

# **C# and .NET Frameworks**

## **Assignment 1**

**1.Develop the C# program to initialize two dimensional array and print all the elements of the array on the same line separated with space.**

### **AIM:**

To initialize and print a 2D array containing numbers 1 to 9.

### **PROGRAM:**

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        int[,] array = {
```

```
            { 2,3,4},
```

```
            { 5, 6,7 },
```

```
            { 8, 9 ,1}
```

```
        };
```

```
        for (int i = 0; i < array.GetLength(0); i++)
```

```
        {
```

```
            for (int j = 0; j < array.GetLength(1); j++)
```

```
            {
```

```
                Console.Write(array[i, j] + " ");
```

```

    }
}

Console.WriteLine();
}
}

```

The screenshot shows the Programiz C# Online Compiler interface. The code editor on the left contains the following C# code:

```

1 using System;
2
3 class Program
4 {
5     static void Main()
6     {
7         int[,] array = {
8             { 2,3,4 },
9             { 5,6,7 },
10            { 8,9,1 }
11        };
12        for (int i = 0; i < array.GetLength(0); i++)
13        {
14            for (int j = 0; j < array.GetLength(1); j++)
15            {
16                Console.Write(array[i, j] + " ");
17            }
18        }
19
20        Console.WriteLine();
21    }
22 }

```

The output window on the right shows the execution result:

```

mono /tmp/2cDGZM5oEs.exe
2 3 4 5 6 7 8 9 1
=== Code Execution Successful ===

```

## OUTPUT:

2 3 4 5 6 7 8 9 1

**2. Aravind wants to apply for competitive exam. He needs to know whether he is eligible to apply. The eligibility criteria is given below:**

- **Age should be greater than 18 years, but not more than 30.**
- **The candidate should have passed 10 std with a minimum pass percentage of 65.**

**Design the C# program to help him to know his eligibility. If the criteria gets satisfied, print he is eligible else print he is not eligible.**

### **AIM:**

To determine and print whether a person named Aravind is eligible to apply for a competitive exam based on their age and 10th standard pass percentage.

### **PROGRAM:**

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.WriteLine("Enter your age: ");
```

```
        int age = int.Parse(Console.ReadLine());
```

```
        Console.WriteLine("Enter your 10th standard pass percentage: ");
```

```
        double passPercentage = double.Parse(Console.ReadLine());
```

```
        if (age > 18 && age <= 30 && passPercentage >= 65)
```

```
        {
```

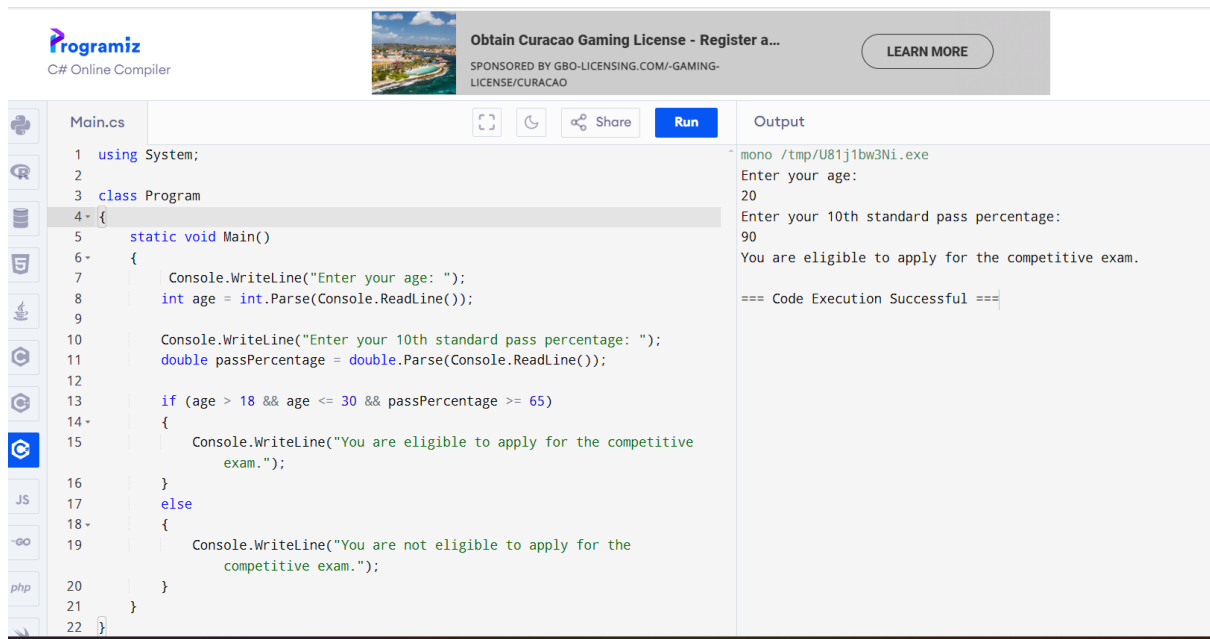
```
            Console.WriteLine("You are eligible to apply for the competitive exam.");
```

```
        }
```

```

        else
        {
            Console.WriteLine("You are not eligible to apply for the competitive
exam.");
        }
    }
}

