this.Event1 = new balanceEvents(this.zerobalance);  
        this.Event1();  
    }  
  
    else  
    {  
        this.Event2 = new balanceEvents(this.underbalance);  
        this.Event2();  
    }  
}  
  
public void zerobalance()  
{  
    Console.WriteLine("Account Alert!");
```

```
        Console.WriteLine("The account now has zero balance");

    }

public void underbalance()
{
    Console.WriteLine("The transaction has failed!");

    Console.WriteLine("The account does not have sufficient funds for withdrawal");
}

public void Deposit(double amount)
{
    balance += amount;

    Console.WriteLine("Deposited "+Convert.ToString(amount)+" successfully");
}

class Assignment6
{
    public static void Main(string[] args)
    {
        Account A1 = new Account(1304, 10000, "rajesh");

        A1.Withdraw(11000);

    }
}
```

Solution Explorer

Assignment6

Program.cs

```
Assignment6

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

namespace Assignment6
{
    class Account
    {
        public double balance { get; set; }
        public event EventHandler<EventArgs> Event1;
        public event EventHandler<EventArgs> Event2;

        public void Withdraw(double amount)
        {
            if (balance > amount)
            {
                balance -= amount;
                this.Event1 = new balanceEvents(this.zerobalance);
                this.Event1();
            }
            else
            {
                this.Event2 = new balanceEvents(this.underbalance);
                this.Event2();
            }
        }

        public void zerobalance()
        {
            Console.WriteLine("Account Alert!");
            Console.WriteLine("The account now has zero balance");
        }

        public void underbalance()
        {
            Console.WriteLine("The transaction has failed!");
            Console.WriteLine("The account does not have sufficient funds for withdrawal");
        }

        public void Deposit(double amount)
        {
            balance += amount;
            Console.WriteLine("Deposited " + Convert.ToString(amount) + " successfully");
        }
    }

    class Assignment6
    {
        public static void Main(string[] args)
        {
            Account A1 = new Account(1304, 10000, "rajesh");
            A1.Withdraw(1000);
        }
    }
}
```

Terminal – Assignment6

```
Withdrawal of 1000 successful
The balance of the account is 9000
```

0 5 | Build: 0 errors, 5 warnings

Solution Explorer

Assignment6

Program.cs

```
Assignment6

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

namespace Assignment6
{
    class Account
    {
        public double balance { get; set; }
        public event EventHandler<EventArgs> Event1;
        public event EventHandler<EventArgs> Event2;

        public void Withdraw(double amount)
        {
            if (balance > amount)
            {
                balance -= amount;
                this.Event1 = new balanceEvents(this.zerobalance);
                this.Event1();
            }
            else
            {
                this.Event2 = new balanceEvents(this.underbalance);
                this.Event2();
            }
        }

        public void zerobalance()
        {
            Console.WriteLine("Account Alert!");
            Console.WriteLine("The account now has zero balance");
        }

        public void underbalance()
        {
            Console.WriteLine("The transaction has failed!");
            Console.WriteLine("The account does not have sufficient funds for withdrawal");
        }

        public void Deposit(double amount)
        {
            balance += amount;
            Console.WriteLine("Deposited " + Convert.ToString(amount) + " successfully");
        }
    }

    class Assignment6
    {
        public static void Main(string[] args)
        {
            Account A1 = new Account(1304, 10000, "rajesh");
            A1.Withdraw(10000);
        }
    }
}
```

Terminal – Assignment6

```
Account Alert!
The account now has zero balance
```

0 5 | Build: 0 errors, 5 warnings

The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure with files: Assignment6.cs, Connected Services, Dependencies, and Program.cs.
- Code Editor:** Displays the content of Program.cs. The code defines a class Assignment6 with a Main method. It contains methods for zero balance and underbalance, and a Deposit method. It also includes a constructor for Account and a Withdraw method.
- Terminal - Assignment6:** Shows the output of the application. It prints two error messages:
 - The transaction has failed!
 - The account does not have sufficient funds for withdrawal
- Status Bar:** Shows build statistics: 0 errors, 5 warnings.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus are visible along the top and right side of the interface.

Assignment - 7

1. Write a console application which will read text files from mentioned file system location. And list subdirectories from mentioned folder on file system using System.IO namespace and use DirectoryInfo, Directory, File and FileInfo Classes with all the methods present in these classes.

