**Project - E-Comm website**

Role → 1. Admin 2. Customer

Role - Admin

Database

Users

Roles

Products

Category

Cart

Orders

Should be able to perform CRUD on the product, CRUD on category

All users

Register page

Login page

Customers

View products

Add to cart page

View orders page

https://www.programiz.com/csharp-programming/online-compiler/

<https://eap.myexamo.com/>

[**https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/**](https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/)

**https://www.programiz.com/csharp-programming/online-compiler/**

**https://learn.microsoft.com/en-us/visualstudio/ide/csharp-developer-productivity?view=vs-2022**

Software & Hardware requirement

1. Windows OS - 8 / 10 /11
2. Visual Studio - 2019 / 2022
3. MS - SQL Server with management studio - 2012 / 2019
4. Postman
5. Github for desktop
6. Github web login

Hardware :

12 GB - RAM

C drive -> 10 GB space -> 25 GB

VS Setup

<https://visualstudio.microsoft.com/downloads/>

MS -SQL server

<https://www.microsoft.com/en-in/sql-server/sql-server-downloads>

Install SSMS

https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Training Syllabus

C# -> .NET framework 4.7

Lang fundamental

OOPS

Class

Interfaces

Delegates

Exception

Collections

ADO.NET

File handling

SQL -> MS SQL server

DDL, DML, DQL, joins, subquery,.....

Web Development :

HTML 5

CSS 3

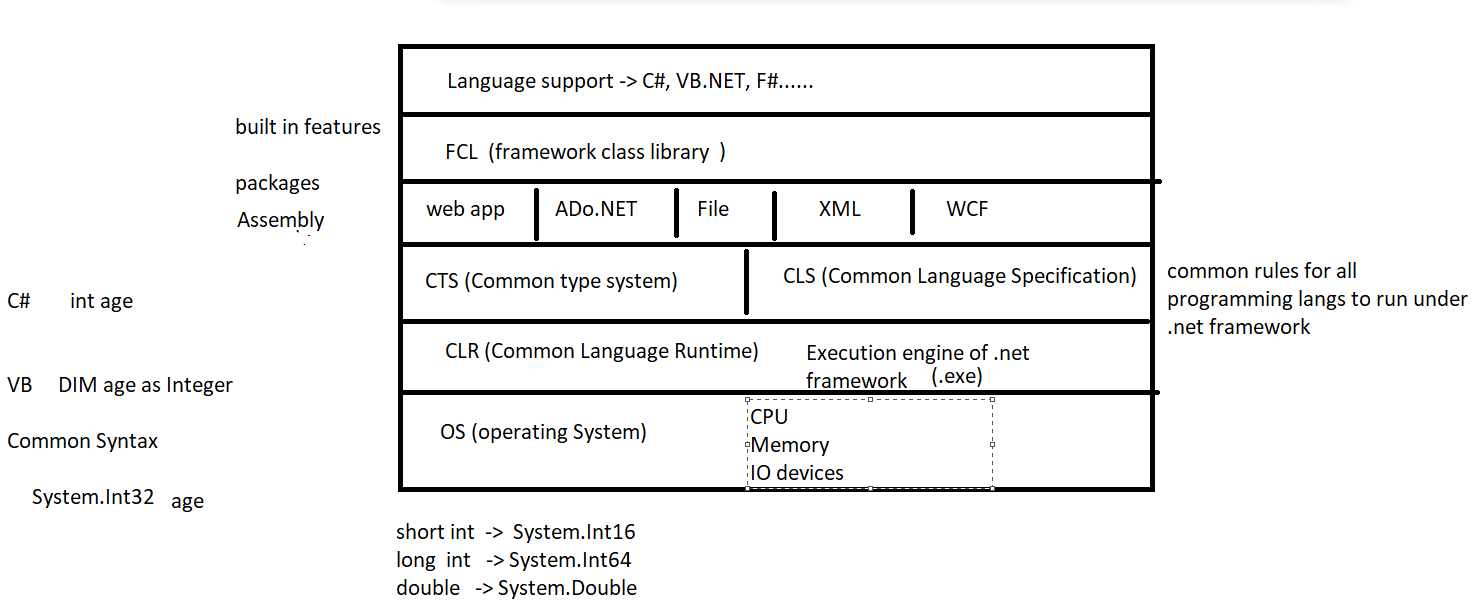
JavaScript

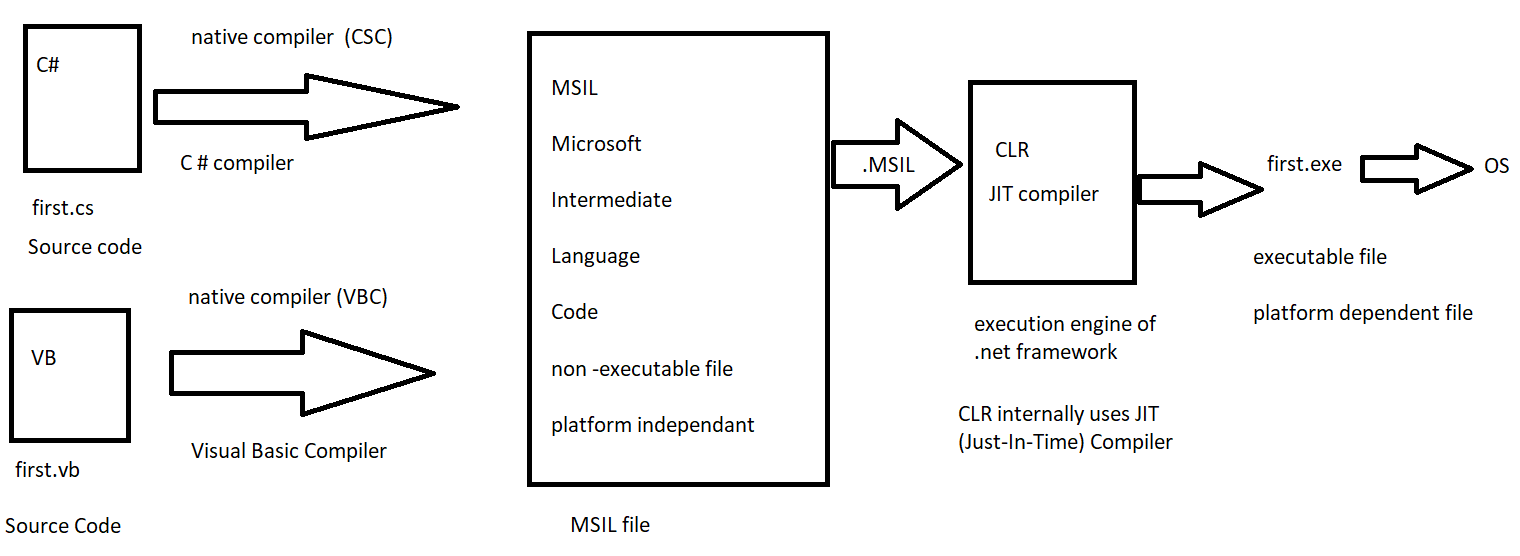
Bootstrap 5

ASP.NET core 6.0 web development using MVC

Introduction to .NET framework

1. What is .NET framework -> It is a software framework, design & maintain by Microsoft to design, develop, host, test, build the applications & services
2. Components of .Net framework





**Services of CLR :**

1. Automatic memory management (GC -garbage collection)
2. Handling errors at runtime (Exception handling)

**Language Fundamental**

1. **Variables**
2. **Data types**
3. **Operators**
4. **Input / output statements in C#**
5. **Creating simple console based project using C#**
6. **Variables -> a name given to memory location is called variable**

**0 - 65635 address locations**

| **Pune** |
| --- |

**100**

**city—--> variable name**

**Rules for variable declaration**

1. Variable must start with alphabet
2. Special symbols are not allowed in variable declaration (excepts \_ (under score)
3. White space not allowed in variable declaration