```



The screenshot shows the Programiz C# Online Compiler interface. At the top, there is a banner for "Obtain Curacao Gaming License - Register a..." with a "LEARN MORE" button. Below the banner, the compiler interface is divided into two main sections: a code editor on the left and an output window on the right.

The code editor shows a C# program named "Main.cs" with the following code:

```

1 using System;
2
3 class Program
4 {
5     static void Main()
6     {
7         Console.WriteLine("Enter your age: ");
8         int age = int.Parse(Console.ReadLine());
9
10        Console.WriteLine("Enter your 10th standard pass percentage: ");
11        double passPercentage = double.Parse(Console.ReadLine());
12
13        if (age > 18 && age <= 30 && passPercentage >= 65)
14        {
15            Console.WriteLine("You are eligible to apply for the competitive
exam.");
16        }
17        else
18        {
19            Console.WriteLine("You are not eligible to apply for the
competitive exam.");
20        }
21    }
22 }

```

The output window on the right shows the execution results:

```

mono /tmp/U81j1bw3Ni.exe
Enter your age:
20
Enter your 10th standard pass percentage:
90
You are eligible to apply for the competitive exam.
=== Code Execution Successful ===

```

## INPUT:

Enter your age:20

Enter your 10th standard pass percentage:90

## OUTPUT:

You are eligible to apply for the competitive exam.

**3.Design the C# console application named validation to get mobile number as input from the user. Validate the mobile number with the following cases:**

- **The first four number must be followed by then followed by next six numbers(eg:9894-256874).**
- **Should contains only numbers.**
- **Should be of length 10.**
- **The first number should start only with 9 Or 8.**

**AIM:**

To validate and print whether a given mobile number is valid or not.

**PROGRAM:**

using System;

using System.Text.RegularExpressions;

class Validation

{

static void Main()

{

Console.WriteLine("Enter your mobile number in the format XXXX-XXXXXX:");

string mobileNumber = Console.ReadLine();

string pattern = @"^[98]\d{3}-\d{6}\$";

if (Regex.IsMatch(mobileNumber, pattern))

{

Console.WriteLine("Valid mobile number.");

}

```

else
{
    Console.WriteLine("Invalid mobile number.");
}
}
}

```

The screenshot shows the Programiz C# Online Compiler interface. The code editor contains a C# program named 'Main.cs' that validates a mobile number. The program prompts the user to enter a mobile number in the format 'XXXX-XXXXXX'. The input '1234567890' is entered, which does not match the required format, resulting in the output 'Invalid mobile number.'.

```

1- using System;
2- using System.Text.RegularExpressions;
3- class Validation
4- {
5-     static void Main()
6-     {
7-         Console.WriteLine("Enter your mobile number in the format XXXX-XXXXXX:");
8-         string mobileNumber = Console.ReadLine();
9-         string pattern = @"^[98]\d{3}-\d{6}$";
10-
11-         if (Regex.IsMatch(mobileNumber, pattern))
12-         {
13-             Console.WriteLine("Valid mobile number.");
14-         }
15-         else
16-         {
17-             Console.WriteLine("Invalid mobile number.");
18-         }
19-     }
20- }

```

Output:

```

mono /tmp/Jq3F1Y18Bc.exe
Enter your mobile number in the format XXXX-XXXXXX:
1234567890
Invalid mobile number.

=== Code Execution Successful ===

```

**INPUT:**

1234567890

**OUTPUT:**

Invalid mobile number.

