**C#**
**Assignment - 1**

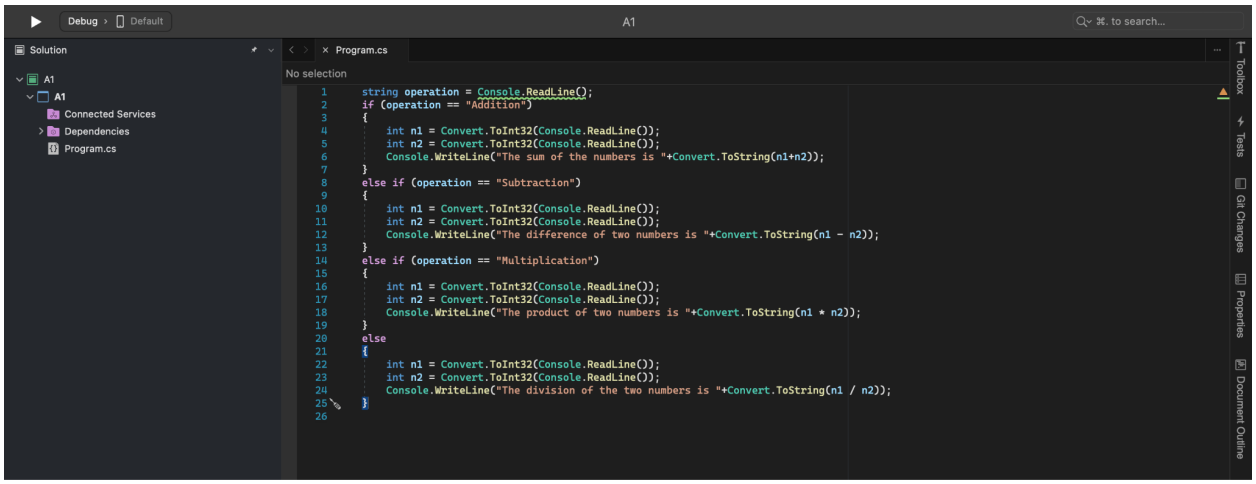
1. Write a Simple console Application Calculator with the help of Visual Studio .NET IDE which will perform following operations on two numbers:

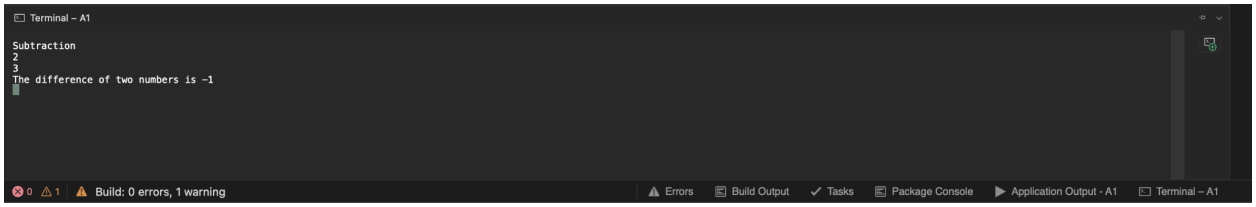
- Addition.
- Subtraction.
- Multiplication.
- Division

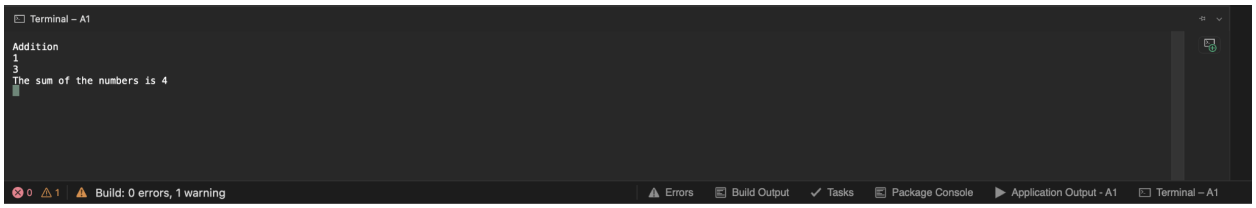
Accept input from the user and display results on the console. Make use of loops, switch case wherever required.

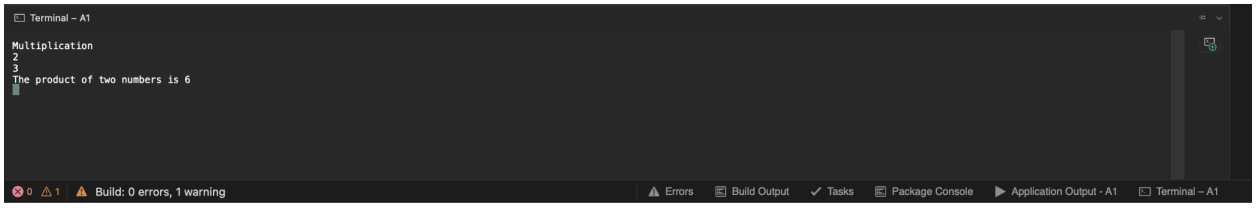
```
string operation = Console.ReadLine();
if (operation == "Addition")
{
   int n1 = Convert.ToInt32(Console.ReadLine());
   int n2 = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("The sum of the numbers is "+Convert.ToString(n1+n2));
}
else if (operation == "Subtraction")
{
   int n1 =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

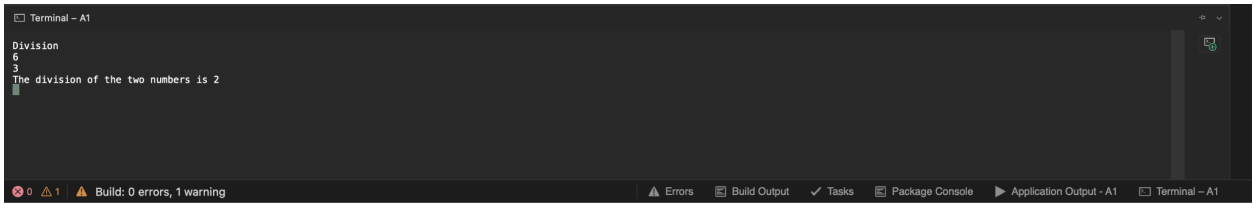
```
    int n1 = Convert.ToInt32(Console.ReadLine());
    int n2 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("The division of the two numbers is "+Convert.ToString(n1 / n2));
}
```



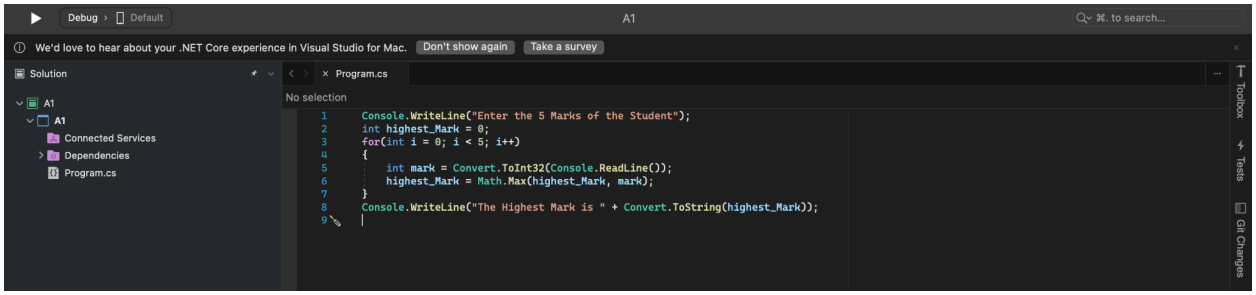



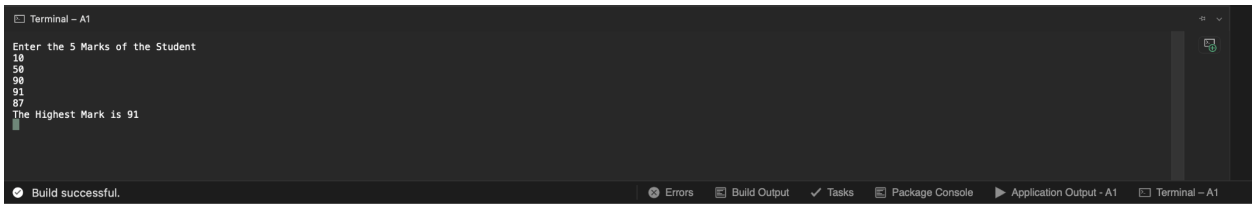



2. Accept average marks of five students. Display the highest marks obtained.

```
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));
```

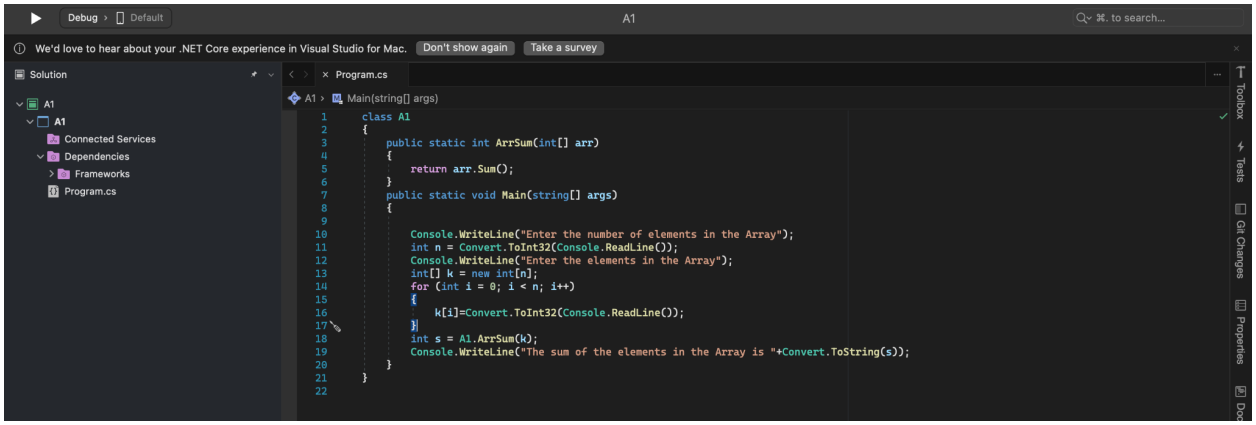


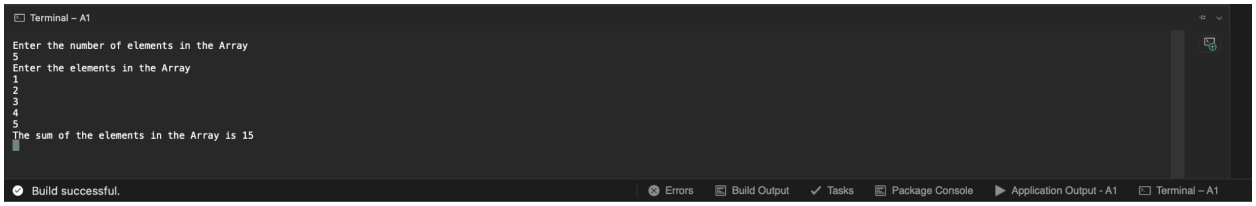




3. Write a static method to accept a param array of integers. The method should find the sum of all the integers passed and display the result. Write a client program to call the method.

```
class A1

{

   public static int ArrSum(int[] arr)

   {

      return arr.Sum();
```

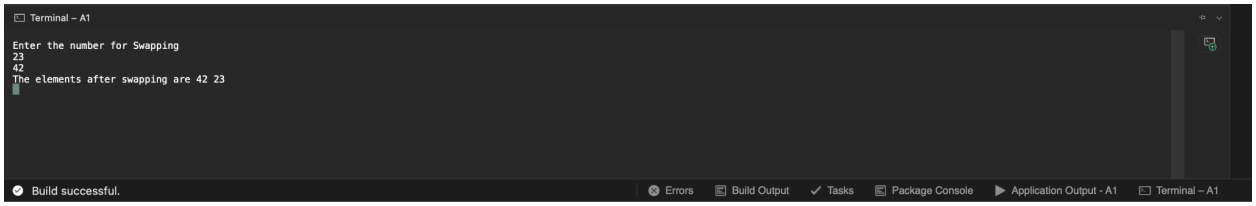
```
    }
    public static void Main(string[] args)
    {

        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));
    }
}
```





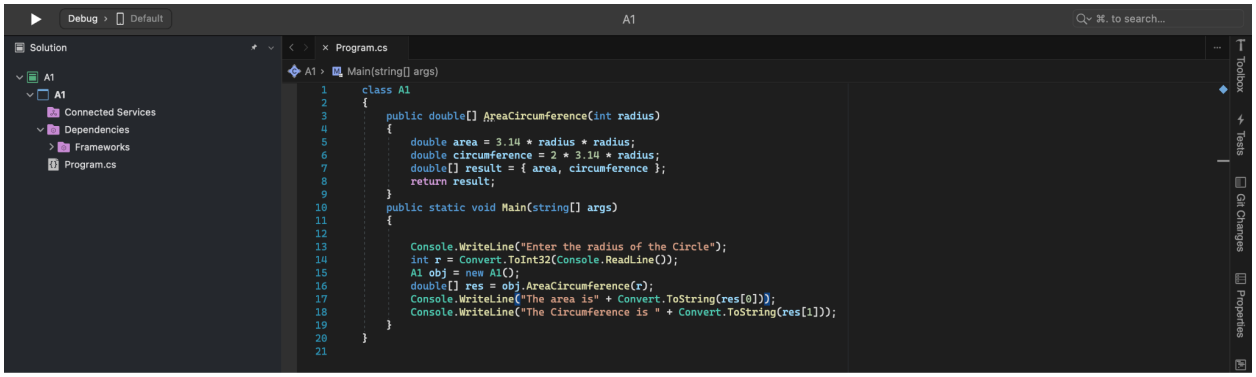

4. Write a method to swap two integers. The client code should call the method and print the swapped value.

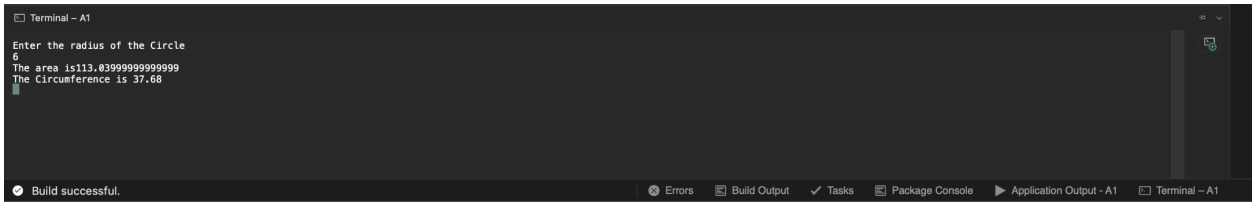
```
class A1
{
   public int[] ArrSum(int[] arr)
   {
      int temp = arr[0];
      arr[0] = arr[1];
      arr[1] = temp;
      return arr;
   }
   public static void Main(string[] args)
   {
      Console.WriteLine("Enter the number for Swapping");
      int n1 = Convert.ToInt32(Console.ReadLine());
      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]));
   }
}
```




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]));
   }
}
```