Valid declaration

First\_name

Firstname

N1

Number1

Not valid while variable declarations

First name

1n

n@me

**Data types :**  data type defines the type of data that we want to store in the program

String city = “Pune” - to hold collection of characters - string data type -> String

Short - > Int16

Int year=2022 - to hold whole number int data type Int32

Long Int64

Float per=78.45f; f- stands for float - data type float -> Single

Double per=78.45; default data type is double -> Double

Bool True / false - boolean data type in C# -> Bool

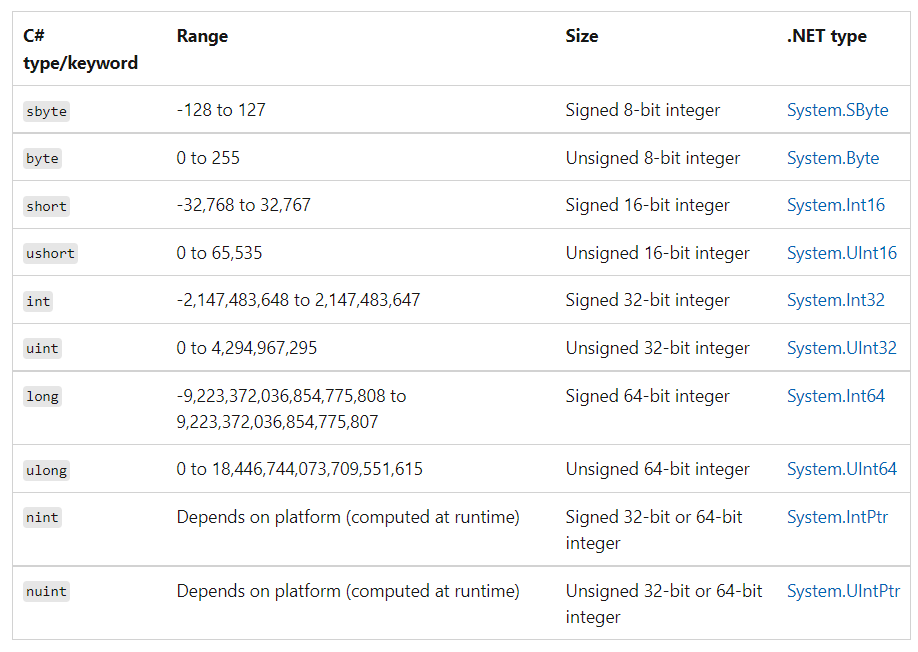
Bool isAdmin=true isAdmin=false

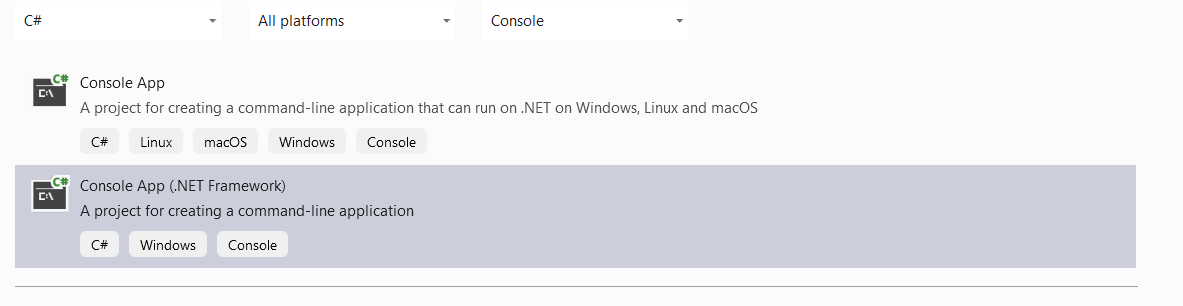
Bool isUser=false

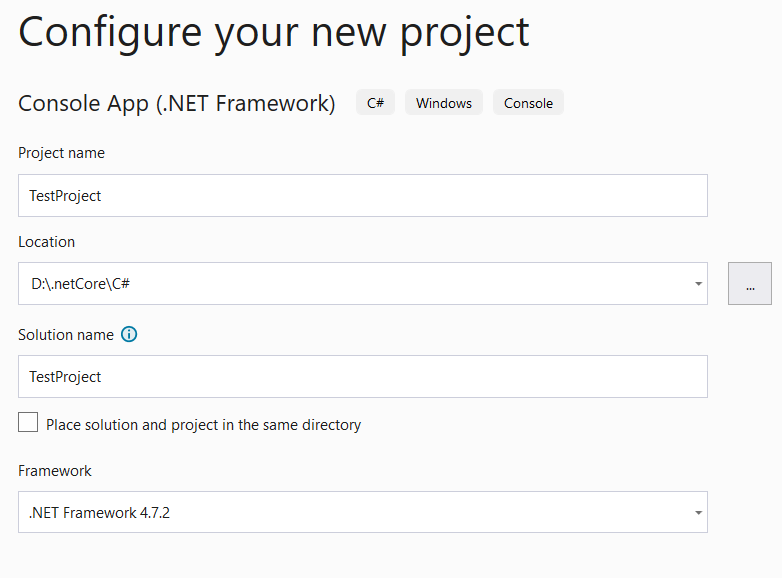
Char gender=’M’ char is to hold single character , value must be enclosed in single quote

Char gender =’F’

M







static void Main(string[] args)

{

Console.WriteLine("This is my first program");

int age = 20;

string city = "pune";

float per = 78.45f;

//1 + sign

Console.WriteLine("age is "+age+" my city is "+city);

//2 place holder

Console.WriteLine("age is {0} my city is {1} per is {2}",age,city,per);

// 3. interpolation -> $

Console.WriteLine($"age is {age} city is {city} per is {per}");

}

static void Main(string[] args)

{

Console.WriteLine("Enter name");

string name=Console.ReadLine();

Console.WriteLine("Enter a number");

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

Console.WriteLine("Enter percentage");

double per=Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"your name {name} , selected num {num} & per {per}");

}

**Operators in C#**

1. **Arithmetic operator**
2. **Relational**
3. **Logical**
4. **Unnary**
5. **Tarnary**
6. **Arithmetic operator**
   1. + addition
   2. - subtraction
   3. \* multiplication
   4. / division
   5. % mod -> used to find the remainder

static void Main(string[] args)

{

Console.WriteLine("Enter two numbers");

int num1 =Convert.ToInt32(Console.ReadLine());

int num2 = Convert.ToInt32(Console.ReadLine());

double div = (double)num1 / num2;

Console.WriteLine($"division is {div}");

}

1. Relational Operator : check the relation between two oprants

| < | Less than |
| --- | --- |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not equal to |

Int a=50, b=40;

a==b → true

a!=b → true

1. Logical Operator
2. && -> logical and
3. || -> logical or
4. ! -> logical not

&& logical and

nationality==”IND” && age >=18

Allow for vote

IND 17 → false

AUS 21 → false

IND 25 → true

USA 17 → false

Logical OR

Person ==”aadhar” || person==”Pan card” || person==”election card”

Valid ID for vote

T T → T

T F → T

F T → T

F F → F

**4. Unary Operator:**

It will be used with one operant

1. ++ → increment the value by one
2. - - → decrement the value by one

Int a=10;

a++;

a=11

a- -

a=9

1. Post fix operator a++
2. Prefix operator ++a

static void Main(string[] args)

{

// a++ -> it will assign the value & then increment by one

// ++a => it will increment the value & then assign

int a = 10, b,c;

b = a++;

Console.WriteLine($" a = {a} b= {b}");

c = ++a;

Console.WriteLine($" a = {a} c= {c}");

}

static void Main(string[] args)

{

int a = 10;

int b = a++;

int c= ++b + ++a;

int d = (++a + b--) \* ++c;