**4. Write the missing code snippets and the statements in the C# program given below.**

**Class person {**

\_\_\_\_\_name;

\_\_\_\_\_age;

\_\_\_\_\_weight;

**Void printperson() {**

```

// write the code to print name, age and weight of a person
}
}
Class persondata {
    Static void Main(string[] args) {
        person_____ = _____;
        _____.name = "Kannan";
        _____.age = 19;
        _____.weight = 58;
        // write the statement to access printperson() function
    }
}

```

### AIM:

To create a Person class, instantiate it, and print out the person's name, age, and weight using a method.

### PROGRAM:

```

using System;

class Person
{
    public string name;
    public int age;
    public double weight;
    public void PrintPerson()
    {
        Console.WriteLine("Name: " + name);
        Console.WriteLine("Age: " + age);
        Console.WriteLine("Weight: " + weight);
    }
}


```


```

}

class PersonData
{
    static void Main(string[] args)
    {
        Person person1 = new Person();
        person1.name = "Santhiya";
        person1.age = 20;
        person1.weight = 56;
        person1.PrintPerson();
    }
}

```


**Programiz**  
C# Online Compiler



On Any Device & OS - IDBI Bank Star Salary Ac...  
SPONSORED BY WWW.SIGNNOW.COM

LEARN MORE

Main.cs

```

1 using System;
2 class Person
3 {
4     public string name;
5     public int age;
6     public double weight;
7     public void PrintPerson()
8     {
9         Console.WriteLine("Name: " + name);
10        Console.WriteLine("Age: " + age);
11        Console.WriteLine("Weight: " + weight);
12    }
13 }
14 class PersonData
15 {
16     static void Main(string[] args)
17     {
18         Person person1 = new Person();
19         person1.name = "Santhiya";
20         person1.age = 20;
21         person1.weight = 56;
22         person1.PrintPerson();

```

Run

Output

```

mono /tmp/zLOW0mphDp.exe
Name: Santhiya
Age: 20
Weight: 56

=== Code Execution Successful ===

```

## OUTPUT:

Name:Santhiya

Age:20

Weight:56



**5. A hospital wants to create a console application to maintain its inpatient details. The information to store includes:**

- **Name of the patient**
- **Date of admission**
- **Age of patient**
- **Disease**
- **Date of discharge**
- **Total bills paid**

**Design the C# program with the class name patient with necessary data members to store the above information. The class should have two member functions, one to get the patients information and other to display the information. Create a main class called hospital to create necessary instances, methods calling statements and display all the details about the patient.**

**AIM:**

To create a Patient class, collect patient information through user input, and display the collected information using methods.

**PROGRAM:**

using System;

class Patient

{

public string Name;

public string DateOfAdmission;

public int Age;

public string Disease;

public string DateOfDischarge;

```
public double TotalBillsPaid;
```

```
public void GetPatientInfo()
```

```
{
```

```
    Console.Write("Enter patient name: ");
```

```
    Name = Console.ReadLine();
```

```
    Console.Write("Enter date of admission (dd/mm/yyyy): ");
```

```
    DateOfAdmission = Console.ReadLine();
```

```
    Console.Write("Enter patient age: ");
```

```
    Age = int.Parse(Console.ReadLine());
```

```
    Console.Write("Enter the disease: ");
```

```
    Disease = Console.ReadLine();
```

```
    Console.Write("Enter date of discharge (dd/mm/yyyy): ");
```

```
    DateOfDischarge = Console.ReadLine();
```

```
    Console.Write("Enter total bills paid: ");
```

```
    TotalBillsPaid = double.Parse(Console.ReadLine());
```

```
}
```

```
public void DisplayPatientInfo()
```

```
{
```

```

        Console.WriteLine("\n--- Patient Details ---");
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"Date of Admission: {DateOfAdmission}");
        Console.WriteLine($"Age: {Age}");
        Console.WriteLine($"Disease: {Disease}");
        Console.WriteLine($"Date of Discharge: {DateOfDischarge}");
        Console.WriteLine($"Total Bills Paid: {TotalBillsPaid}");
    }
}

class Hospital
{
    static void Main(string[] args)
    {

        Patient patient1 = new Patient();

        patient1.GetPatientInfo();
        patient1.DisplayPatientInfo();
    }
}