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

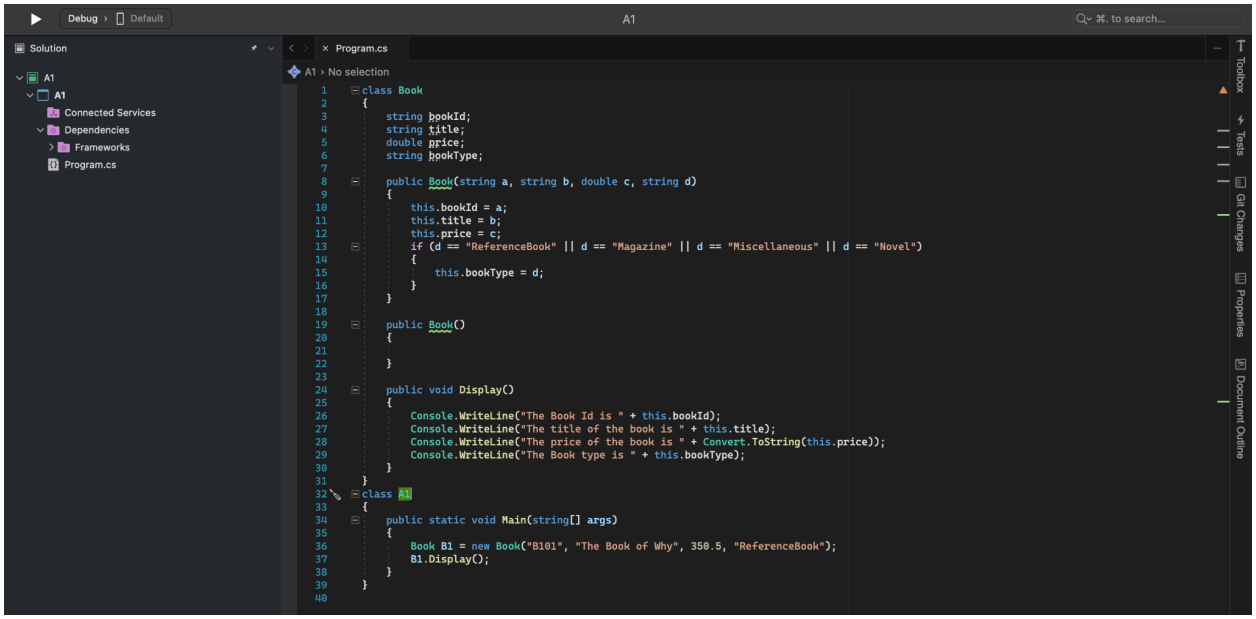
Display the details of the book. The type of book should be displayed as a string e.g.:`Magazine`

Note: Use methods for accepting and displaying details.

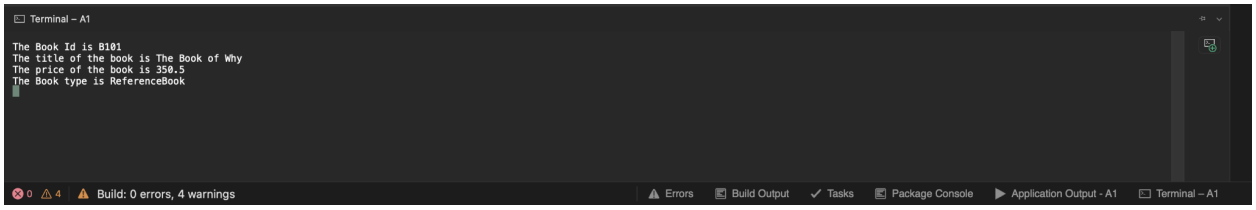
```
class Book
{
    string bookId;
    string title;
    double price;
    string bookType;

    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();
```

```
    }
}
```




```
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();
    }
}
```




**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()”

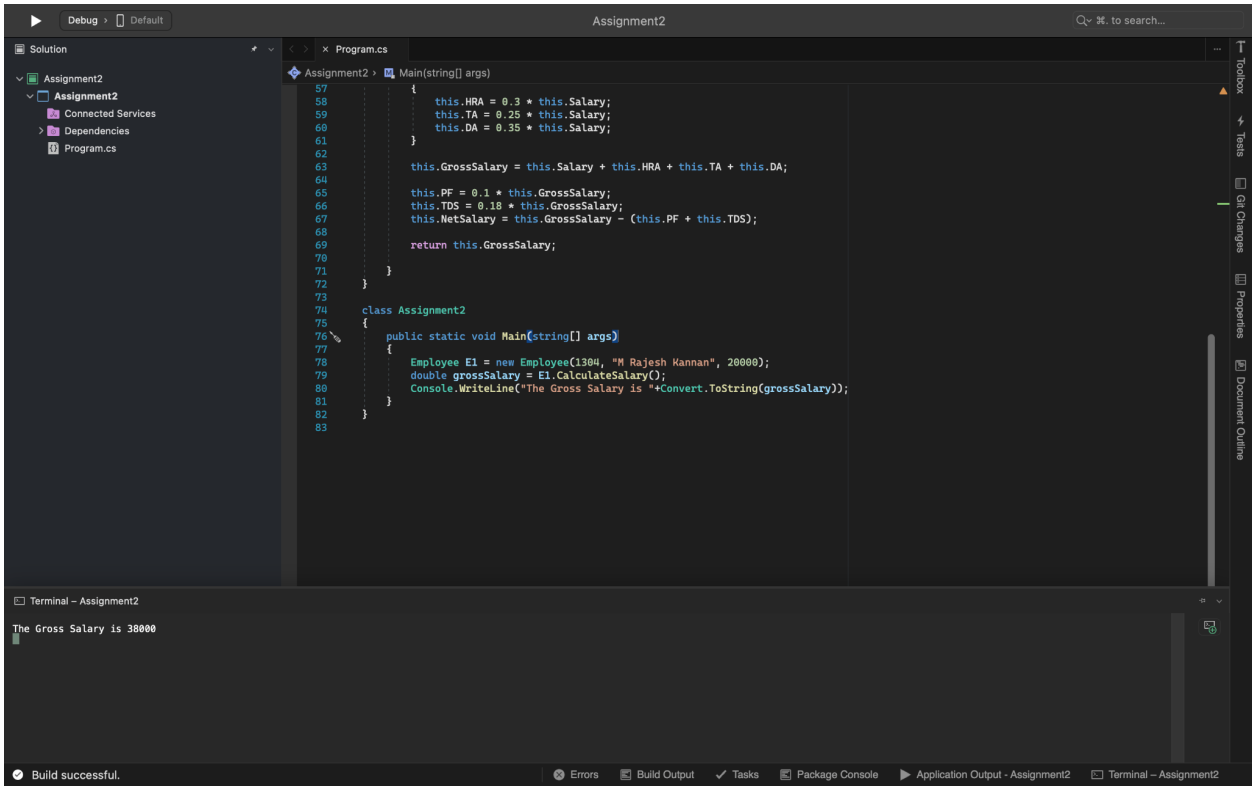
PF = 10 % of GrossSalary. TDS = 18 % of GrossSalary.
NetSalary = GrossSalary – (PF + TDS).

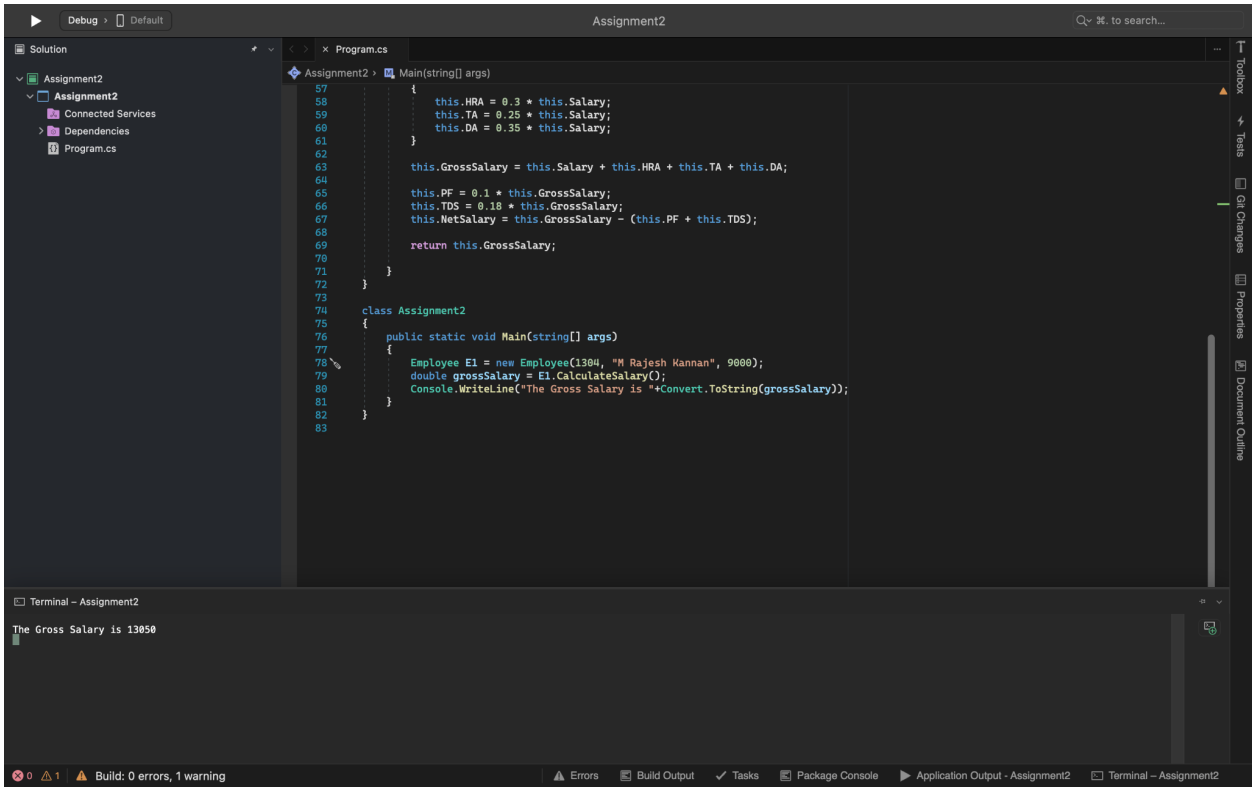
Write a console application Employee Management which allow HR staff member to register newly joined employee with EmpNo, EmpName and Salary. Display gross salary of employee on console. LitwareLib class Library will be used in Test console application for creating objects and invoking functionality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


Debug
Default
Assignment2
Solution
Assignment2
Assignment2
Connected Services
Dependencies
Program.cs
Program.cs
Assignment2 › Main(string[] args)
{
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));
}
}
Terminal – Assignment2
The Gross Salary is 30000
Build successful.
Errors
Build Output
Tasks
Package Console
Application Output - Assignment2
Terminal – Assignment2
Toolbox
Tests
Git Changes
Properties
Document Outline


Debug
Default
Assignment2
Solution
Assignment2
Assignment2
Connected Services
Dependencies
Program.cs
Program.cs
Assignment2 › Main(string[] args)
{
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", 9000);
double grossSalary = E1.CalculateSalary();
Console.WriteLine("The Gross Salary is "+Convert.ToString(grossSalary));
}
}
Terminal – Assignment2
The Gross Salary is 13050
0
1
Build: 0 errors, 1 warning
Errors
Build Output
Tasks
Package Console
Application Output - Assignment2
Terminal – Assignment2
Toolbox
Tests
Git Changes
Properties
Document Outline