//

Console.WriteLine(a); // 13

Console.WriteLine(b); // 10

Console.WriteLine(c); // 24

Console.WriteLine(d); // 576

}

**Ternary Operator:**

**? : → this is ternary operator**

**To check any expression if the expression is true then print or assign value else assign default or false value**

**(expression ) ? true statement : false statement**

static void Main(string[] args)

{

int a = 10, b = 11;

int c =(++a> ++b) ? a : b;

Console.WriteLine(c);

}

**Conditional statements :**

1. **If….else**
2. **If else if …. Else if … else**
3. **Nested if else**
4. **Switch case**

**if …. Else → check the condition & based on the result print the appropriate statements or display some output**

static void Main(string[] args)

{

Console.WriteLine("Enter your age");

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

if (age >= 18) // true

{

Console.WriteLine("you are eligible for vote");

}

else // false or default statement

{

Console.WriteLine("you are not eligible for vote");

}

}

static void Main(string[] args)

{

Console.WriteLine("Enter your percentage");

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

if (per >= 75)

{

Console.WriteLine("Grade is A+");

}

else if(per >=50 && per <75) // range 50 - 74

{

Console.WriteLine("Grade is A");

}

else if (per >= 40 && per < 50) // range 40 - 49

{

Console.WriteLine("Grade is B");

}

else

{

Console.WriteLine("You are Fail");

}

Console.WriteLine("End of program");

}

**Assignments : (Homework)**

1. Enter a number & check entered number is even or odd
2. Accept the number from user & check number is +ve or -ve or zero
3. WAP to accept a character from user & print whether it is vowel or consonant
4. Accept a year from user & find whether it is leap year or not

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

**{**

**Console.WriteLine("Enter a char");**

**char ch = Convert.ToChar(Console.ReadLine());**

**if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')**

**{**

**Console.WriteLine($"{ch} is a vowel");**

**}**

**else**

**{**

**Console.WriteLine($"{ch} is a consonent");**

**}**

**}**

**Nested if else**

**if(condition 1)**

**{**

**if(condition 2)**

**{**

**if(condtion 3)**

**{**

**Code**

**}**

**}**

**Else{**

**}**

**}**

**Else{**

**}**

1. **Accept 3 numbers from user & find the greatest number from 3**

Switch case in C#

General e.g of switch case is mobile call operator

Lang selection

1. Eng
2. Hindi
3. Marathi
4. Gujrathi

Another options

1. Prepaid
2. Postpaid
3. Internet plan

static void Main(string[] args)

{

Console.WriteLine("select any one option from following");

Console.WriteLine("1. English");

Console.WriteLine("2. Hindi");

Console.WriteLine("3. Marathi");

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

switch (num)

{

case 1: // num==1

Console.WriteLine("You have selected English Language");

Console.WriteLine();

break;

case 2: // num==2

Console.WriteLine("You have selected Hindi Language");

break;

case 3: // num==3

Console.WriteLine("You have selected Marathi Language");

break;

default:

Console.WriteLine("Wrong selection of number");

break ;

}

}

Accept two numbers from user & ask for following options

1. Add
2. Sub
3. Mul
4. Div
5. Mod

Based on user selection perform the arithmetic operation & display result

**Homework :**

Accept marks of 3 subjects physics ,chemistry , maths out of 100. Calculate percentage. Display grade as below.

Percentage > 80% - Grade A

Percentage > 70% - Grade B

Percentage > 60% - Grade C

Percentage < 60% - Grade D

Input basic salary and calculate gross salary as follows ,

Basic salary <= 10000 ,

DA = 80% , HRA = 20%

Basic salary <= 20000 ,

DA = 90% , HRA = 35%

Else

DA = 95% , HRA = 40%

Enter a bs -> 25000

DA 95% -> BS -> x

HRA 40 % -> BS -> y

Gross =bs + x +y;

Loop

While

Do while

For

What is Loop ? -> when we want to iterate code form more than one times… we can use loop

3 things

1. Initial value
2. Condition
3. Increment / decrement the value based on condition

For loop:

static void Main(string[] args)

{

// print Hello for the 10 times

for (int i = 1; i <=10; i++)

{

Console.WriteLine("Hello");

}

}

i=1 i<=10 print -> Hello i++

1 1<=10 Hello 2

2<=10 Hello 3

3<=10 Hello 4

.

.

10<=10 Hello 11

11<=10

1. WAP to print numbers from 100 - 50
2. WAP to print numbers from 1- 50 even numbers & also print total count
3. WAP to print numbers from 50 - 80 odd numbers & its count
4. WAP to accept a number from user & find the prime number
5. WAP to accept a number from user & find the number is Armstrong number

153 → (1\*1\*1) +(5\*5\*5) + (3\*3\*3) → 153

1 + 125 + 27 => 153

11 -> 1 + 1 => 2

1. Print the fibonacci series (accept the upper limit from user - 8)

0 1 1 2 3 5 8 13

1. Print prime numbers between 1- 100
2. Accept two numbers from user & calculate multiplication , without using multiplication / \*

Operator

**While loop**

Initial value

Condition

Increment & decrement

int i = 10; // initial

while (i <=500) // condition

{

Console.WriteLine(i);

i +=50; // -= is a short hand operator i=i-3; += \*= /= --> i=i\*2

}

int i = 10; // initial

do

{

Console.WriteLine(i);

i += 50;

} while (i <= 500);

For loop & while are called as entry control loop

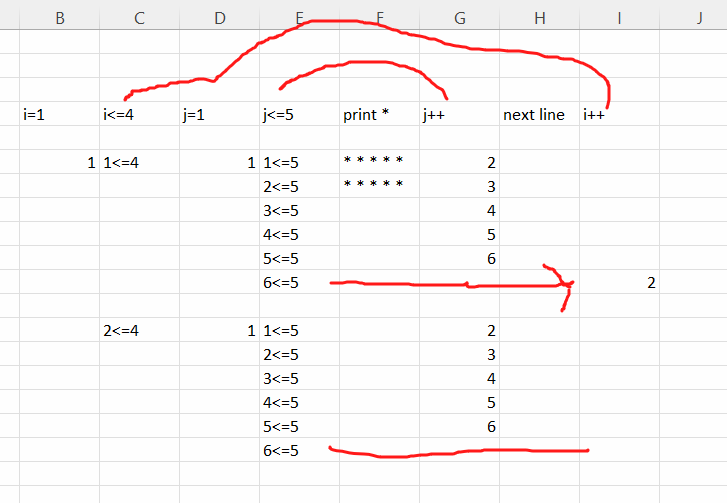
Do .. while is called as exit control loop

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*



\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

1

12

123

1234

12345

54321

5432

543

54

5

12345

1234

123

12

1

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

for (int i = 1; i <=5; i++) //row

{

for (int j = 1; j <=i; j++) // column

{

Console.Write("\* ");

}

Console.WriteLine();

}

static void Main(string[] args)

{

// nested loop

bool flag= true;

for (int i = 2; i <=100; i++) //row

{

for (int j = 2; j <=100; j++) // column

{

if(i!=j && i%j==0)

{

flag = false;

break;

}

}

if (flag)

{

Console.WriteLine(i);

}

isPrime = true;

}

}

static void Main(string[] args)

{

int num=Convert.ToInt32(Console.ReadLine()); //153

int temp = num;

int sum=0,r;

while (num > 0)

{

r = num % 10; // 1

sum = sum + (r \* r \* r); //27 +125 +1

num = num / 10; // 153/10 -> 15 15/10 -> 1

}

if (temp == sum)

{

Console.WriteLine("Armstrong");

}

else

{

Console.WriteLine("Not Armstrong");

}

}

static void Main(string[] args)