```

The screenshot displays the Programiz C# Online Compiler interface. The top header includes the Programiz logo, a navigation bar with 'Premium Coding Courses by Programiz', and a 'Programiz PRO' button. The main workspace is divided into two panels: a code editor on the left and an output console on the right. The code editor shows a C# file named 'Main.cs' with the following content:

```
1 using System;
2
3 class Patient
4 {
5
6     public string Name;
7     public string DateOfAdmission;
8     public int Age;
9     public string Disease;
10    public string DateOfDischarge;
11    public double TotalBillsPaid;
12
13
14    public void GetPatientInfo()
15    {
16        Console.WriteLine("Enter patient name: ");
17        Name = Console.ReadLine();
18
19        Console.WriteLine("Enter date of admission (dd/mm/yyyy): ");
20        DateOfAdmission = Console.ReadLine();
21
22        Console.WriteLine("Enter patient age: ");
23        Age = int.Parse(Console.ReadLine());
24    }
25 }
```

The output console on the right shows the execution results of the program, including the prompts and user inputs, followed by a summary of patient details and a success message:

```
mono /tmp/E53ACvYST4.exe
Enter patient name: Santhiya
Enter date of admission (dd/mm/yyyy): 7/6/2024
Enter patient age: 20
Enter the disease: Fever
Enter date of discharge (dd/mm/yyyy): 15/6/2024
Enter total bills paid: 6000

--- Patient Details ---
Name: Santhiya
Date of Admission: 7/6/2024
Age: 20
Disease: Fever
Date of Discharge: 15/6/2024
Total Bills Paid: 6000

=== Code Execution Successful ===
```

## INPUT:

Enter patient name: Santhiya

Enter date of admission (dd/mm/yyyy): 7/6/2024

Enter patient age: 20

Enter the disease: Fever

Enter date of discharge (dd/mm/yyyy): 15/6/2024

Enter total bills paid: 6000

## OUTPUT:

Name: Santhiya

Date of Admission: 7/6/2024

Age: 20

Disease: Fever

Date of Discharge: 15/6/2024

Total Bills Paid: 6000

**6. Implement the C# code to get two vector number as input, add them and print the sum as another vector. Make use of operator overloading to perform addition of vector numbers.**

**AIM:**

To create a Vector class, overload the '+' operator to add two vectors, and demonstrate vector addition by taking user input for two vectors and displaying their sum.

**PROGRAM:**

```
using System;
```

```
class Vector
```

```
{
```

```
    public int X { get; set; }
```

```
    public int Y { get; set; }
```

```
        public Vector(int x, int y)
```

```
        {
```

```
            X = x;
```

```
            Y = y;
```

```
        }
```

```
        public static Vector operator +(Vector v1, Vector v2)
```

```
        {
```

```
            return new Vector(v1.X + v2.X, v1.Y + v2.Y);
```

```
        }
```

```
        public void Display()
```

```
        {
```

```
            Console.WriteLine($"Vector: ({X}, {Y})");
```

```
}  
}
```

```
class Program
```

```
{
```

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

```
    {
```

```
        Console.WriteLine("Enter the components of the first vector:");
```

```
        Console.Write("X1: ");
```

```
        int x1 = int.Parse(Console.ReadLine());
```

```
        Console.Write("Y1: ");
```

```
        int y1 = int.Parse(Console.ReadLine());
```

```
        Console.WriteLine("Enter the components of the second vector:");
```

```
        Console.Write("X2: ");
```

```
        int x2 = int.Parse(Console.ReadLine());
```

```
        Console.Write("Y2: ");
```

```
        int y2 = int.Parse(Console.ReadLine());
```

```
        Vector v1 = new Vector(x1, y1);
```

```
        Vector v2 = new Vector(x2, y2);
```

```
        Vector result = v1 + v2;
```

```

        Console.WriteLine("The sum of the two vectors is:");
        result.Display();
    }
}

```

The screenshot shows the Programiz C# Online Compiler interface. The code editor on the left contains the following C# code:

```

1 using System;
2
3 class Vector
4 {
5     public int X { get; set; }
6     public int Y { get; set; }
7
8     public Vector(int x, int y)
9     {
10         X = x;
11         Y = y;
12     }
13     public static Vector operator +(Vector v1, Vector v2)
14     {
15         return new Vector(v1.X + v2.X, v1.Y + v2.Y);
16     }
17
18     public void Display()
19     {
20         Console.WriteLine($"Vector: ({X}, {Y})");
21     }
22 }
23
24 class Program