## 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.

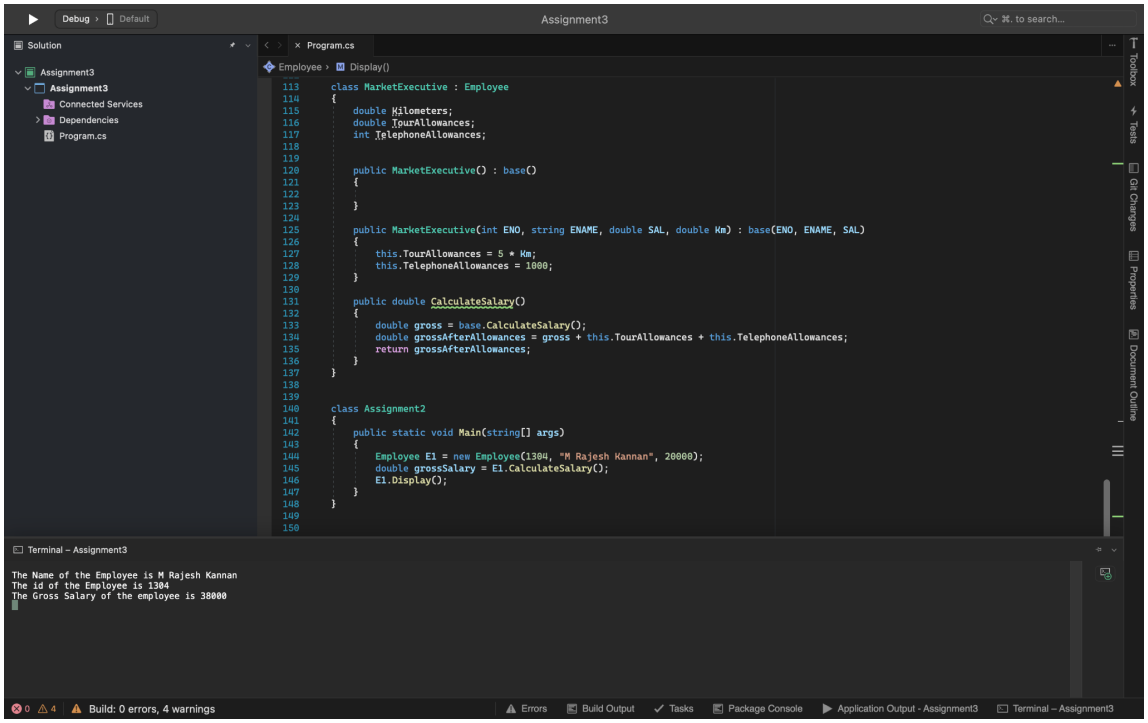
```
interface IPrintable
{
   public void Display();
}

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;
```

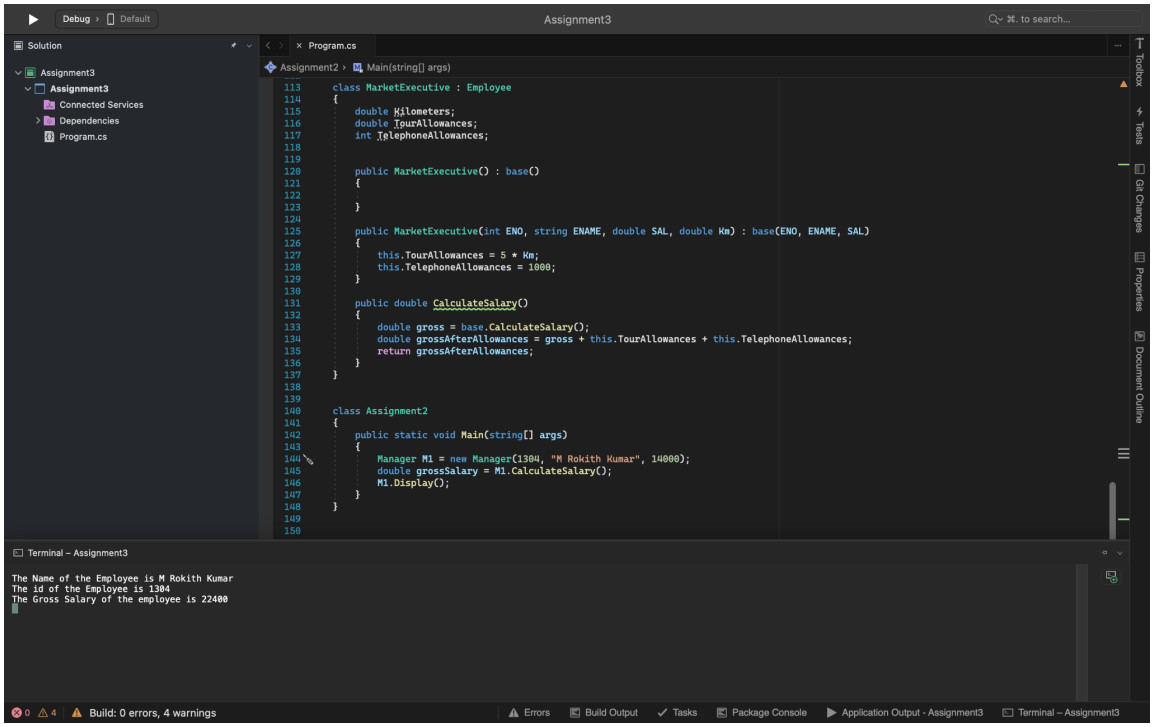
```
public Employee()
{

}

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
   {
```

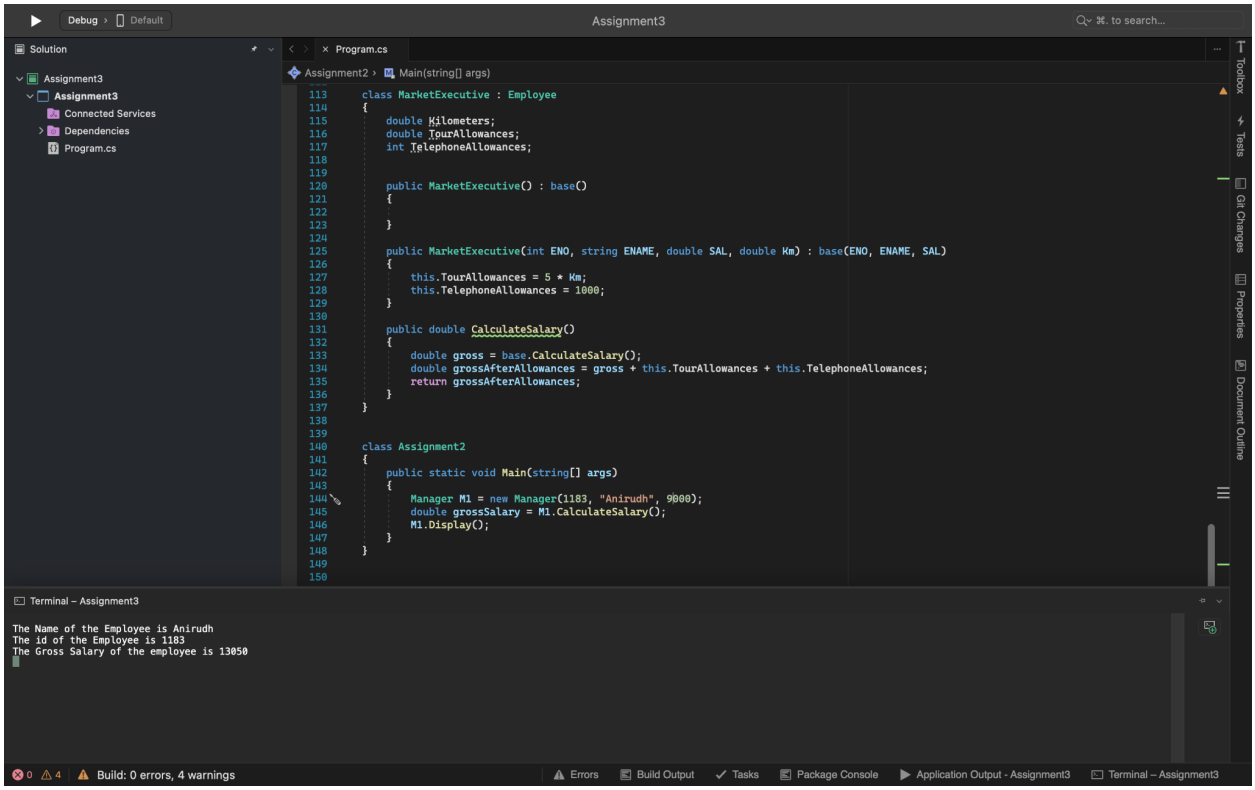
```
            this.HRA = 0.3 * this.Salary;
            this.TA = 0.25 * this.Salary;
            this.DA = 0.35 * this.Salary;
        }

        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();
    }
```

}


Debug
Default
Assignment3
Solution
Assignment3
Assignment3
Connected Services
Dependencies
Program.cs
Program.cs
Employee › Display()
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, "M Rajesh Kannan", 20000);
double grossSalary = E1.CalculateSalary();
E1.Display();
}
}
Terminal – Assignment3
The Name of the Employee is M Rajesh Kannan
The id of the Employee is 1304
The Gross Salary of the employee is 38000
Build: 0 errors, 4 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment3
Terminal – Assignment3
Toolbox
Tests
Git Changes
Properties
Document Outline


Debug
Default
Assignment3
Solution
Assignment3
Assignment3
Connected Services
Dependencies
Program.cs
Program.cs
Assignment2 › Main(string[] args)
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(1304, "M Rokith Kumar", 14000);
double grossSalary = M1.CalculateSalary();
M1.Display();
}
}
Terminal – Assignment3
The Name of the Employee is M Rokith Kumar
The id of the Employee is 1304
The Gross Salary of the employee is 22400
Build: 0 errors, 4 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment3
Terminal – Assignment3
Toolbox
Tests
Git Changes
Properties
Document Outline


Debug
Default
Assignment3
⌘. to search...
Solution
Assignment3
Assignment3
Connected Services
Dependencies
Program.cs
Program.cs
Assignment2 › Main(string[] args)
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();
}
}
Terminal – Assignment3
The Name of the Employee is Anirudh
The id of the Employee is 1183
The Gross Salary of the employee is 13050
Build: 0 errors, 4 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment3
Terminal – Assignment3
Toolbox
Tests
Git Changes
Properties
Document Outline

## 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.

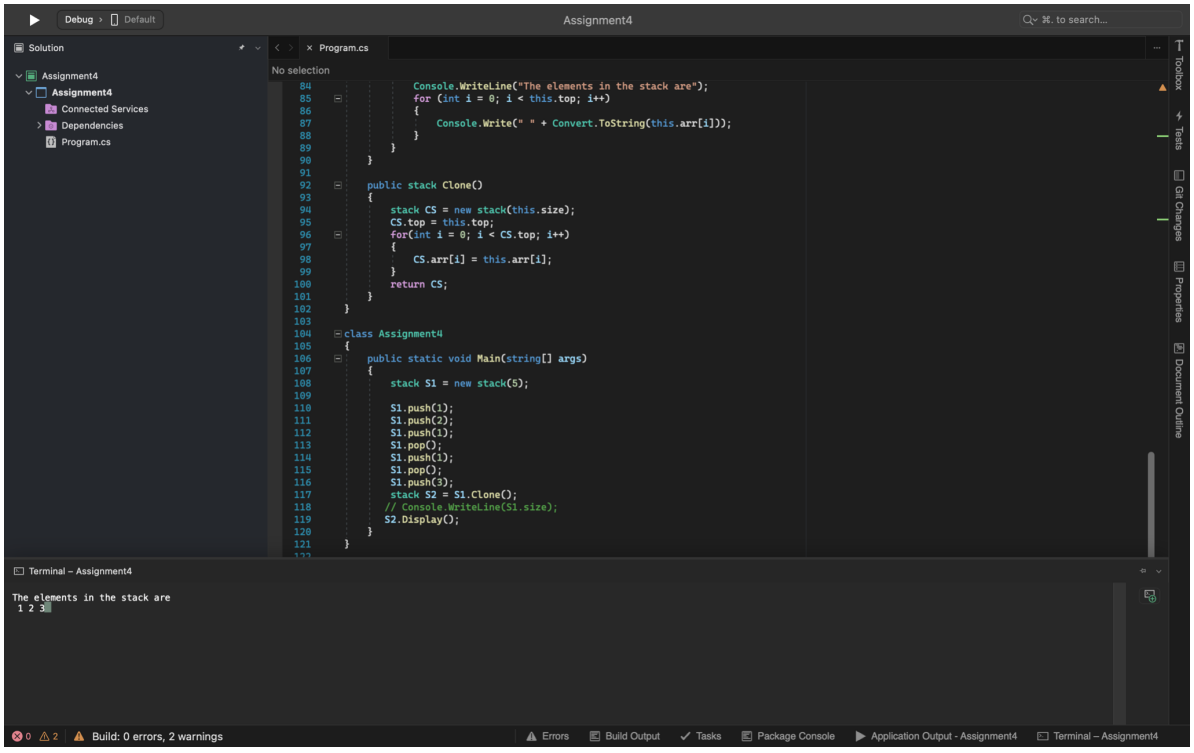
Implement Push() and Pop() operation. Implement ICloneable interface to perform cloning. Write a client application to perform cloning.

```
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()
   {
```

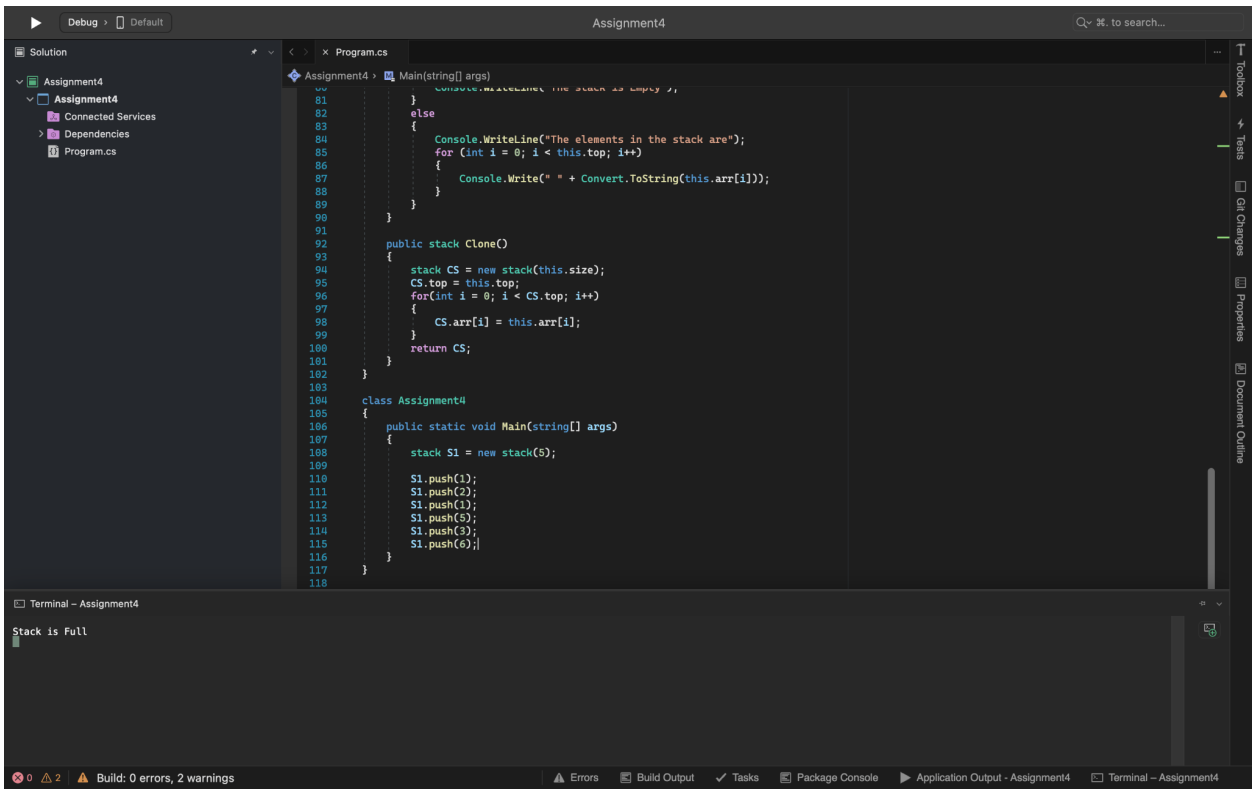
```
}

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");
      }
```