{

int n1 = Convert.ToInt32(Console.ReadLine());// 3

int n2 = Convert.ToInt32(Console.ReadLine());//4

int m = 0;

for(int i = 1; i <= n2; i++) //1--> 4

{

m+= n1; //3 +3 +3+3

}

Console.WriteLine($"Multiplication {m}");

}

**Ref → call by reference**

internal class Class1

{

static void Swap(ref int a,ref int b)

{

int temp;

temp = a;

a = b;

b = temp;

}

static void Main(string[] args)

{

// call by ref

int a = 10, b = 20;

Swap(ref a, ref b);

Console.WriteLine($"a= {a} b= {b}");

}

}

**Out→ you want to out multiple response from the called function / method**

internal class Class1

{

static void Calculation(int a,int b,out int sum,out int sub,out int multiply)

{

sum = a + b;

sub = a - b;

multiply = a \* b;

}

static void Main(string[] args)

{

int a = 10, b = 20,sum,sub,multiply;

Calculation(a, b, out sum, out sub, out multiply);

Console.WriteLine($"sum= {sum} sub= {sub} multiplication {multiply}");

}

}

**Params → accept variable number of parameters of same data type**

internal class Class1

{

static void AcceptNames(params string[] names)

{

for (int i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

Console.WriteLine("-------------------");

}

static void Main(string[] args)

{

AcceptNames("Kishor", "Rahul", "Amol");

AcceptNames("Tushar");

AcceptNames("Rohan", "Suraj");

}

}

##### A Duck number is a positive number that has zero (0) present in it. It may contain more than one 0 as an element

##### but not as a leading element only. 0 should be present somewhere in the middle or as the last element of the digit. E.g.: 403,4002,43320,039830,034043 are all examples of duck number.

##### Condition: If a number has only led 0 as an element, then it will not be considered as a duck number. E.g.: 0432,0229,0643 are not examples of duck numbers.

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

**{**

**int flag = 0;**

**int count = 0;**

**string str=Console.ReadLine();**

**for (int i = 1; i < str.Length; i++)**

**{**

**if (str[0] == '0')**

**{**

**flag = 1;**

**}**

**if (str[i] == '0')**

**{**

**count++;**

**flag = 0;**

**break;**

**}**

**}**

**if (flag == 1)**

**{**

**Console.WriteLine("Not Duck");**

**}**

**else if(count>0 || flag==0)**

**{**

**Console.WriteLine("Duck");**

**}**

**}**

**Object Oriented Programming Structure (OOPs)**

**What is Object**

Object is a real world entity which has attributes & behaviour

Chair, car, mobile, laptop, bank, cheque, table , mouse….tv, ac, person, employee, student

Living things, / non living things / tangible / intangible / conceptual entity

Attributes → characteristics of that object

Behaviour → task / features of that object

**Vehicle is an object**

Attributes

Comp name

Model number

Model name

Color

Cost

Avg

Fuel type

Behaviour

Start

Accelerate

Apply break

Shift gear

On/ off the lights

Stop

Employee

Student

Mobile

**What is Object oriented programming ?**

Map a real world entity (object ) into the specific programming language. C# , C++, Java, Python….

**4 pillars of OOPs**

* **Abstraction →** It is a mechanism in which we will select the essential features(attributes & behaviour) of an object & ignore the rest

**Mobile (nokia)**

Compname, model name, color, cost, no of sim

Make a call, receive a call, send a message, receive a message, start, restart, switch off

Person

| **Social survey** | **Hospital** |
| --- | --- |
| name | name |
| Mobile no | Mobile no |
| address | address |
| gender | gender |
| Income group | BG |
| religion | Ht, wt |
| No of person in family | age |
|  | Previous health record |

* **Encapsulation :**It is a mechanism in which we hide the data, hide the internal structure & implementation details of an object

**- Security**

**- reduce the complexity**

* **Inheritance :** It is a mechanism in which we can create a new object from existing object, so new object reuse the features of existing object & new object have its own features
* Reusability
* Extensibility

Smart Phone

Comp name, model name, color, cost……make call, send / receive msg— reuse

ROM, RAM, processor

Connect device using BT, using Hotspot, take a picture using camera

* **Polymorphism:** It is a mechanism in which multiple objects may have same behaviour but its implementation is different

**Mobile -> receive a call -> press a button**

**Smart Phone -> receive a call -> swipe a screen / touch a screen**

**Student:**

Abstraction

Attributes

Rollno, name, gender, address, contact, branch, DOB, email

Behaviour

Do Study, attend lectures, complete the assignments, wear ID card, uniform, give exam

Encapsulation

Hide data, internal structure & implementation details

Hide data→ Rollno, name, gender, address, contact, branch, DOB, email

Internal / implementation details → give exam

Inheritance

Engg Student

Reuse → from student

**extend→ Do Practical, industry visit , industrial projects**

Polymorphism

**Student -> give exam -> written test**

**Engg Student -> give exam → written + practical +project submit**

**Class :**

What is class ? → class is a user defined type in C#, Which is a blueprint of an object

Object → class

Attributes → variables / data member

Behaviour → methods / functions

Class Employee

{

Int empid;

String empname, gender;

Double salary;

Void CalculateSalary()

{

}

}

Main()

{

Int a=10;

for(int i=1;i<=10;i++)

{

}

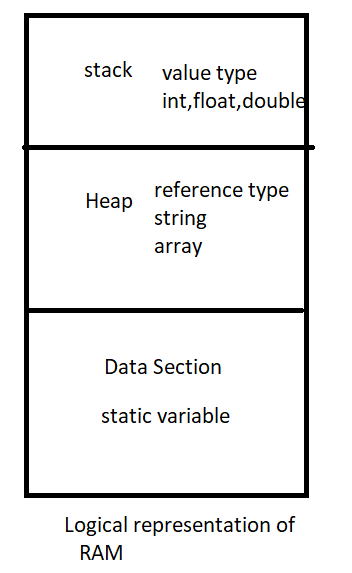
}

Types in C#

Two types

1. Value type → memory will be allocated on the stack ( struct, enum, int,float, double)
2. Reference type → memory will be allocated on the heap area ( string, array, class, interface,

delegate) -> memory management done by CLR (garbage collector)



**Standard coding practice :**

Project Name → capital case. Do not use under\_score , do not use space

Class name , Method name, Property name → capital case , do not use under\_score ,do not use space

Data member / variables → small case / lower case , do not use under\_score,do not use space

Interface name → should start with I (capital I)

using System;

using System.Collections.Generic; // system namespaces

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// namespace -> is act as container can contains all the types

// struct, enum, class, interface, delegate--> always add these types in namespace

namespace TestProject

{

// members --> variables, methods,constructor, property , event

internal class Program // class contains members

{

int a;

static void Main(string[] args)

{

}

void Test()

{

}

}

}

namespace N1

{

class Stud

{

void Test()

{

}

}

}

namespace N2

{

class Emp

{

}

}

Access specifiers in C#

| public | Access of members within all namespaces |
| --- | --- |
| private | Access within class only |
| protected | Access within class & access in child class |
| Internal | Inside the namespace direct access |
| Protected internal | Direct access within namespace & access within child classes |
| Private protected | Access within class , access in child class but child should be in same namespace |

In side namespace → all types have default access as internal

Inside the class → default access is private

public class Date

{

private int day,year;

private string month;

// using method assign value to the member

public void AddValues(int d,string m , int y)

{

day = d;

month = m;

year = y;

}

public string GetValues()

{

return $"Date : {day}/{month}/{year}"; //Date : 29/Sept/2022

}

}

public class Program

{

static void Main(string[] args)

{

// create object of Date class

Date d1 = new Date(); // new allocate memory dynamically on the heap

Console.WriteLine("Enter day month & year");

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

string m = Console.ReadLine();

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

d1.AddValues(d,m,y);

Console.WriteLine(d1.GetValues());

Date d2 = new Date();

d2.AddValues(10,"Jan",2022);