```

The output window on the right shows the following text:

```

mono /tmp/rRCid2wpTi.exe
Enter the components of the first vector:
X1: 2
Y1: 3
Enter the components of the second vector:
X2: 4
Y2: 5
The sum of the two vectors is:
Vector: (6, 8)
=== Code Execution Successful ===

```

## INPUT:

Enter the components of the first vector:

X1: 2

Y1: 3

Enter the components of the second vector:

X2: 4

Y2: 5

## OUTPUT:

The sum of the two vectors is:

Vector: (6, 8)

**7. Create the class student with necessary members to maintain the basic details of a student such as name, age, address and mobile number. Add method getDate() to read the basic details and printData() to print the details of the student. Inherit the student class into the sub class called studentmark with necessary members to maintain student mark details. Override the getDate() and printData() in student mark class to read mark details and print the marks, respectively. Also, define a method to find the grade of the student based on his/her marks. Design the student main class to access the member of both the classes in C#.**

**AIM:**

To create a Student class and a derived StudentMark class, which inherits and extends the base class to include mark details, calculates grades based on marks, and demonstrates polymorphism through overridden methods.

**PROGRAM:**

using System;

```
class Student
{
    public string Name { get; set; }
    public int Age { get; set; }
    public string Address { get; set; }
    public string MobileNumber { get; set; }

    public virtual void GetData()
    {
        Console.Write("Enter student's name: ");
        Name = Console.ReadLine();

        Console.Write("Enter student's age: ");
        Age = int.Parse(Console.ReadLine());
    }
}
```



```
Console.Write("Enter student's address: ");
```

```
Address = Console.ReadLine();
```

```
Console.Write("Enter student's mobile number: ");
```

```
MobileNumber = Console.ReadLine();
```

```
}
```

```
public virtual void PrintData()
```

```
{
```

```
    Console.WriteLine("\n--- Student Details ---");
```

```
    Console.WriteLine($"Name: {Name}");
```

```
    Console.WriteLine($"Age: {Age}");
```

```
    Console.WriteLine($"Address: {Address}");
```

```
    Console.WriteLine($"Mobile Number: {MobileNumber}");
```

```
}
```

```
}
```

```
class StudentMark : Student
```

```
{
```

```
    public int Marks { get; set; }
```

```
    public override void GetData()
```

```
{
```

```
        base.GetData();
```

```
        Console.Write("Enter student's marks: ");
        Marks = int.Parse(Console.ReadLine());
    }

    public override void PrintData()
    {
        // Call the base class method to print basic details
        base.PrintData();

        Console.WriteLine($"Marks: {Marks}");
        Console.WriteLine($"Grade: {CalculateGrade()}");
    }

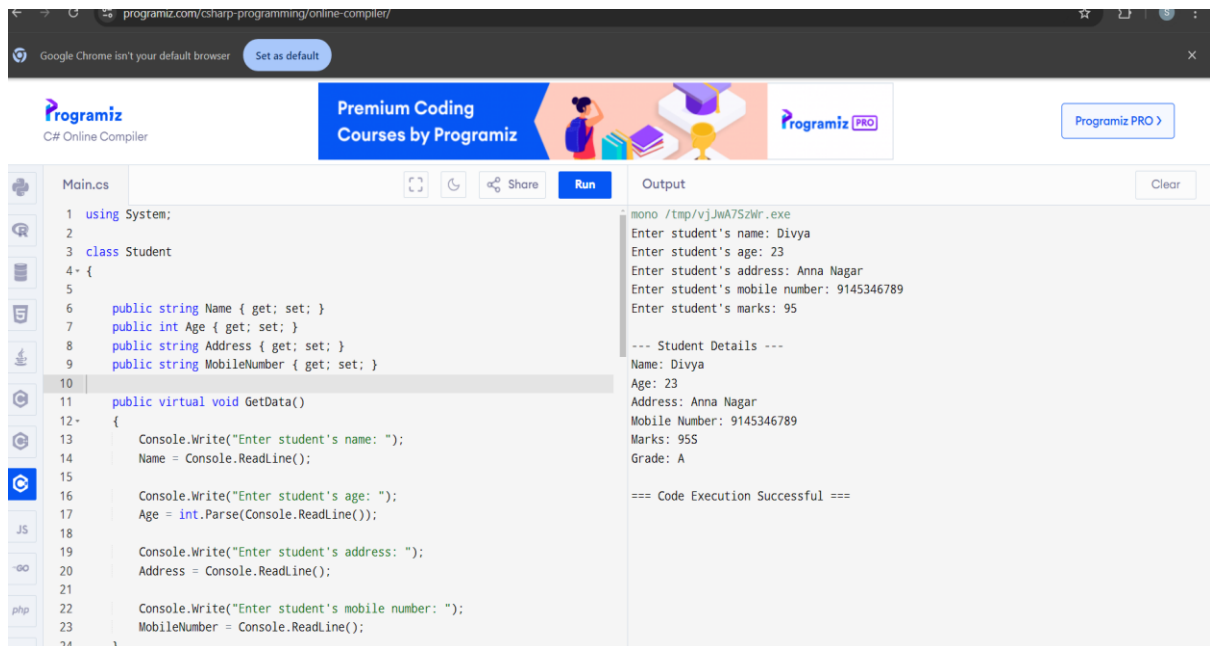
    public string CalculateGrade()
    {
        if (Marks >= 90)
            return "A";
        else if (Marks >= 75)
            return "B";
        else if (Marks >= 50)
            return "C";
        else
            return "F";
    }
}
```

```

class StudentMain
{
    static void Main(string[] args)
    {
        StudentMark student = new StudentMark();

        student.GetData();
        student.PrintData();
    }
}