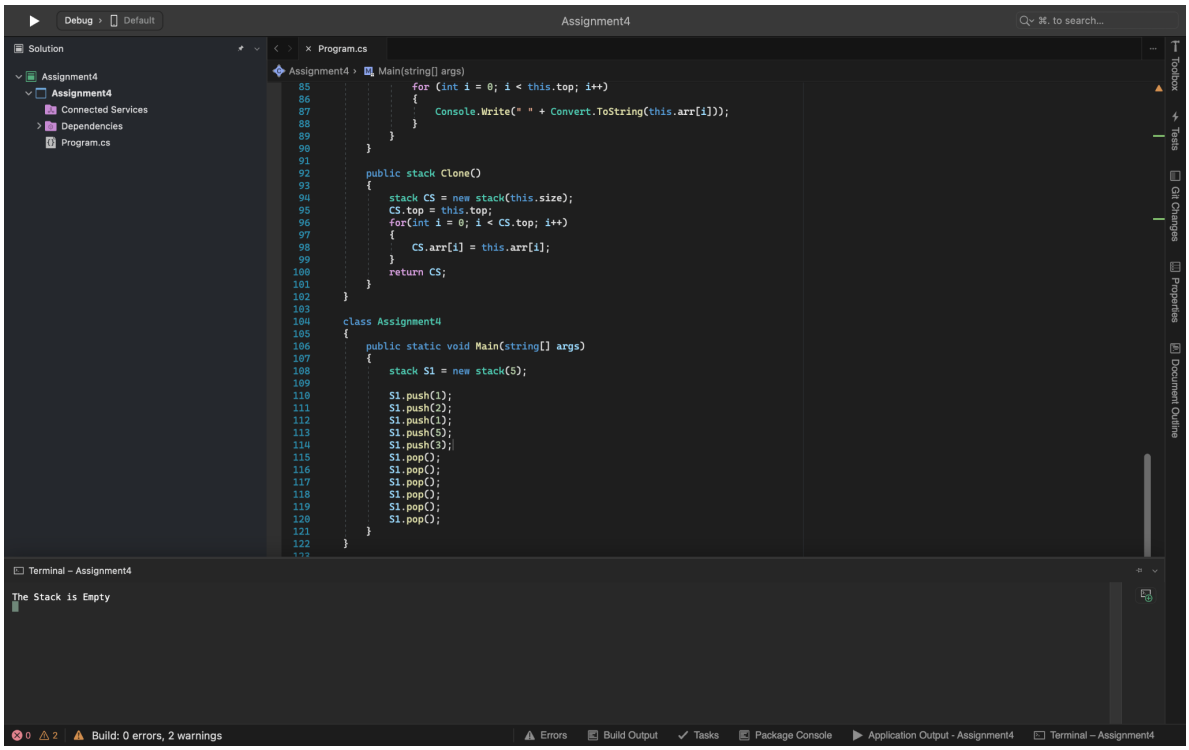
```
        catch(StackException SE)
        {
            Console.WriteLine(SE.ErrorMessage);
        }
    }
    else
    {
        this.top--;
    }
}

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();
    }
}
```



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.



## Assignment - 5

1. Create following types of arrays

    a. Integer
    b. String

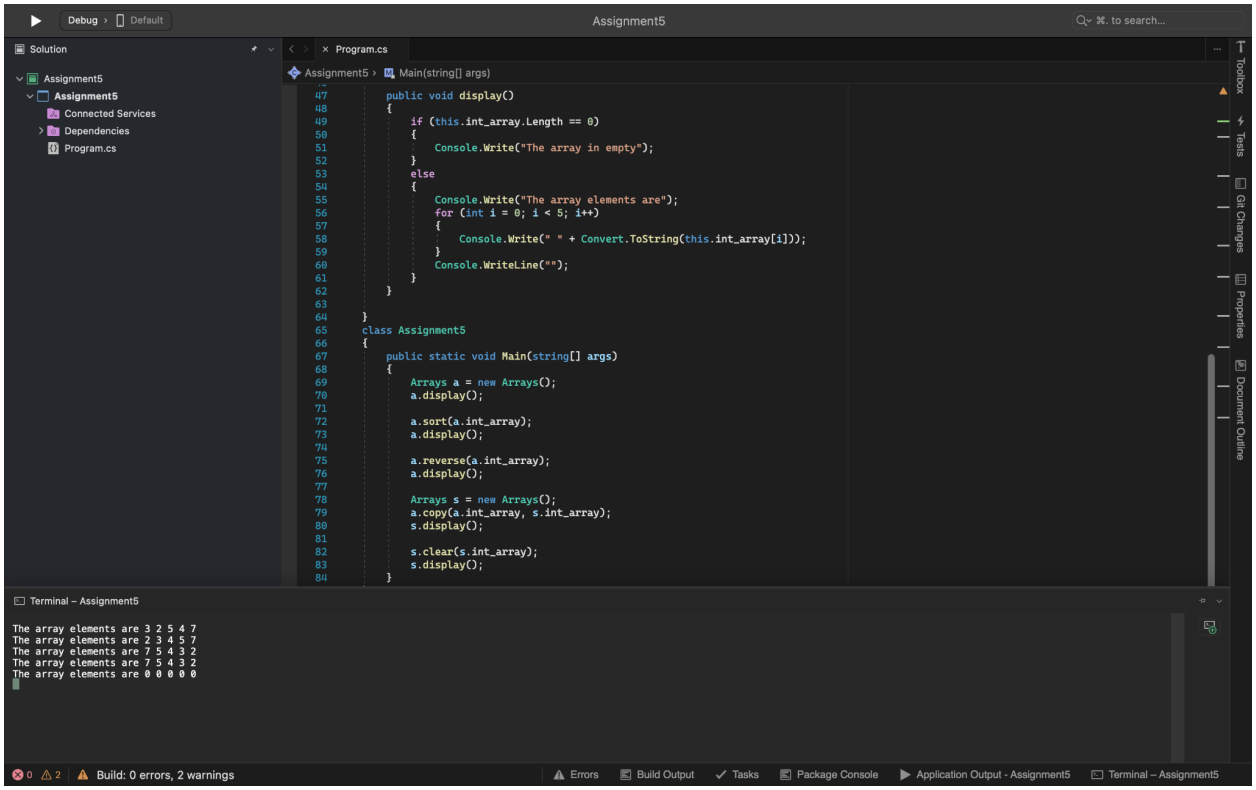
Use System.Array class to perform following operations on them

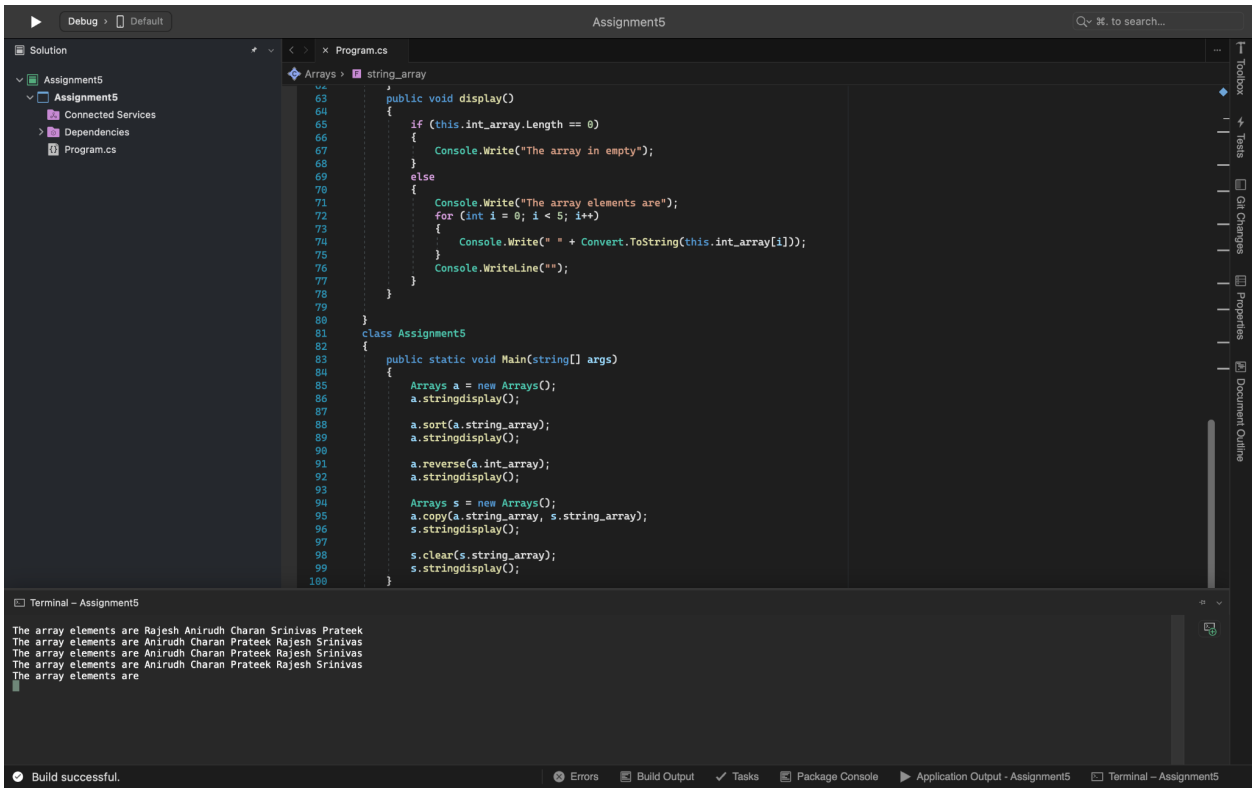
Copy, Sort, Clear, Reverse

```
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.Write("The array in 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();
    }
}
```


Debug
Default
Assignment5
⌘. to search...
Solution
Assignment5
Assignment5
Connected Services
Dependencies
Program.cs
Program.cs
Assignment5 › Main(string[] args)
public void display()
{
if (this.int_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.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();
}
Terminal – Assignment5
The array elements are 3 2 5 4 7
The array elements are 2 3 4 5 7
The array elements are 7 5 4 3 2
The array elements are 7 5 4 3 2
The array elements are 0 0 0 0 0
Build: 0 errors, 2 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment5
Terminal – Assignment5
Toolbox
Tests
Git Changes
Properties
Document Outline


Debug
Default
Assignment5
⌘. to search...
Solution
Assignment5
Assignment5
Connected Services
Dependencies
Program.cs
Program.cs
Arrays › string_array
public void display()
{
if (this.int_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.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();
}
Terminal – Assignment5
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.
Errors
Build Output
Tasks
Package Console
Application Output - Assignment5
Terminal – Assignment5
Toolbox
Tests
Git Changes
Properties
Document Outline

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)
```

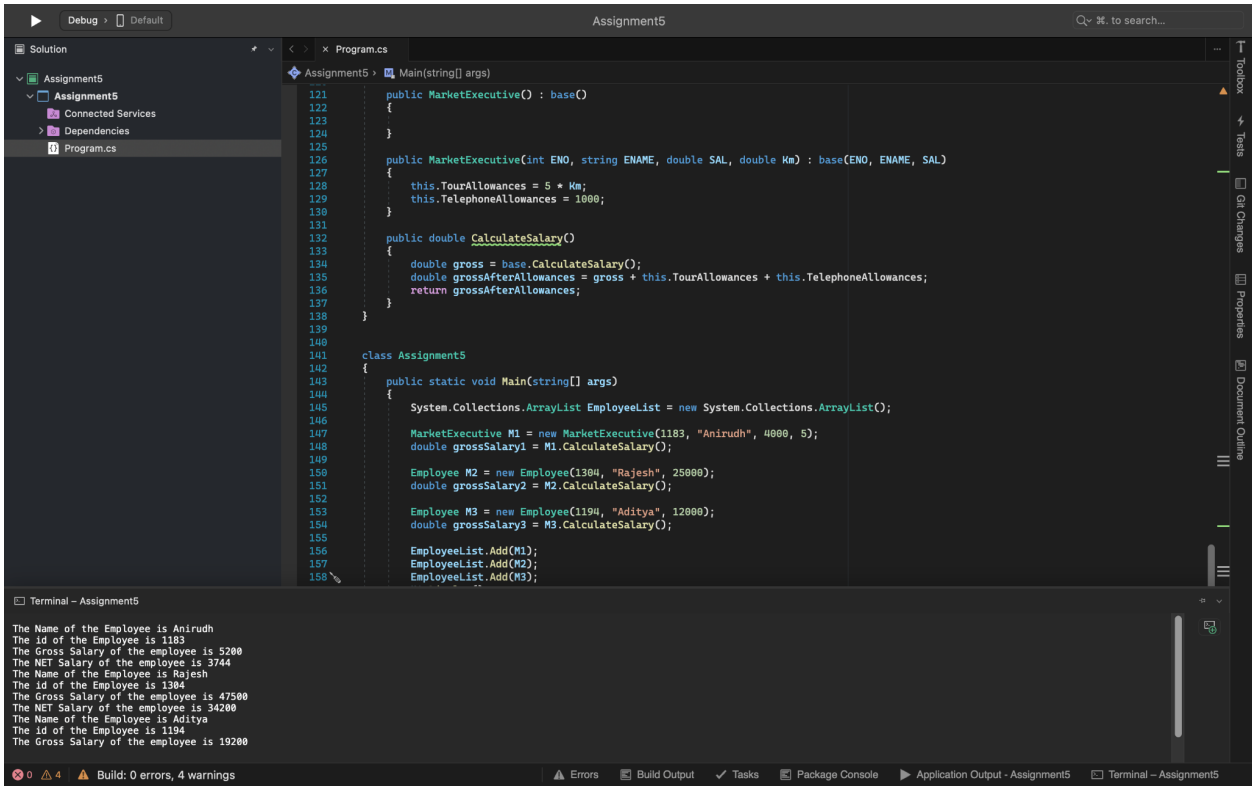
```
{
    this.HRA = 0.2 * this.Salary;
    this.TA = 0.15 * this.Salary;
    this.DA = 0.25 * this.Salary;
}