Console.WriteLine(d2.GetValues());

}

}

**Homework:**

Student class → roll no , name, percentage, 3 subjects marks , calculate total & percentage

Employee -> empid, empname & salary

Course → id, course name, fees -> create 3 objects to display 3 diff courses

public class Student

{

private int rollno,sub1,sub2,sub3,total;

private string name;

private double percentage;

public void GetData(int rollno,string name,int sub1,int sub2,int sub3)

{

this.rollno = rollno; // this keyword refers to data member

this.name = name;

this.sub1 = sub1;

this.sub2 = sub2;

this.sub3 = sub3;

}

public void Calculate()

{

total = sub1 + sub2 + sub3;

percentage = (double)total / 3;

}

public string Print()

{

return $"Roll No={rollno} Name={name} Total= {total} Percentage={percentage}";

}

}

static void Main(string[] args)

{

Student s1 = new Student();

s1.GetData(1, "Tushar", 89, 78, 91);

s1.Calculate();

Console.WriteLine(s1.Print());

}

public class Course

{

private int id;

private string name;

private float fees;

public void Add(int id,string name,float fees)

{

this.id = id;

this.name = name;

this.fees = fees;

}

public string Display()

{

return $" Course Id {id} name {name} Fees {fees}";

}

}

static void Main(string[] args)

{

Course c1 = new Course();

c1.Add(101, "Java", 45000);

Console.WriteLine(c1.Display());

Course c2 = new Course();

c2.Add(102, "C#", 40000);

Console.WriteLine(c2.Display());

}

Create class product id, name, price, discount

Accept the details

Calculate the discounted price

Display product with name & price

1

HP laptop

40000

10

40000 → 10 % → 4000

Hp Laptop Price 36000

Homework :

Employee → empid, empname, bs & calculate allowances as follows

Hra → 40 % of bs

Da→ 20 % of bs

Pf → 12 % of bs

Calculate total salary & net paid salary

Bs+hra+da → total

Total- pf → net paid salary

**IMP Qustions**

1. what is .net framework --> it is a software platform, develop & maintain by microsoft, dev app & services

2. explain CLR & its services -->execution engine for .net farmework

memory management, handles exception

3. explain object oriented programming

4. what is namespace --> logical group of types (class, struct, interface, delegate, enum)

5. what is class --> class is a user defined type, which is a blueprint of an object

6. value type --> memory will be allocated on stack (int, float, double, struct, enum)

7. ref type --> memory will be allocated on heap (class, string ,array, interface, delegate)

8. access specifier

**Constructor:**

public class Date

{

private int day,year;

private string month;

// using method assign value to the member

// constructor - non parametric

public Date()

{

day = 1;

month = "Jan";

year = 2022;

}

// parametric constructor

public Date(int day, int year, string month)

{

this.day = day;

this.year = year;

this.month = month;

}

public string GetValues()

{

return $"Date : {day}/{month}/{year}"; //Date : 29/Sept/2022

}

}

static void Main(string[] args)

{

Date d1 = new Date(); // non parametric

Console.WriteLine(d1.GetValues());

Date d2 = new Date(3, 2022, "Oct"); // parametric constructor

Console.WriteLine(d2.GetValues());

}

**ToString() method in C# →**  ToString method is used to display string representation of an object

public class Course

{

private int id;

private string name;

private float fees;

public Course(int id,string name,float fees)

{

this.id = id;

this.name = name;

this.fees = fees;

}

public override string ToString()

{

return $" Course Id {id} name {name} Fees {fees}";

}

}

public class Student

{

private int rollno,sub1,sub2,sub3,total;

private string name;

private double percentage;

public Student(int rollno,string name,int sub1,int sub2,int sub3)

{

this.rollno = rollno; // this keyword refers to data member

this.name = name;

this.sub1 = sub1;

this.sub2 = sub2;

this.sub3 = sub3;

}

public void Calculate()

{

total = sub1 + sub2 + sub3;

percentage = (double)total / 3;

}

public override string ToString()

{

return $"Roll No={rollno} Name={name} Total= {total} Percentage={percentage}";

}

}

public class Date

{

private int day,year;

private string month;

// using method assign value to the member

// constructor - non parametric

public Date()

{

day = 1;

month = "Jan";

year = 2022;

}

// parametric constructor

public Date(int day, int year, string month)

{

this.day = day;

this.year = year;

this.month = month;

}

public override string ToString()

{

return $"Date : {day}/{month}/{year}";

}

}

static void Main(string[] args)

{

Date d1 = new Date(4,2022,"Oct"); // parametric

Console.WriteLine(d1); // implicity call to ToString

Date d2 = new Date(3, 2022, "Oct"); // parametric constructor

Console.WriteLine(d2);

Student s1 = new Student(1, "Test", 67, 45, 56);

s1.Calculate();

Console.WriteLine(s1);

Course c2 = new Course(101,"C#",45000);

Console.WriteLine(c2); // ToString()

}

**Properties in C#**

1. **Property is a special method of a class**
2. **Which has get & set accessors**
3. **Get is used to retrieve value from data member**
4. **Set accessor is used to assign value to the data member**

**public class Course**

**{**

**private int id;**

**private string name;**

**private float fees;**

**// property to assign value & return from data member**

**public int Id // Id is property name**

**{**

**get { return id; } // return value from data member**

**set { id = value; } // value is built in keyword to assign value**

**}**

**public string Name**

**{**

**get { return name; }**

**set { name = value; }**

**}**

**public float Fees**

**{**

**get { return fees; }**

**set { fees = value;}**

**}**

**public override string ToString()**

**{**

**return $" Course Id {id} name {name} Fees {fees}";**

**}**

**}**

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

**{**

**// property initializer syntax**

**Course c1 = new Course();**

**c1.Id= Convert.ToInt32(Console.ReadLine()); // internally app calls to set accessor**

**c1.Name =Console.ReadLine();**

**c1.Fees = Convert.ToSingle(Console.ReadLine());**

**Console.WriteLine($"{c1.Id} {c1.Name} {c1.Fees}"); // call the get accessor**

**// using object initializer**

**Course c2 = new Course { Id = 2, Name = "Java", Fees = 50000f };**

**Console.WriteLine($"{c2.Id} {c2.Name} {c2.Fees}");**

**}**

**Assignment :**

**Department → deptid,deptname,location (data member)**

**Create property for each member**

**Use property initializer & object initializer syntax to assign the values**

**& print the dept details**

**Auto implemented properties in C#**

Auto implemented properties are also called as smart fields, because these properties will create data members by their own to store & retrieve data from it

public class Course

{

// auto implemented property prop -> tab button 2

public int Id { get; set; }

public string Name { get; set; }

public float Fees { get; set; }

public override string ToString()

{

return $"{Id} {Name} {Fees}";

}

}

public class Program

{

static void Main(string[] args)

{

// property initializer syntax

Course c1 = new Course();

c1.Id= Convert.ToInt32(Console.ReadLine()); // internally app calls to set accessor

c1.Name =Console.ReadLine();

c1.Fees = Convert.ToSingle(Console.ReadLine());

Console.WriteLine($"{c1.Id} {c1.Name} {c1.Fees}"); // call the get accessor

// using object initializer

Course c2 = new Course { Id = 2, Name = "Java", Fees = 50000f };

Console.WriteLine($"{c2.Id} {c2.Name} {c2.Fees}");

}

}

**HomeWork :**

Using property –Emp salary calculation

Basic salary 20000- 30000

Hra → 40% da →20 % pf → 12 %

Basic salary 10000-20000

Hra → 45% da → 25 % pf→ 12 %

<10000 —> only deduct 200 PT (professional Tax)

Show all the details

Emp id

Emp name

Hra

Da

Pf

Gross(Total)