```
using System;
```

```
using System.IO;
```

```
class Assignment7
```

```
{
```

```
    public static void Main(string[] args)
```

```
    {
```

```
        string directory = @"/Users/rajeshkannan/Desktop/RCI Sample Data";
```

```
        var txtFiles = Directory.EnumerateFiles(directory, "*.txt");
```

```
        Console.WriteLine("The text files in the "+directory+" directory are:");
```

```
        foreach (string currentFile in txtFiles)
```

```
        {
```

```
            Console.WriteLine(currentFile);
```

```
        }
```

```
}
```

```
}
```

The screenshot shows the Visual Studio IDE interface. The Solution Explorer on the left lists a project named 'Assignment7' containing a file 'Program.cs'. The code editor in the center displays the following C# code:

```

using System;
using System.IO;
class Assignment7
{
    public static void Main(string[] args)
    {
        string directory = @"C:\Users\rajeshkannan\Desktop\RCI Sample Data";
        var txtFiles = Directory.EnumerateFiles(directory, "*.*");
        Console.WriteLine("The text files in the "+directory+" directory are:");
        foreach (string currentFile in txtFiles)
        {
            Console.WriteLine(currentFile);
        }
    }
}

```

The terminal window at the bottom shows the output of the program:

```

The text files in the /Users/rajeshkannan/Desktop/RCI Sample Data directory are:
/Users/rajeshkannan/Desktop/RCI Sample Data/h1.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/d1.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/d1.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/data.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/d2.txt

```

The status bar at the bottom indicates a build was successful.

```

using System;
using System.IO;

class Assignment7
{
    public static void Main(string[] args)
    {
        string directory = @"/Users/rajeshkannan/Desktop/RCI Sample Data";
        var txtFiles = Directory.EnumerateFiles(directory, "*.*");
        Console.WriteLine("The text files in the "+directory+" directory are:");
        foreach (string currentFile in txtFiles)
        {
            Console.WriteLine(currentFile);
        }

        var directories = Directory.EnumerateDirectories(directory);
        Console.WriteLine("The text files in the " + directory + " directory are:");
        foreach (string currentdirectory in directories)
        {
            Console.WriteLine(currentdirectory);
        }
    }
}

```

```
}
```

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the Solution Explorer displays a project named "Assignment7" with files "Assignment7.cs", "Program.cs", and "Properties". The main code editor window shows the following C# code:

```

using System;
using System.IO;
class Assignment7
{
    public static void Main(string[] args)
    {
        string directory = @"C:\Users\rajeshkannan\Desktop\RCI Sample Data";
        var txtFiles = Directory.EnumerateFiles(directory, "*.*");
        Console.WriteLine("The text files in the " + directory + " directory are:");
        foreach (string currentfile in txtFiles)
        {
            Console.WriteLine(currentFile);
        }

        var directories = Directory.EnumerateDirectories(directory);
        Console.WriteLine("The text files in the " + directory + " directory are:");
        foreach (string currentdirectory in directories)
        {
            Console.WriteLine(currentdirectory);
        }
    }
}

```

The terminal window at the bottom shows the output of the program's execution:

```

The text files in the /Users/rajeshkannan/Desktop/RCI Sample Data directory are:
/Users/rajeshkannan/Desktop/RCI Sample Data/h1.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/d1.txt
/Users/rajeshkannan/Desktop/RCI Sample Data/d2.txt
The text files in the /Users/rajeshkannan/Desktop/RCI Sample Data directory are:
/Users/rajeshkannan/Desktop/RCI Sample Data/Sub1
/Users/rajeshkannan/Desktop/RCI Sample Data/Sub2

```

The status bar at the bottom indicates a successful build.

2. Create a simple user interface to accept account related information of a customer.[Account class from Lab session on Delegates and Events can be used]. Save the information about the customers in a file using StreamWriter and retrieve the information using StreamReader.

```

using System;
using System.IO;

public delegate void EDisplay();

interface IPrintable
{
    public void Display();
}

class Employee : IPrintable
{
    int EmpNo { get; set; }
    string EmpName { get; set; }
    double Salary { get; set; }
}
```