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();
          }
    }
}
```




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:

a. Add a new employee
b. Display the list of employees.
c. 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();

    }
```

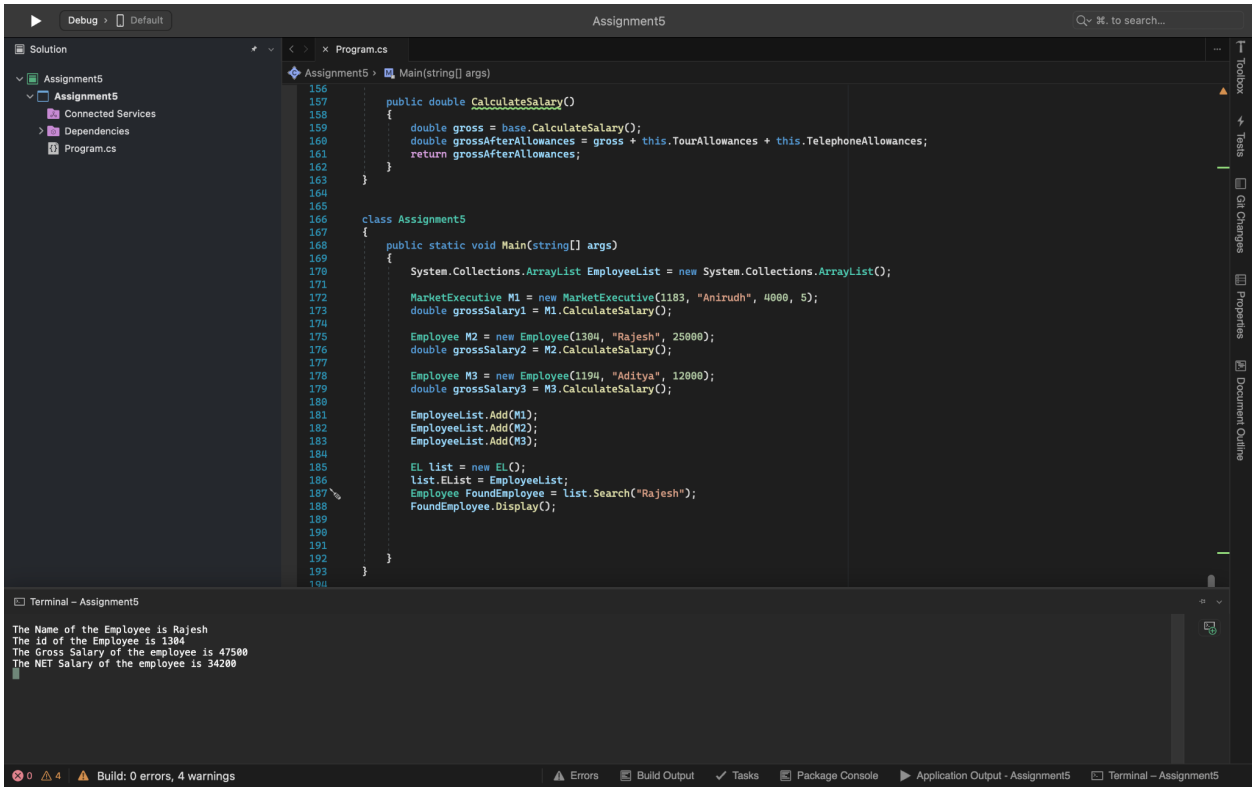
```
    public Employee Search(string name)
    {
        foreach(Employee Emp in this.EList)
        {
            if (Emp.EmpName == name)
            {
                return Emp;
            }
        }
        return new Employee(0, "NULL", 0);
    }
}

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();


    }
}
```



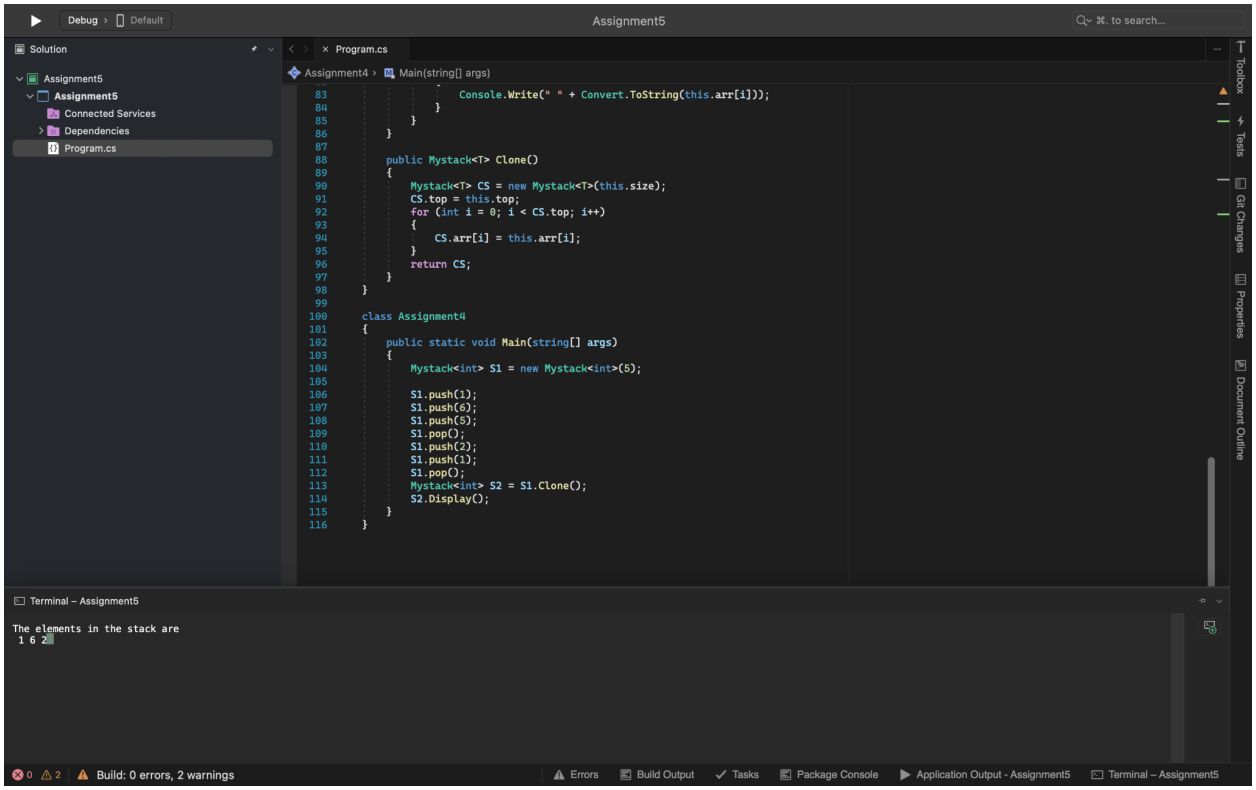
4. Write Custom Generic class MyStack based on assignment of previous session, with

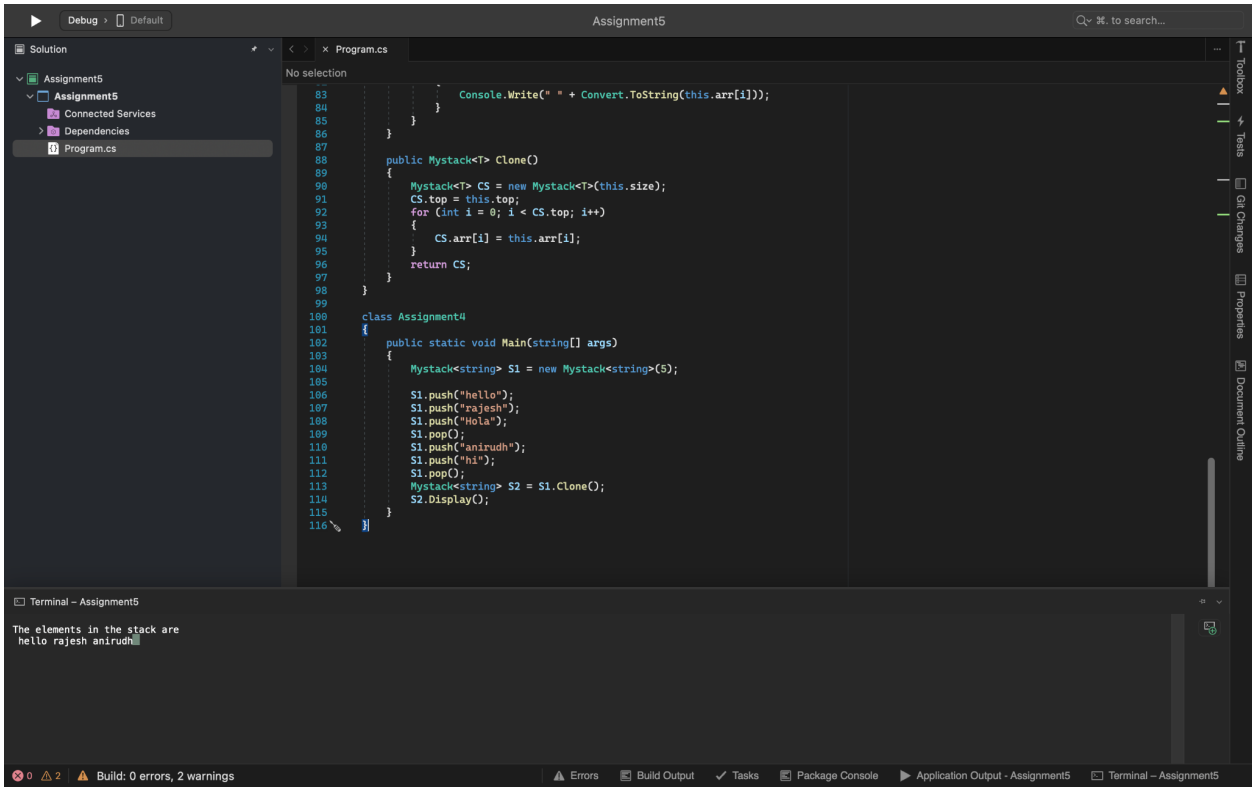
Push() and Pop() methods to store any kind of .NET Type.

```
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();
    }
}
```


Debug
Default
Assignment5
Solution
Assignment5
Assignment5
Connected Services
Dependencies
Program.cs
Program.cs
Assignment4 › Main(string[] args)
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<int> S1 = new Mystack<int>(5);
S1.push(1);
S1.push(6);
S1.push(5);
S1.pop();
S1.push(2);
S1.push(1);
S1.pop();
Mystack<int> S2 = S1.Clone();
S2.Display();
}
}
Terminal – Assignment5
The elements in the stack are
1 6 2
Build: 0 errors, 2 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment5
Terminal – Assignment5
Toolbox
Tests
Git Changes
Properties
Document Outline