Net paid

public class Course

{

// auto implemented property prop -> tab button 2

public int Id { get; set; }

public string Name { get; set; }

public float Fees { get; set; }

public float Discount { get; set; }

public float FinalAmount { get; set; }

public void Calculation()

{

float amt = Fees \* Discount;

FinalAmount = Fees - amt;

}

public override string ToString()

{

return $"{Id} {Name} {Fees} {FinalAmount}";

}

}

Discount=0.10;

**10/10/2022**

static void Main(string[] args)

{

char ch = 'A';

int i, j, k, m;

for (i = 1; i <= 5; i++)

{

for (j = 5; j >= i; j--)

Console.Write(" ");

for (k = 1; k <= i; k++)

Console.Write(ch++);

ch--;

for (m = 1; m < i; m++)

Console.Write(--ch);

Console.Write("\n");

ch = 'A';

}

}

**Constructor in C#**

1. It is method of a class which has same name as class name
2. It is used to assign value to the data member
3. constructor get called automatically when object is created
4. Does not return any value
5. We can overload the constructor

Code that get executed when object is initialised we can use constructor

ON the wifi of mobile → search the hotspot device in a range

namespace TestProject

{

public class Product

{

private int id, price;

private string name;

public Product(int id, int price, string name)

{

this.id = id;

this.price = price;

this.name = name;

}

public string Print()

{

return $"{id} {name} {price}";

}

}

public class Program

{

static void Main(string[] args)

{

Product p1 = new Product(1, 999, "Mouse");// call to constructor

Product p2 = new Product(2,1999,"keyboard");

Console.WriteLine(p1.Print());

Console.WriteLine(p2.Print());

}

}

}

**Homework: constructor + ToString()**

Course class

Student class

Employee class

ToString () method

ToString () method gives the string representation of an object

namespace TestProject

{

public class Product

{

private int id, price;

private string name;

public Product(int id, int price, string name)

{

this.id = id;

this.price = price;

this.name = name;

}

public override string ToString() // it is from base class object

{

return $"{id} {name} {price}";

}

}

public class Program

{

static void Main(string[] args)

{

Product p1 = new Product(1, 999, "Mouse");// call to constructor

Product p2 = new Product(2,1999,"keyboard");

Console.WriteLine(p1);

Console.WriteLine(p2);

}

}

}

**Property in C#:**

1. Property is a special method which has get & set accessors
2. Set accessor is used to assign value to the data member
3. Get accessor is used to return the value from data member

namespace TestProject

{

public class Product

{

private int id, price;

private string name;

public int Id

{

get { return id; }

set { id = value; } // value is a built in keyword to assign value

}

public int Price

{

get { return price; }

set { price = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

public override string ToString() // it is from base class object

{

return $"{id} {name} {price}";

}

}

public class Program

{

static void Main(string[] args)

{

// using property initializer

Product p1 = new Product();

p1.Name = "Mouse"; // set

p1.Id = 1;

p1.Price = 999;

Console.WriteLine(p1.Name+ " "+p1.Price); //get

Console.WriteLine(p1);

// object initializer syntax

Product p2 = new Product { Id=2,Name="keyboard",Price=1200};

Console.WriteLine(p2);

}

}

}

**Homework: property + ToString()--> property & object initializer syntax**

Course class

Student class

Employee class

Auto implemented property:

This property will create a variables by their own, when we create a property, so property internally creates variable to store the value & also return the value from it

namespace TestProject

{

public class Product

{

// prop --> tab + 2

public int Id { get; set; }

public string Name { get; set; }

public int Price { get; set; }

public override string ToString() // it is from base class object

{

return $"{Id} {Name} {Price}";

}

}

public class Program

{

static void Main(string[] args)

{

// using property initializer

Product p1 = new Product();

p1.Name = "Mouse"; // set

p1.Id = 1;

p1.Price = 999;

Console.WriteLine(p1.Name+ " "+p1.Price); //get

Console.WriteLine(p1);

// object initializer syntax

Product p2 = new Product { Id=2,Name="keyboard",Price=1200};

Console.WriteLine(p2);

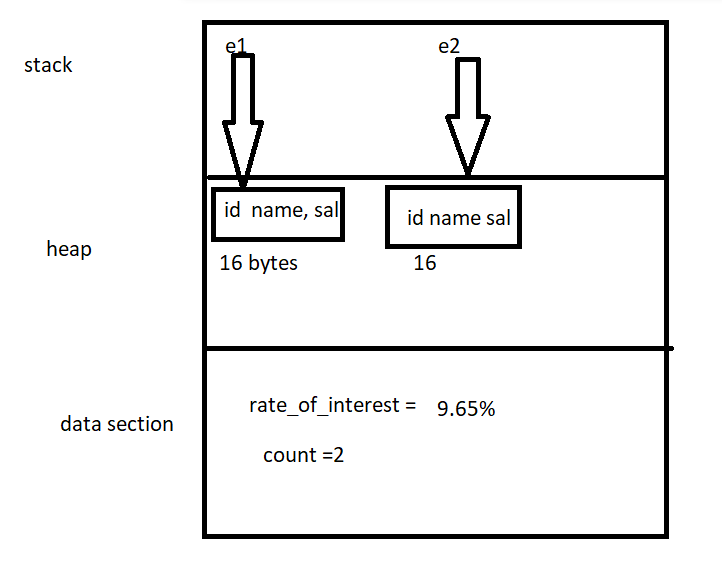
}

}

}

Student → create auto implemented property & call it in the main method

**Static variable & static method**

****

**Static variable :**

1. Static variable is also called as class variable
2. Single copy of static variable will be created
3. All the objects of class shared the same value
4. Static variable have default value 0
5. It will be initialised only once in the program execution
6. The value which is common in the class can be a static
7. Like company name, auto generate the id of emps

**Static method:**

1. It is a class method
2. It is used to print the static data, values
3. Static method needs to call using class name

public class Employee

{

private int id;//4

private string name;//4

private double basicsalary;//8

private static int count;

public Employee(string name, double basicsalary)

{

count++;

this.id =count;

this.name = name;

this.basicsalary = basicsalary;

}

public static string TotalCount()

{

return $"Total count is {count}";

}

public void CalculateSalary()

{

hra=basicsalary\* 0.40;

da = basicsalary \* 0.20;

pf = basicsalary \* 0.12;

netpaid = (hra + da + basicsalary) - pf;

}

public override string ToString()

{

return $"Id :{id} Name :{name} Salary {basicsalary}";

}

}

static void Main(string[] args)

{

Employee e1 = new Employee("emp1", 23000);

Employee e2 = new Employee("emp2", 24000);

Employee e3 = new Employee("emp3", 24000);

e1.CalculateSalary();

e2.CalculateSalary();

e3.CalculateSalary();

Console.WriteLine(e1);

Console.WriteLine(e2);

Console.WriteLine(e3);

Console.WriteLine(Employee.TotalCount());

}

Add hra, da, pf, netpaid

Create a method CaluclateSalary

Hra 40 % bs

Da 20 % bs

Pf 12 % bs

Calculate net paid salary → print

Using ToString()

**Homework:**

Create class student

Auto generate the rollno

Accept name & 3 subject marks from constructor

Use calculate method to calculate total & percentage

Display details using ToString()

**Inheritance :** we can create new class from the existing class,

**Base :** it is used to call the base class members

Polymorphism : to achieve polymorphism in C#

1. Using virtual & override keyword
2. Using new keyword

Virtual → allow base class method to be overridden (apply to base class method)

Override → override the method which is in derived class (apply to the derived class method)

Method Overriding

1. Base class method name & derived class method names are same
2. Number of parameters , data type should be same
3. Return type should be same

public class Employee

{

protected int id;

protected string name;

protected double basicsalary,hra,da,pf,netpaid;

private static int count;

public Employee(string name, double basicsalary)

{

count++;

this.id =count;

this.name = name;

this.basicsalary = basicsalary;