```
double HRA;
double TA;
double DA;
double PF;
double TDS;
double NetSalary;
double GrossSalary;

public Employee()
{
}

public Employee(int ENO, string ENAME, double SAL)
{
    this.EmpNo = ENO;
    this.EmpName = ENAME;
    this.Salary = SAL;
}

public double CalculateSalary()
{
    if (this.Salary < 5000)
    {
        this.HRA = 0.1 * this.Salary;
        this.TA = 0.05 * this.Salary;
        this.DA = 0.15 * this.Salary;
    }

    else if (this.Salary < 10000)
    {
        this.HRA = 0.15 * this.Salary;
        this.TA = 0.1 * this.Salary;
        this.DA = 0.2 * this.Salary;
    }

    else if (this.Salary < 15000)
    {
        this.HRA = 0.2 * this.Salary;
        this.TA = 0.15 * this.Salary;
        this.DA = 0.25 * this.Salary;
    }

    else if (this.Salary < 20000)
```

```

{
    this.HRA = 0.25 * this.Salary;
    this.TA = 0.2 * this.Salary;
    this.DA = 0.3 * this.Salary;
}

else
{
    this.HRA = 0.3 * this.Salary;
    this.TA = 0.25 * this.Salary;
    this.DA = 0.35 * this.Salary;
}

this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;

this.PF = 0.1 * this.GrossSalary;
this.TDS = 0.18 * this.GrossSalary;
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);

return this.GrossSalary;
}

public void Display()
{
    Console.WriteLine("The Name of the Employee is " + this.EmpName);
    Console.WriteLine("The id of the Employee is " + this.EmpNo);
    Console.WriteLine("The Gross Salary of the employee is " +
Convert.ToString(this.GrossSalary));
    Console.WriteLine("The NET Salary of the employee is " +
Convert.ToString(this.NetSalary));
}

public void Write()
{
    FileStream f = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/EmployeeDetails.txt", FileMode.OpenOrCreate);
    StreamWriter s = new StreamWriter(f);
    s.WriteLine("The name of the employee is "+this.EmpName+". The Id of the employee is
"+Convert.ToString(this.EmpNo)+". The gross Salary is "+Convert.ToString(this.GrossSalary));
    s.Close();
    f.Close();
}

```

```

public void Read()
{
    FileStream f = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/EmployeeDetails.txt", FileMode.OpenOrCreate);
    StreamReader sr = new StreamReader(f);
    string details = sr.ReadLine();
    Console.WriteLine(details);
    sr.Close();
    f.Close();
}

class Manager : Employee
{
    double PetrolAllowance;
    double FoodAllowance;
    double OtherAllowances;

    public Manager() : base()
    {

    }

    public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)
    {
        this.PetrolAllowance = 0.08 * SAL;
        this.FoodAllowance = 0.13 * SAL;
        this.OtherAllowances = 0.03 * SAL;
    }

    public double CalculateSalary()
    {
        double gross = base.CalculateSalary();
        double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance +
this.OtherAllowances;
        return grossAfterAllowances;
    }
}

class MarketExecutive : Employee
{
    double Kilometers;

```

```
double TourAllowances;
int TelephoneAllowances;

public MarketExecutive() : base()
{
}

public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,
ENAME, SAL)
{
    this.TourAllowances = 5 * Km;
    this.TelephoneAllowances = 1000;
}

public double CalculateSalary()
{
    double gross = base.CalculateSalary();
    double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
    return grossAfterAllowances;
}

}

class Assignment2
{
    public static void Main(string[] args)
    {
        Employee E1 = new Employee(1304, "Rajesh", 10000);
        double grossSalary = E1.CalculateSalary();
        E1.Write();
        E1.Read();

    }
}
```



Assignment7

```
Employee > Read()
147    {
148        t
149    }
150
151    public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME, SAL)
152    {
153        this.TourAllowances = 5 * Km;
154        this.TelephoneAllowances = 1000;
155    }
156
157    public double CalculateSalary()
158    {
159        double gross = base.CalculateSalary();
160        double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
161        return grossAfterAllowances;
162    }
163
164
165    class Assignment2
166    {
167        public static void Main(string[] args)
168        {
169            Employee E1 = new Employee(1304, "Rajesh", 10000);
170            double grossSalary = E1.CalculateSalary();
171            E1.Write();
172            E1.Read();
173        }
174    }
175
176}
177
```

The name of the employee is Rajesh. The Id of the employee is 1304. The gross Salary is 16000

0 5 | Build: 0 errors, 5 warnings

3. Make the Employee, MarketingExecutive and Manager class as Serializable
created in LitwareLib.dll .

4. Create a user interface to accept information about Manager(For simplicity accept only employee id , name and basic salary). Serialize the object using Binary Serialization and retrieve its information by deserializing the object.