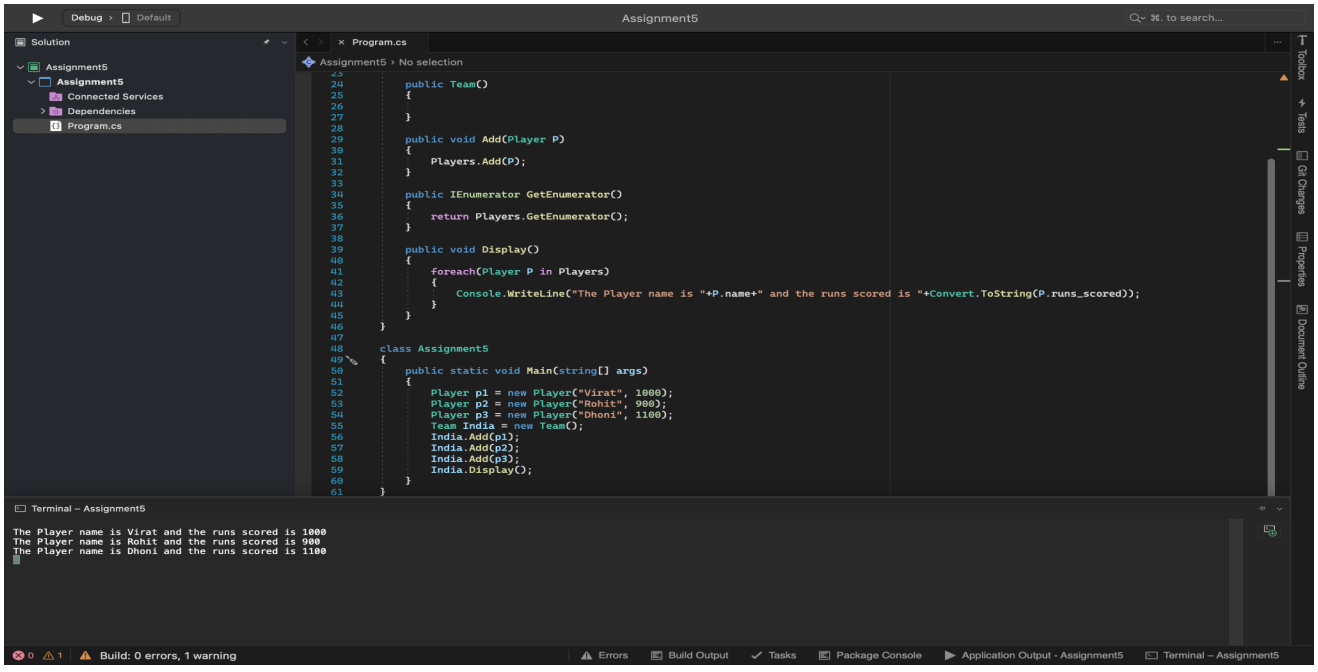

Debug
Default
Assignment5
Solution
Assignment5
Assignment5
Connected Services
Dependencies
Program.cs
Program.cs
No selection
Console.Write(" " + Convert.ToString(this.arr[i]));
}
}
}
public Mystack<T> Clone()
{
Mystack<T> CS = new Mystack<T>(this.size);
CS.top = this.top;
for (int i = 0; i < CS.top; 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();
}
}
Terminal – Assignment5
The elements in the stack are
hello rajesh anirudh
Build: 0 errors, 2 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment5
Terminal – Assignment5
Toolbox
Tests
Git Changes
Properties
Document Outline

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();
    }
}
```

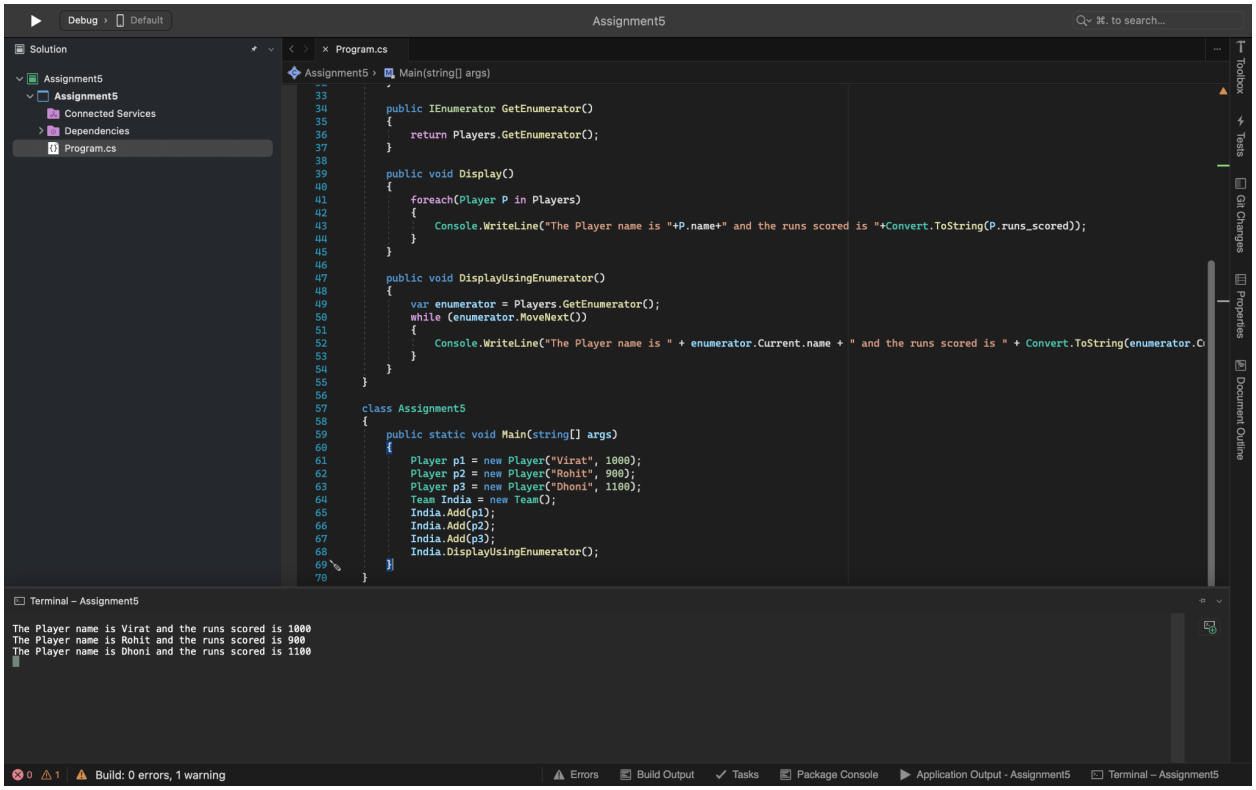


7. Use an iterator to iterate through the players in the above example.

```
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();
    }
}
```


Debug
Default
Assignment5
Solution
Assignment5
Assignment5
Connected Services
Dependencies
Program.cs
Program.cs
Assignment5 › Main(string[] args)
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.C
}
}
}
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();
}
}
Terminal – Assignment5
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
Build: 0 errors, 1 warning
Errors
Build Output
Tasks
Package Console
Application Output - Assignment5
Terminal – Assignment5
Toolbox
Tests
Git Changes
Properties
Document Outline

**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;
```

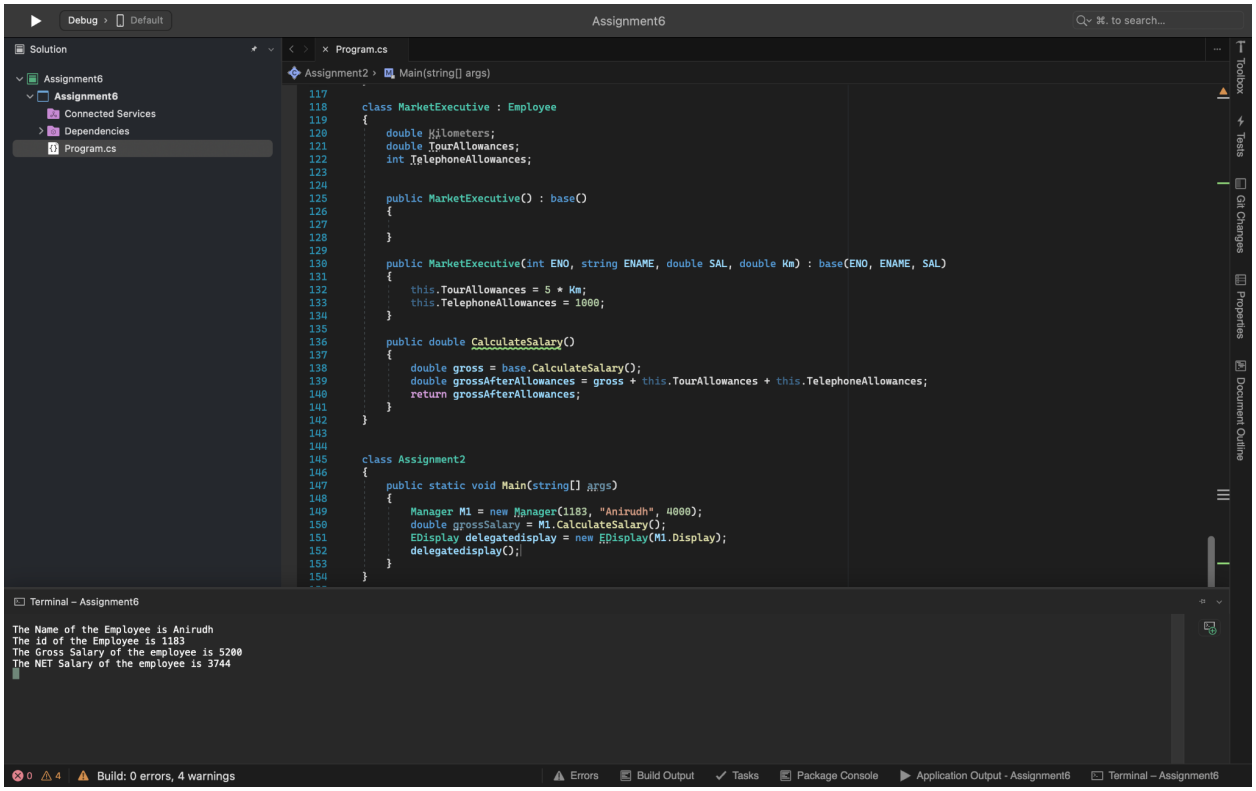
```
}

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 delegatedisplay = new EDisplay(M1.Display);
        delegatedisplay();
    }
}
```



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;
```

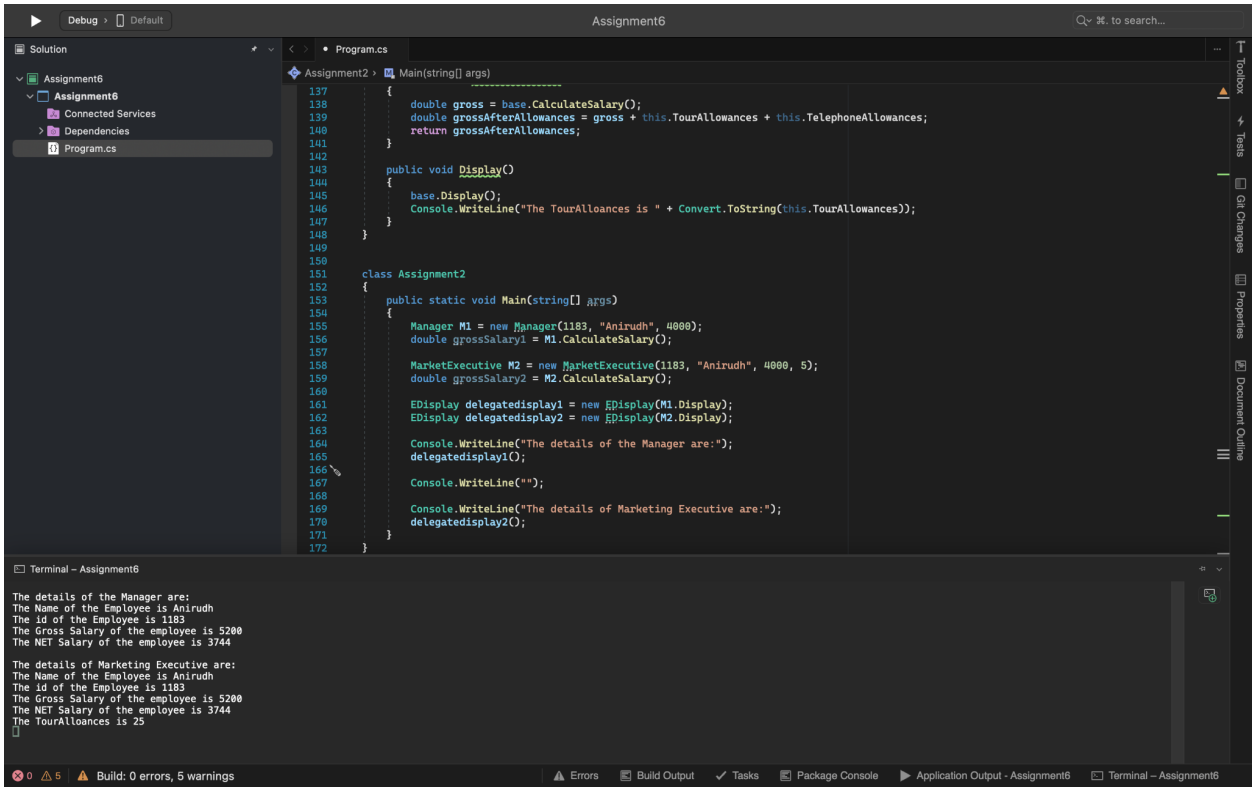
```
        this.PF = 0.1 * this.GrossSalary;
        this.TDS = 0.18 * this.GrossSalary;
        this.NetSalary = this.GrossSalary - (this.PF + this.TDS);

        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 TourAlloances 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();