}

public virtual void CalculateSalary()

{

hra=basicsalary\* 0.40;

da = basicsalary \* 0.20;

pf = basicsalary \* 0.12;

netpaid = (hra + da + basicsalary) - pf;

}

public static string TotalCount()

{

return $"Total count is {count}";

}

public override string ToString()

{

return $"Employee details -->Id :{id} Name :{name} Salary {netpaid}";

}

}

public class Manager : Employee

{

private double ta;

public Manager(string name,double bs, double ta) : base(name,bs) //calls the emp ctor

{

this.ta = ta;

}

public override void CalculateSalary()

{

hra = basicsalary \* 0.40;

da = basicsalary \* 0.20;

pf = basicsalary \* 0.12;

netpaid = (hra + da + basicsalary+ta) - pf;

}

public override string ToString()

{

return $"Manager details -->Id :{id} Name :{name} Salary {netpaid}";

}

}

static void Main(string[] args)

{

Manager m1 = new Manager("man1", 33000, 4000);

m1.CalculateSalary();

Console.WriteLine(m1);

}

OR

static void Main(string[] args)

{

Manager m1 = new Manager("man1", 33000, 4000);

m1.CalculateSalary();

Console.WriteLine(m1);

Employee e1 = new Manager("Man2", 34000, 3000);

e1.CalculateSalary(); // run time binding

Console.WriteLine(e1);

}

Create a class President → as derived class

Create a class SalesManager → as derived class

Use of new keyword in polymorphism:

New keyword hide the base class method & implement derived class method

* When you don't have access to base class member

**Method Overloading:**

In method overloading method names are same, but have different parameters

1. Method name should be same
2. Number of parameters should be different
3. Data types & their sequence should be different
4. It is in the same class
5. We can overload static method also
6. Return type does not matter

public class Calculation

{

public int Add(int a,int b)

{

return a + b;

}

public int Add(int a, int b,int c)

{

return a + b +c;

}

public double Add(double a, int b)

{

return a + b;

}

public double Add(float a, double b)

{

return a + b;

}

}

static void Main(string[] args)

{

Calculation c1 = new Calculation();

Console.WriteLine(c1.Add(1, 2, 3));

Console.WriteLine(c1.Add(2, 2));

}

Difference between method overloading & overriding

| **Method Overloading** | **Method Overriding** |
| --- | --- |
| It is in the same class | It is in the child class |
| Parameters should be different | Parameters should be same |
| Return type does not matter | Return type should be same |
| It is also called compile time polymorphism, compile time binding | Run time polymorphism / run time binding |
| Binding of object with method is at compile time | Binding of object with method is at run time |
| Static methods can be overload | Static methods can not be override |

**Abstract class**

**Abstract :**  which does not have implementation

Abstract public Class Animal

{

Public abstract void Sound();

Public void DisplayDetails()

{

}

}

Public class Dog:Animal{

Public override void Sound()

{

}

}

Dog d=new Dog();

d.Sound();

d.DisplayDetails();

1. A method which does not implementation, that method needs to be declare in the program so that method should be an abstract method
2. If a class contains an abstract method class must be an abstract
3. If a child class inherits an abstract class then must provide definition to the abstract method
4. An abstract method is by default virtual, so we will override the method in the child class
5. We can not create object of abstract class
6. Abstract class may contains non abstract method

namespace TestProject

{

abstract public class Shape

{

public abstract void CalculateArea();

}

public class Circle : Shape

{

public int r;

public double result;

public Circle(int r)

{

this.r = r;

}

public override void CalculateArea()

{

result = (double)r \* r \* 3.14;

}

public override string ToString()

{

return $"Area of circle is {result}";

}

}

}

class Program

{

static void Main(string[] args)

{

Circle c1 = new Circle(5);

c1.CalculateArea();

Console.WriteLine(c1);

}

}

Ref keyword

Call by value → we pass values through the parameter

Call by reference → we pass address through the parameter, in C# we use ref keyword

Out keyword

class Program

{

static void Swap(ref int a,ref int b)

{

int temp;

temp = a;

a = b;

b = temp;

}

static void Main(string[] args)

{

// call by reference

int a = 10, b = 20;

Swap(ref a, ref b);

Console.WriteLine($"a= {a} b={b}");

}

}

**Out Keyword:**

1. It is used to out multiple values from a method
2. In C# we can return a single value at a time, so when we want to return multiple values we can use out keyword
3. Out keyword can be pass as empty at the time of call to method
4. It is compulsory that defined method gives value to each out variable

class Program

{

static void Calculation(int a,int b,out int sum,out int sub,out int mul)

{

sum = a +b;

sub = a - b;

mul = a \*b;

}

static void Main(string[] args)

{

// call by value

int a = 45, b = 23, sum, sub, mul;

Calculation(a, b, out sum, out sub, out mul);

Console.WriteLine(sum);

Console.WriteLine(sub);

Console.WriteLine(mul);

}

}

**Interface in c#**

**What is the need of an interface ?**

When multiple classes are not in hierarchy but having common features that can be kept into an interface. It is a contract between provider & consumer

interface–

Cut () → common feature

Tailor → to cut the cloths

Barber → to cut the hairs

Interface→

PayTax () → pay a tax to govt

Private job -> 3L - 8 L -> 5 %

Business → 3L - 8L -> 8 %

Business → 10 –15L -> 10%

namespace TestProject

{

public interface ICut

{

void Cut();

}

public class Barber : ICut

{

public void Cut()

{

//

}

}

public interface IPritable

{

void Display();

}

public interface IPayable {

void PayTax();

}

public class PrivateJob : IPayable, IPritable

{

public void PayTax()

{

// 5%

}

public void Display()

{

}

}

public class Business : IPayable

{

private double incomegrp,amt;

public void PayTax()

{

// 10%

}

}

class Program

{

static void Main(string[] args)

{

}

}

}

Rules :

1. Interface is a reference type in C#,
2. All the members in an interface are by default public
3. Interface contains method declaration , auto implemented properties & events
4. Data members can not included in an interface
5. When a class implements an interface needs to implement all methods from an interface
6. A class can implement multiple interfaces

Assignment :

Create an interface with name IPrintable -> which contains String Print () method

Implement IPrintable interface in to the Emp class & student class to print respective details

→ to implement above example comment the ToString()

**Explicit interface implementation :**

namespace TestProject

{

public interface ICustomer

{

string Details();

}

public interface IOrder

{

string Details();

}

public class Transaction : ICustomer, IOrder

{

// implement interface explicitly

string ICustomer.Details()

{

return "Customer details";

}

string IOrder.Details()

{

return "Order details";

}

}

class Program

{

static void Main(string[] args)

{

ICustomer c=new Transaction();

Console.WriteLine(c.Details());

IOrder o = new Transaction();

Console.WriteLine(o.Details());

}

}

}

**Const keyword**

**Readonly keyword**

To define fix value in the program we can use const & readonly

| **Const** | **readonly** |
| --- | --- |
| It is a compile time const | It is a run time constant |
| Value needs to be assign at the time of declaration | Value can be assign at declaration or inside constructor |
| It can be data member or local variable | It must be data member |

public class Circle1

{

private const double PI=3.14;

private readonly double pi;

public Circle1(double d)

{

pi = d;

}

public void Calculate()

{

const int MAX = 100;

}

}

**Sealed keyword →**  it is used to restrict inheritance & method to be overridden make a class sealed

Public sealed class Brush

{

Public void Draw()

{

}

}

**Collections in C#**

1. Array
2. Advanced collections

Array → it is used to store group of similar data elements, at contiguous memory location

Array size need to be fixed at compile time

Why array ?