```



The screenshot shows the Programiz C# Online Compiler interface. The code editor on the left contains the following C# code:

```

1 using System;
2
3 class Student
4 {
5
6     public string Name { get; set; }
7     public int Age { get; set; }
8     public string Address { get; set; }
9     public string MobileNumber { get; set; }
10
11     public virtual void GetData()
12     {
13         Console.WriteLine("Enter student's name: ");
14         Name = Console.ReadLine();
15
16         Console.WriteLine("Enter student's age: ");
17         Age = int.Parse(Console.ReadLine());
18
19         Console.WriteLine("Enter student's address: ");
20         Address = Console.ReadLine();
21
22         Console.WriteLine("Enter student's mobile number: ");
23         MobileNumber = Console.ReadLine();
24     }

```

The output window on the right shows the following text:

```

mono /tmp/vjJwA7SzWlr.exe
Enter student's name: Divya
Enter student's age: 23
Enter student's address: Anna Nagar
Enter student's mobile number: 9145346789
Enter student's marks: 95

--- Student Details ---
Name: Divya
Age: 23
Address: Anna Nagar
Mobile Number: 9145346789
Marks: 95S
Grade: A

=== Code Execution Successful ===

```

## INPUT:

Enter student's name: Divya

Enter student's age: 23

Enter student's address: Anna Nagar

Enter student's mobile number: 9145346789

Enter student's marks: 95

## OUTPUT:

Name: Divya

Age: 23

Address: Anna Nagar

Mobile Number: 9145346789

Marks: 95

Grade: A

**8. Design sample C# program with class name employee to compute netsalary of the employee using the basic salary, if for the job\_catg is 1 use table-I else use table-II. Use constructor to initialize basic salary,hra,da,pf and loan. The employee class should contain input() method to get input for job\_catg, empno, empname, calculateSalary() method to compute salary and display() method to print the details.**

Table-I	Table-II
<b>BASIC=Rs. 8,000</b> <b>HRA=10% of basic DA=20% of basic LOAN=Rs. 300</b> <b>PF=Rs. 500</b>	<b>BASIC=Rs. 15,000</b> <b>HRA=20% of basic DA=30% of basic LOAN=Rs. 600 PF=1000</b>

### **AIM:**

To create an Employee class that calculates and displays an employee's net salary based on their job category, with salary components and deductions, and demonstrates encapsulation and methods.

### **PROGRAM:**

using System;