    }

}
```

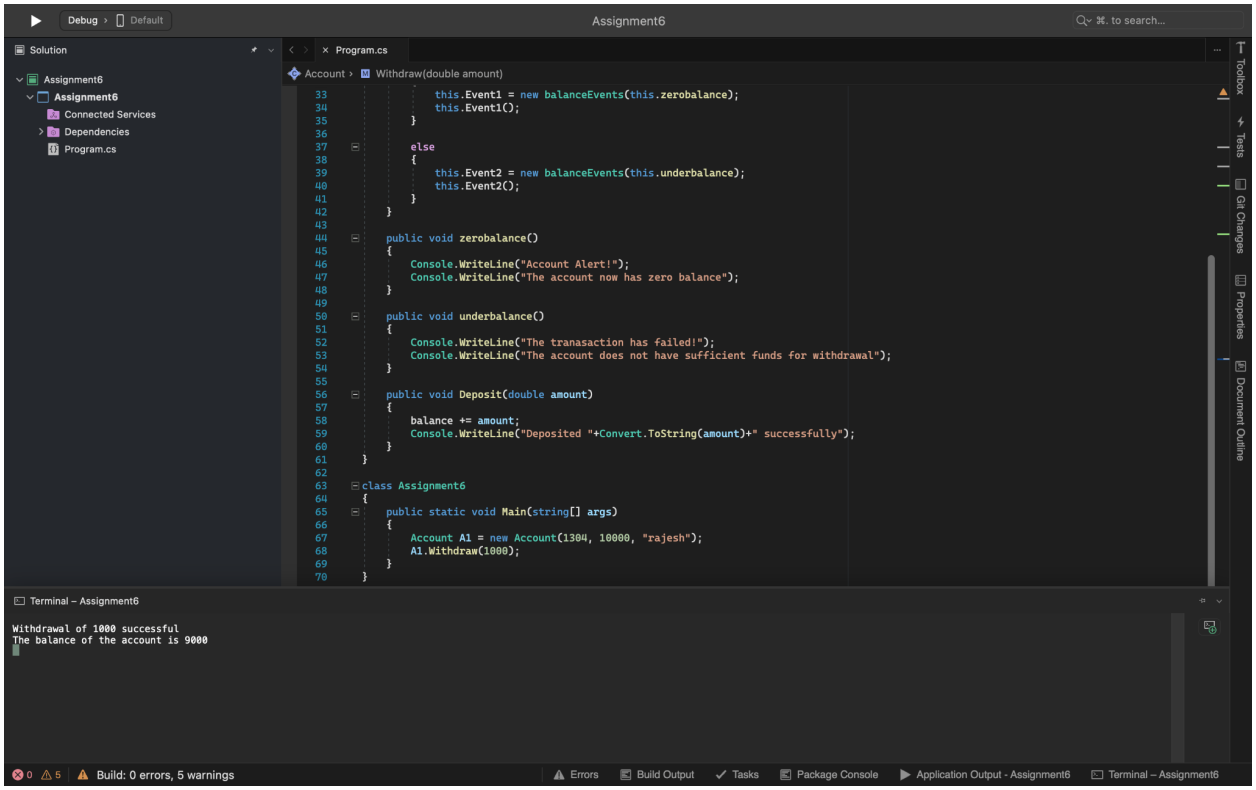


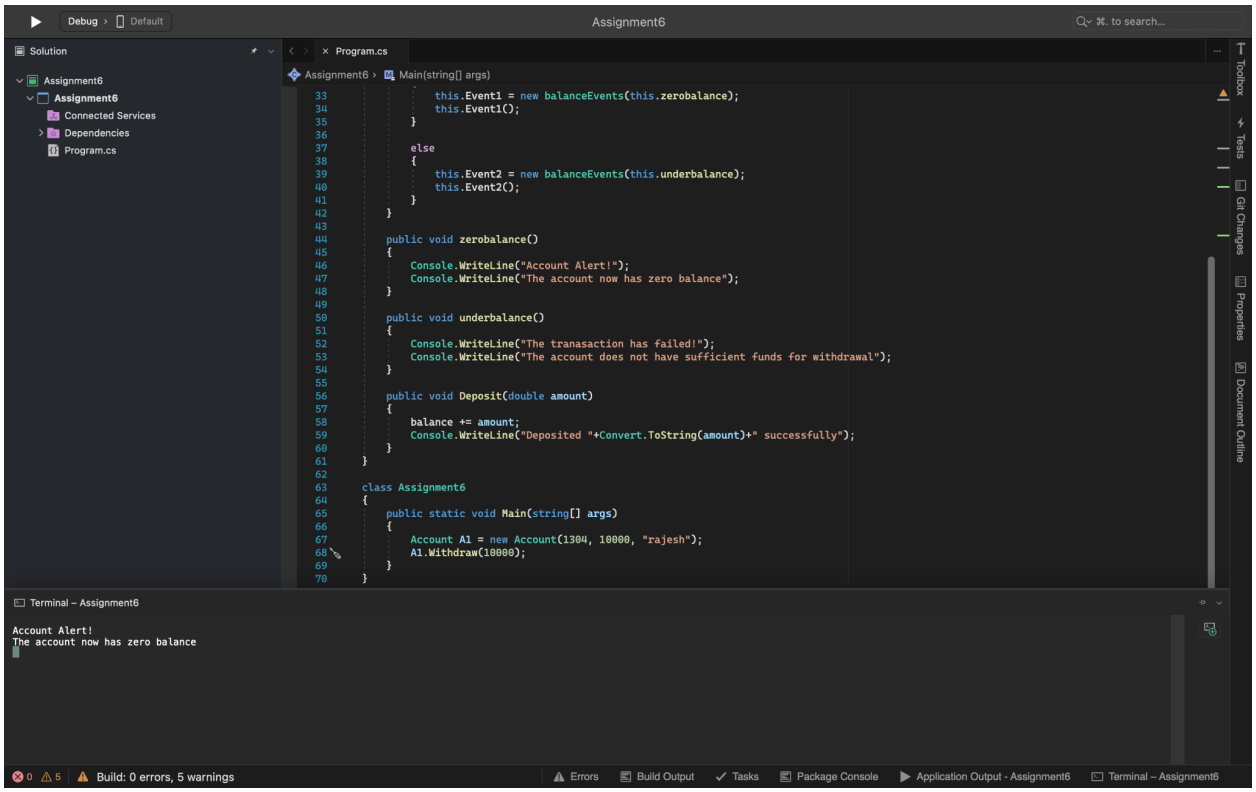
3. Write a console Application for banking domain, having Account class with data members as account number, customer name, balance . It should have Withdraw and Deposit methods for performing banking transaction. It should also define UnderBalance, BalanceZero events. These events would be raised when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

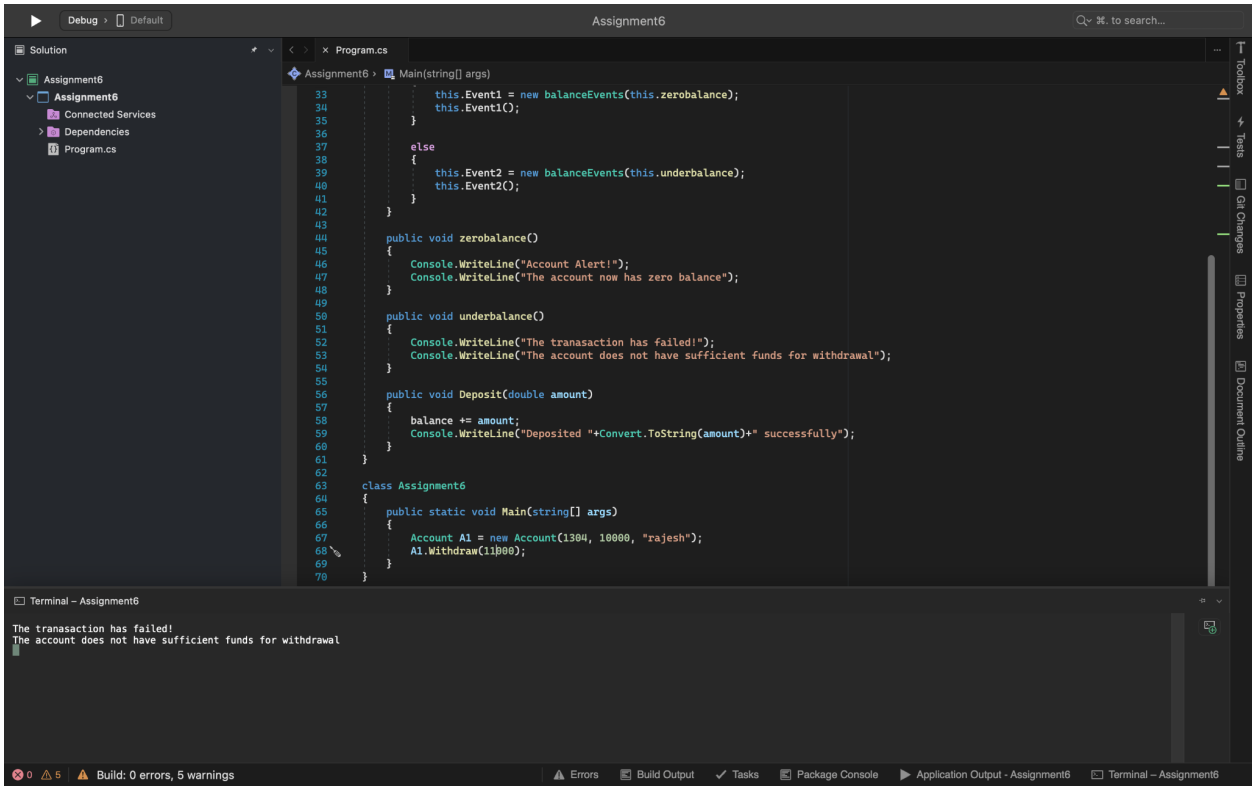
```
public delegate void balanceEvents();

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 tranasaction 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);
    }
}
```


Debug
Default
Assignment6
Solution
Assignment6
Assignment6
Connected Services
Dependencies
Program.cs
Program.cs
Account › Withdraw(double amount)
this.Event1 = new balanceEvents(this.zerobalance);
this.Event1();
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 tranasaction 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
Build: 0 errors, 5 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment6
Terminal – Assignment6
Toolbox
Tests
Git Changes
Properties
Document Outline

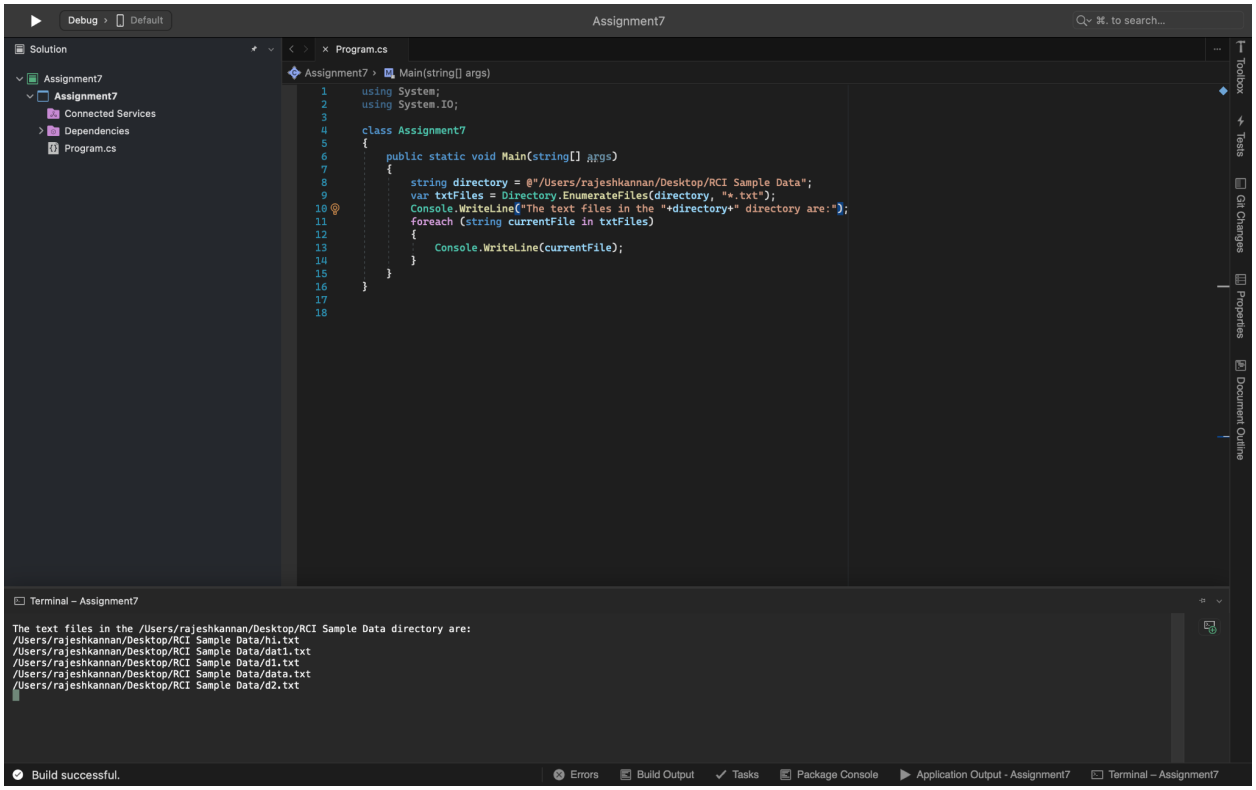

Debug
Default
Assignment6
Solution
Assignment6
Assignment6
Connected Services
Dependencies
Program.cs
Program.cs
Assignment6 › Main(string[] args)
this.Event1 = new balanceEvents(this.zerobalance);
this.Event1();
this.Event2 = new balanceEvents(this.underbalance);
this.Event2();
public void zerobalance()
Console.WriteLine("Account Alert!");
Console.WriteLine("The account now has zero balance");
public void underbalance()
Console.WriteLine("The tranasaction 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
Build: 0 errors, 5 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment6
Terminal – Assignment6
Toolbox
Tests
Git Changes
Properties
Document Outline


Debug
Default
Assignment6
⌘. to search...
Solution
Assignment6
Assignment6
Connected Services
Dependencies
Program.cs
Program.cs
Assignment6 › Main(string[] args)
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 tranasaction 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);
}
}
Terminal – Assignment6
The tranasaction has failed!
The account does not have sufficient funds for withdrawal
Build: 0 errors, 5 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment6
Terminal – Assignment6
Toolbox
Tests
Git Changes
Properties
Document Outline

## Assignment - 7

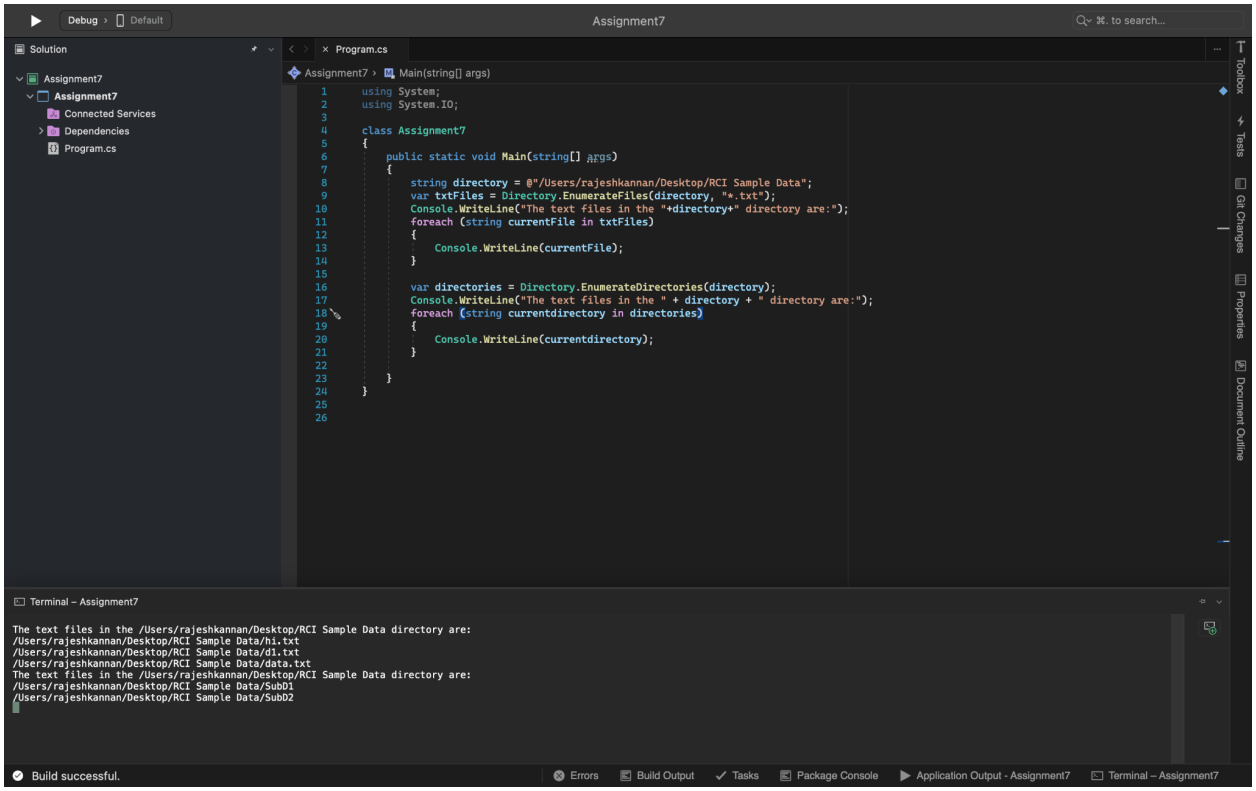
1. Write a console application which will read text files from mentioned file system location. And list subdirectories from mentioned folder on file system using System.IO namespace and use DirectoryInfo, Directory, File and FileInfo Classes with all the methods present in these classes.