```
using System;  
  
using System.IO;  
  
using System.Runtime.Serialization.Formatters.Binary;  
  
  
public delegate void EDisplay();  
  
  
interface IPrintable  
{  
    public void Display();  
}  
  
[Serializable]  
  
class Employee : IPrintable  
{  
    int EmpNo { get; set; }  
  
    string EmpName { get; set; }  
  
    double Salary { get; set; }  
  
    double HRA;  
  
    double TA;  
  
    double DA;  
  
    double PF;
```

```
double TDS;  
double NetSalary;  
double GrossSalary;  
  
public Employee()  
{  
  
}  
  
public Employee(int ENO, string ENAME, double SAL)  
{  
    this.EmpNo = ENO;  
    this.EmpName = ENAME;  
    this.Salary = SAL;  
}  
  
public double CalculateSalary()  
{  
    if (this.Salary < 5000)  
    {  
        this.HRA = 0.1 * this.Salary;  
        this.TA = 0.05 * this.Salary;  
        this.DA = 0.15 * this.Salary;  
    }  
}
```

```
else if (this.Salary < 10000)

{
    this.HRA = 0.15 * this.Salary;
    this.TA = 0.1 * this.Salary;
    this.DA = 0.2 * this.Salary;
}
```

```
else if (this.Salary < 15000)

{
    this.HRA = 0.2 * this.Salary;
    this.TA = 0.15 * this.Salary;
    this.DA = 0.25 * this.Salary;
}
```

```
else if (this.Salary < 20000)

{
    this.HRA = 0.25 * this.Salary;
    this.TA = 0.2 * this.Salary;
    this.DA = 0.3 * this.Salary;
}
```

```
else
{
```

```
this.HRA = 0.3 * this.Salary;  
this.TA = 0.25 * this.Salary;  
this.DA = 0.35 * this.Salary;  
}  
  
this.GrossSalary = this.Salary + this.HRA + this.TA + this.DA;  
  
this.PF = 0.1 * this.GrossSalary;  
this.TDS = 0.18 * this.GrossSalary;  
this.NetSalary = this.GrossSalary - (this.PF + this.TDS);  
  
return this.GrossSalary;
```

```
}
```

```
public void Display()  
{  
    Console.WriteLine("The Name of the Employee is " + this.EmpName);  
    Console.WriteLine("The id of the Employee is " + this.EmpNo);  
    Console.WriteLine("The Gross Salary of the employee is " +  
        Convert.ToString(this.GrossSalary));  
    Console.WriteLine("The NET Salary of the employee is " +  
        Convert.ToString(this.NetSalary));  
}
```

```
public void Write()
{
    FileStream f = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/EmployeeDetails.txt", FileMode.OpenOrCreate);

    StreamWriter s = new StreamWriter(f);

    s.WriteLine("The name of the employee is "+this.EmpName+. The Id of the employee is
"+Convert.ToString(this.EmpNo)+". The gross Salary is "+Convert.ToString(this.GrossSalary));

    s.Close();
}
```

```
public void Read()
{
    FileStream f = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/EmployeeDetails.txt", FileMode.OpenOrCreate);

    StreamReader sr = new StreamReader(f);

    string details = sr.ReadLine();

    Console.WriteLine(details);

    sr.Close();
}

}
```

```
public void Serialize()
{
    FileStream s = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/Object.txt", FileMode.OpenOrCreate);
```

```
BinaryFormatter f = new BinaryFormatter();

f.Serialize(s, this);

s.Close();

}

public Employee Deserialize()

{

    FileStream s = new FileStream(@"/Users/rajeshkannan/Desktop/RCI Sample
Data/Object.txt", FileMode.OpenOrCreate);

    BinaryFormatter f = new BinaryFormatter();

    Employee obj = (Employee) f.Deserialize(s);

    s.Close();

    return obj;

}

}

[Serializable]

class Manager : Employee

{

    double PetrolAllowance;

    double FoodAllowance;

    double OtherAllowances;
```

```
public Manager() : base()  
{  
  
}  
  
public Manager(int ENO, string ENAME, double SAL) : base(ENO, ENAME, SAL)  
{  
    this.PetrolAllowance = 0.08 * SAL;  
    this.FoodAllowance = 0.13 * SAL;  
    this.OtherAllowances = 0.03 * SAL;  
}  
  
public double CalculateSalary()  
{  
    double gross = base.CalculateSalary();  
  
    double grossAfterAllowances = gross + this.PetrolAllowance + this.FoodAllowance +  
        this.OtherAllowances;  
  
    return grossAfterAllowances;  
}  
  
}  
  
[Serializable]  
  
class MarketExecutive : Employee  
{  
    double Kilometers;  
  
    double TourAllowances;
```

```
int TelephoneAllowances;

public MarketExecutive() : base()