class Employee

{

```
public int EmpNo { get; set; }  
public string EmpName { get; set; }  
public int JobCategory { get; set; }  
public double BasicSalary { get; set; }  
public double HRA { get; set; }  
public double DA { get; set; }  
public double PF { get; set; }  
public double Loan { get; set; }  
public double NetSalary { get; set; }
```

```
public Employee(double basicSalary, double hra, double da, double pf,  
double loan)
```

```
{  
    BasicSalary = basicSalary;  
    HRA = hra;  
    DA = da;  
    PF = pf;  
    Loan = loan;  
}
```

```
public void Input()  
{  
    Console.Write("Enter Employee Number: ");  
    EmpNo = int.Parse(Console.ReadLine());  
}
```

```
Console.Write("Enter Employee Name: ");
```

```
EmpName = Console.ReadLine();
```

```
Console.Write("Enter Job Category (1 or 2): ");
```

```
JobCategory = int.Parse(Console.ReadLine());
```

```
if (JobCategory == 1)
```

```
{
```

```
    BasicSalary = 8000;
```

```
    HRA = 0.1 * BasicSalary;
```

```
    DA = 0.2 * BasicSalary;
```

```
    PF = 500;
```

```
    Loan = 300;
```

```
}
```

```
else if (JobCategory == 2)
```

```
{
```

```
    BasicSalary = 15000;
```

```
    HRA = 0.2 * BasicSalary;
```

```
    DA = 0.3 * BasicSalary;
```

```
    PF = 1000;
```

```
    Loan = 600;
```

```
}
```

```
else
```

```
{
```

```
    Console.WriteLine("Invalid Job Category. Setting default to Table I.");
```

```
    BasicSalary = 8000;
```

```
        HRA = 0.1 * BasicSalary;
        DA = 0.2 * BasicSalary;
        PF = 500;
        Loan = 300;
    }
}
```

```
public void CalculateSalary()
{
    NetSalary = BasicSalary + HRA + DA - PF - Loan;
}
```

```
public void Display()
{
    Console.WriteLine("\n--- Employee Details ---");
    Console.WriteLine($"Employee Number: {EmpNo}");
    Console.WriteLine($"Employee Name: {EmpName}");
    Console.WriteLine($"Job Category: {JobCategory}");
    Console.WriteLine($"Basic Salary: {BasicSalary}");
    Console.WriteLine($"HRA: {HRA}");
    Console.WriteLine($"DA: {DA}");
    Console.WriteLine($"PF: {PF}");
    Console.WriteLine($"Loan: {Loan}");
    Console.WriteLine($"Net Salary: {NetSalary}");
}
}
```

```
class Program
```

```
{
```

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

```
    {
```

```
        Employee employee = new Employee(0, 0, 0, 0, 0);
```

```
        employee.Input();
```

```
        employee.CalculateSalary();
```

```
        employee.Display();
```

```
    }
```

```
}
```

The screenshot displays the Programiz C# Online Compiler interface. The editor on the left contains a C# program named 'Main.cs' that defines an 'Employee' class with properties for EmpNo, EmpName, JobCategory, BasicSalary, HRA, DA, PF, Loan, and NetSalary. It includes a constructor to initialize these values and a 'Main' method that creates an employee object, calls 'Input()', 'CalculateSalary()', and 'Display()' methods. The output window on the right shows the program's execution, where user inputs (4 for EmpNo, Nivetha for EmpName, 2 for JobCategory) are processed to calculate and display the employee's details, including a total Net Salary of 20900. The output concludes with '=== Code Execution Successful ==='.

```
1 using System;
2
3 class Employee
4 {
5
6     public int EmpNo { get; set; }
7     public string EmpName { get; set; }
8     public int JobCategory { get; set; }
9     public double BasicSalary { get; set; }
10    public double HRA { get; set; }
11    public double DA { get; set; }
12    public double PF { get; set; }
13    public double Loan { get; set; }
14    public double NetSalary { get; set; }
15
16    // Constructor to initialize salary components
17    public Employee(double basicSalary, double hra, double da, double pf,
18                    double loan)
19    {
20        BasicSalary = basicSalary;
21        HRA = hra;
22        DA = da;
23        PF = pf;
```

Output

```
mono /tmp/vwLauKfsVF.exe
Enter Employee Number: 4
Enter Employee Name: Nivetha
Enter Job Category (1 or 2): 2

--- Employee Details ---
Employee Number: 4
Employee Name: Nivetha
Job Category: 2
Basic Salary: 15000
HRA: 3000
DA: 4500
PF: 1000
Loan: 600
Net Salary: 20900

=== Code Execution Successful ===
```



**INPUT:**

Enter Employee Number: 4

Enter Employee Name: Nivetha

Enter Job Category (1 or 2): 2

**OUTPUT:**

Employee Number: 4

Employee Name: Nivetha

Job Category: 2

Basic Salary: 15000

HRA: 3000

DA: 4500

PF: 1000

Loan: 600

Net Salary: 20900

BY:

Arun K

73772214111

III-B.E CSE