```
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);

        }

    }

}
```




```
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);
      }

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

```
    }
}
```




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)
```

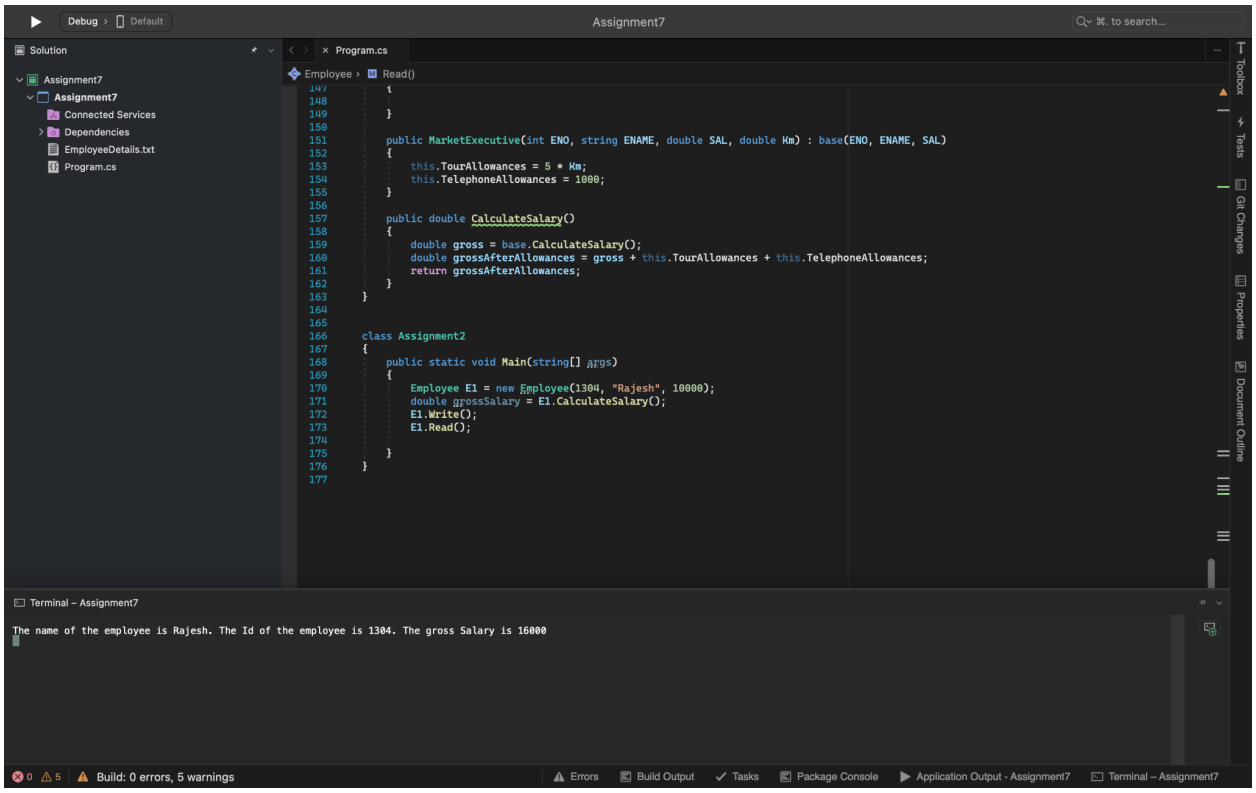
```
        {
            this.HRA = 0.25 * this.Salary;
            this.TA = 0.2 * this.Salary;
            this.DA = 0.3 * this.Salary;
        }

        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();

    }
}
```

EmployeeDetails.txt

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



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));
    }
```

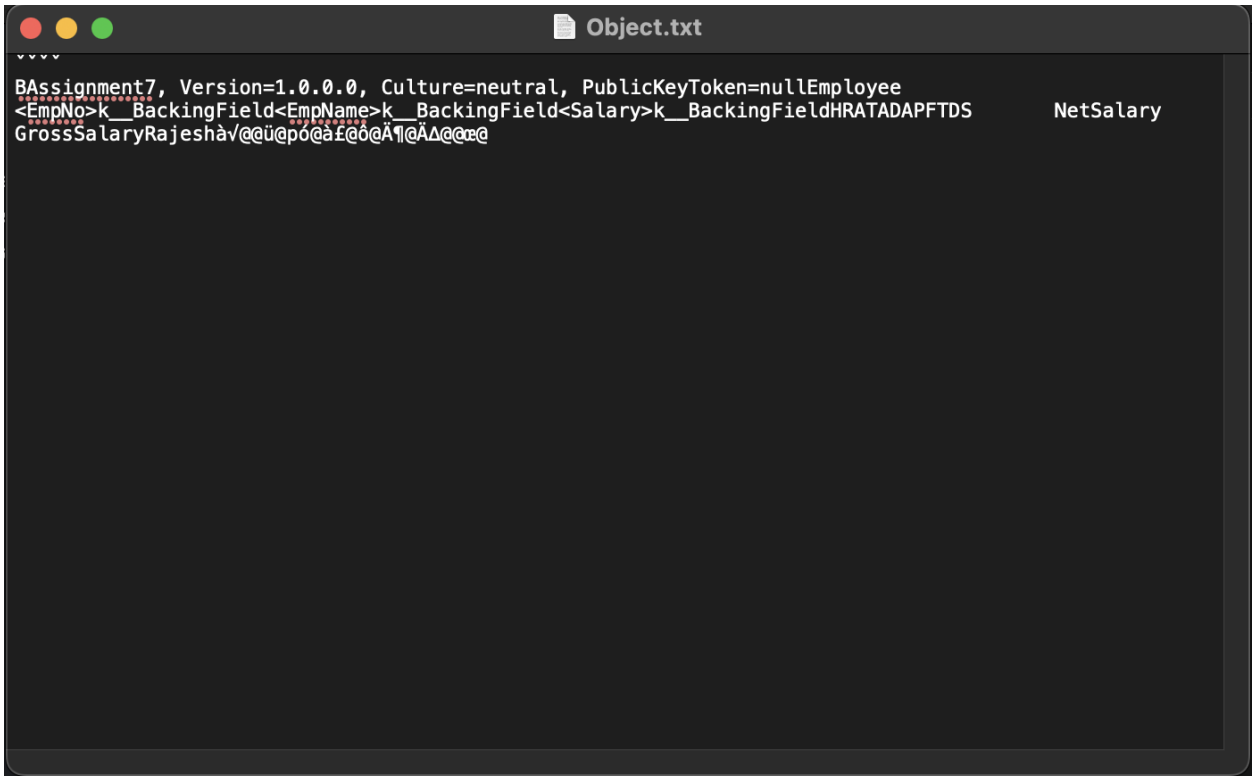
```
    public void Write()

    {

        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();

        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();

    }



    public void Serialize()

    {

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

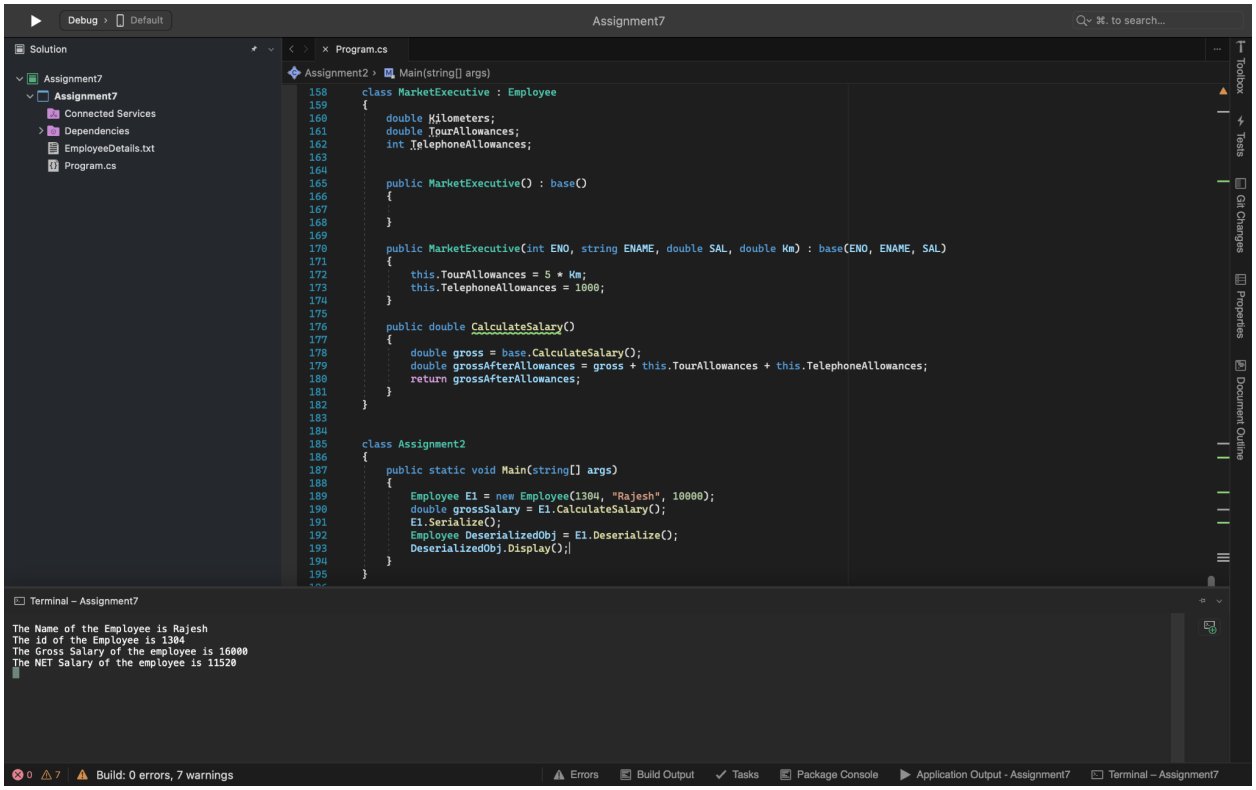
```
        BinaryFormatter f = new BinaryFormatter();
        f.Serialize(s, this);
        s.Close();

    }

    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();
    }
}
```




Debug
Default
Assignment7
⌘. to search...
Solution
Assignment7
Assignment7
Connected Services
Dependencies
EmployeeDetails.txt
Program.cs
Program.cs
Assignment2 › Main(string[] args)
158 class MarketExecutive : Employee
159 {
160 double Kilometers;
161 double TourAllowances;
162 int TelephoneAllowances;
163
164
165 public MarketExecutive() : base()
166 {
167
168 }
169
170 public MarketExecutive(int ENO, string ENAME, double SAL, double Km) : base(ENO, ENAME, SAL)
171 {
172 this.TourAllowances = 5 * Km;
173 this.TelephoneAllowances = 1000;
174 }
175
176 public double CalculateSalary()
177 {
178 double gross = base.CalculateSalary();
179 double grossAfterAllowances = gross + this.TourAllowances + this.TelephoneAllowances;
180 return grossAfterAllowances;
181 }
182 }
183
184
185 class Assignment2
186 {
187 public static void Main(string[] args)
188 {
189 Employee E1 = new Employee(1304, "Rajesh", 10000);
190 double grossSalary = E1.CalculateSalary();
191 E1.Serialize();
192 Employee DeserializedObj = E1.Deserialize();
193 DeserializedObj.Display();
194 }
195 }
Terminal – Assignment7
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
0 7
Build: 0 errors, 7 warnings
Errors
Build Output
Tasks
Package Console
Application Output - Assignment7
Terminal – Assignment7
Toolbox
Tests
Git Changes
Properties
Document Outline