{

}

public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO,
ENAME, SAL)

{

    this.TourAllowances = 5 * Km;

    this.TelephoneAllowances = 1000;

}

public double CalculateSalary()

{

    double gross = base.CalculateSalary();

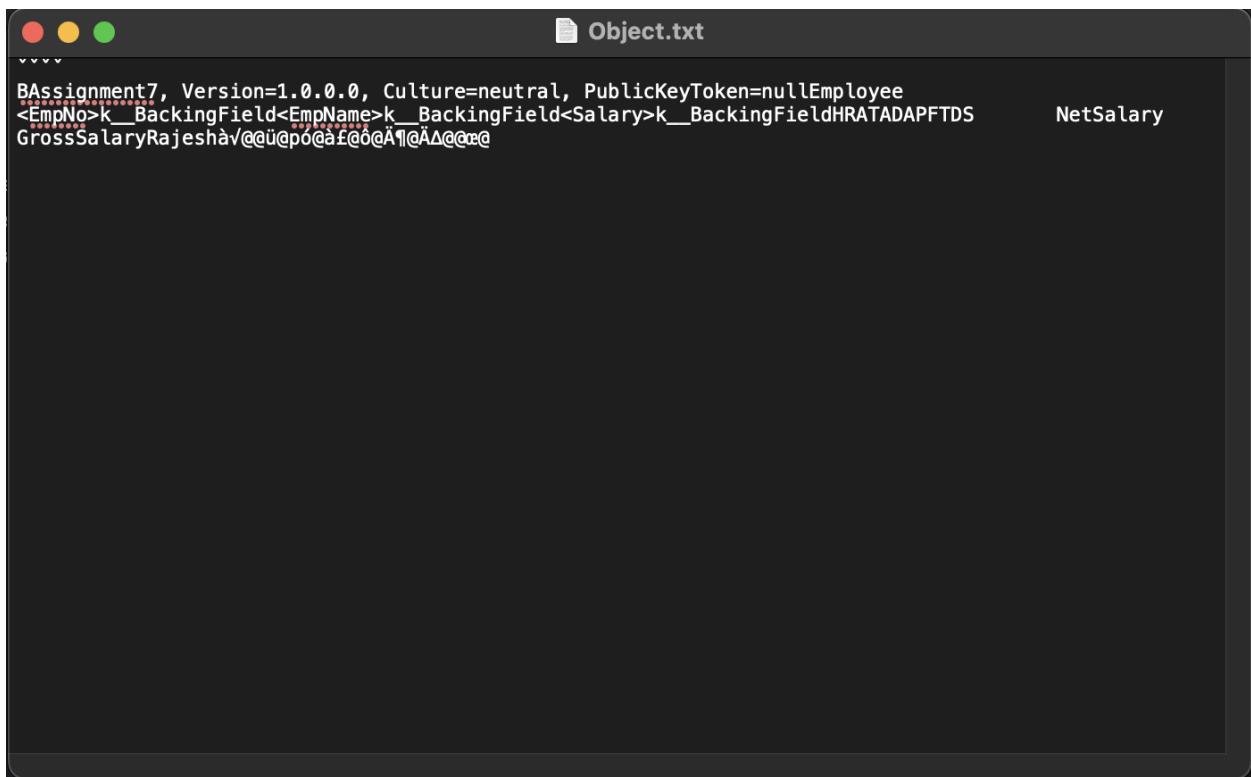
    double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;

    return grossAfterAllowances;

}

class Assignment2
```

```
{  
    public static void Main(string[] args)  
    {  
        Employee E1 = new Employee(1304, "Rajesh", 10000);  
  
        double grossSalary = E1.CalculateSalary();  
  
        E1.Serialize();  
  
        Employee DeserializedObj = E1.Deserialize();  
  
        DeserializedObj.Display();  
    }  
}
```



The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure under "Assignment7".
- Code Editor:** Displays `Program.cs` containing C# code for a class hierarchy. The code includes methods for calculating salary and serializing/deserializing objects.
- Terminal - Assignment7:** Shows the following output:

```
The Name of the Employee is Rajesh
The id of the Employee is 1304
The Gross Salary of the employee is 16000
The NET Salary of the employee is 11520
```
- Status Bar:** Shows build statistics: 0 errors, 7 warnings.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus are visible along the top.