Rainfall for a 3 months → 90 days

Mon

Tue

Wed

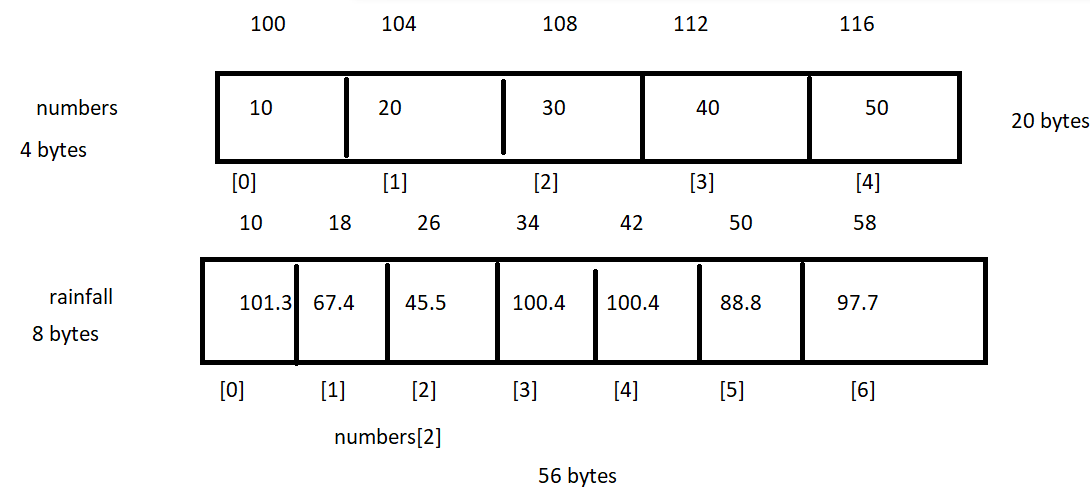
The

Fri

Sat

sun

Can be of int , float, double, string… student, employee…



static void Main(string[] args)

{

// syntax of array in C#

int[] numbers = new int[5]; // 5 size

// to hold rainfall for a week

double[] rainfall = new double[7] {101.3,67.4,45.5,100.4,100.4,88.8,99.7};

string[] names = { "Amol", "Rahul", "Kishor", "Rajesh", "Rohan" };

for (int i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

//for (int i = 0; i < numbers.Length; i++)

//{

// numbers[i] = Convert.ToInt32(Console.ReadLine());

//}

//for (int i = 0; i < numbers.Length; i++)

//{

// Console.WriteLine($"numbers[{i}] = {numbers[i]}");

//}

for (int i = 0; i < rainfall.Length; i++)

{

Console.WriteLine(rainfall[i]);

}

}

| **For loop** | **Foreach loop** |
| --- | --- |
| Works with index | Work with value |
| We can put a condition to iterate | No condition |
| We can modify the data in for loop | Read only |
| Print the elements in forward & in reverse direction | Forward only |

static void Main(string[] args)

{

// syntax of array in C#

int[] numbers = { 10, 20, 30, 40, 50 };

string[] names = { "Amol", "Kishor", "Rajesh" };

for (int i = names.Length-1; i >=0; i--)

{

names[0] = "Test";

Console.WriteLine(names[i]);

}

// foreach loop

foreach (int n in numbers)

{

if(n>0)

Console.WriteLine(n);

}

foreach (string nm in names)

{

Console.WriteLine(nm);

}

}

2D array → 2 dimensions → stores data in the memory at contiguous location

1- row

2- column

Fix value of row & fix value of column

Multiplex

10 rows 10 \* 25 → 250 total people can seat in the auditorium

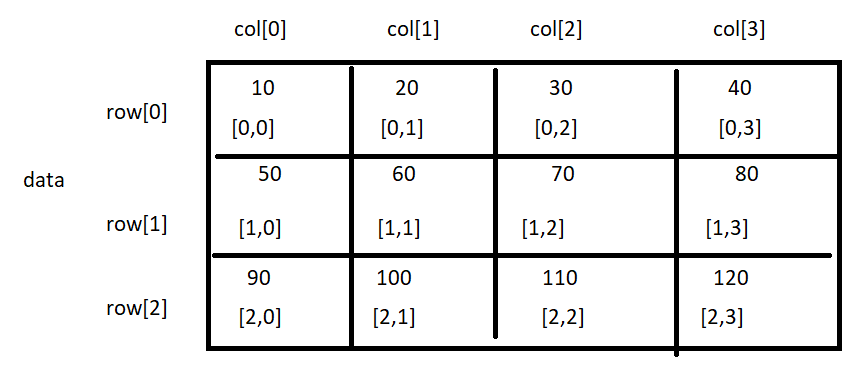
25 seats

Info to store

10

25

250 → values can be store



static void Main(string[] args)

{

// 2D array syntax

int[,] data = new int[3, 4]

{

{10,20,30,40 },

{ 50,60,70,80},

{ 90,100,110,120}

};

int[,] data2 = new int[4, 6]

{

{ 1,2,3,4,5,6},

{1,2,3,4,5,6 },

{1,2,3,4,5,6 },

{ 1,2,3,4,5,6}

};

for (int i = 0; i < data.GetLength(0); i++) // 0--> row size

{

for (int j = 0; j < data.GetLength(1); j++) // 1 --> col size

{

Console.Write($"data=[{i},{j}] = {data[i,j]} ");

}

Console.WriteLine();

}

foreach (int item in data)

{

Console.WriteLine(item);

}

}

Accept 2D array from user & display using for loop

1. Find the maximum number from an array of 10 numbers.
2. Find the minimum number from an array of 10 numbers.
3. Print alternate number from an array
4. Count number of occurrences of a number in an array [1,2,3,1,3,4,5]
5. Reverse an array of 5 numbers

static void Main(string[] args)

{

int[] arr = new int[10];

int[] counter = new int[10];

Console.WriteLine("Enter elements");

for (int i = 0; i < arr.Length; i++)

{

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

counter[i] = -1;

}

for (int i = 0; i < arr.Length; i++)

{

if (counter[i] == 0)

continue;

counter[i] = 1;

for (int j = i+1; j < arr.Length; j++)

{

if (arr[i] == arr[j])

{

counter[i]++;

counter[j] = 0;

}

}

}

for (int i = 0; i < arr.Length; i++)

{

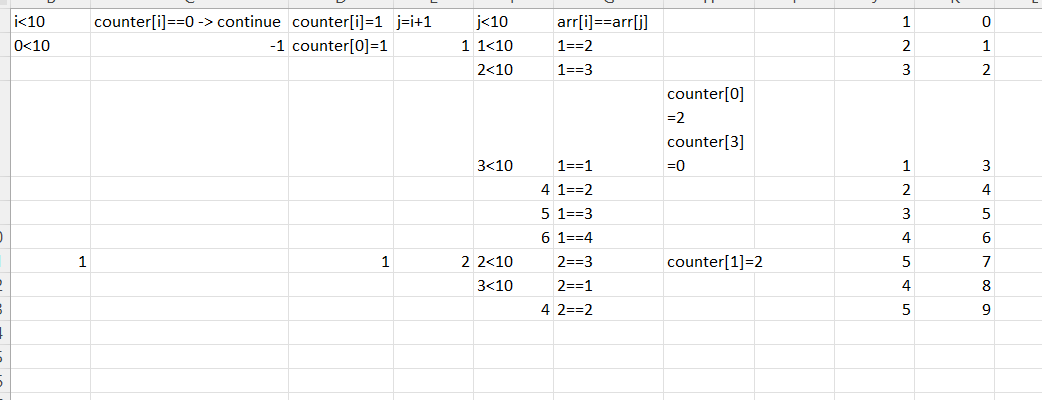
if (counter[i] == 0)

continue;

Console.WriteLine($"{arr[i]} count is {counter[i]}");

}

}



Array class Methods → Sort, Reverse, Clear , Copy

static void Main(string[] args)

{

// methods in 1D array

int[] arr = new int[] { 23, 56, 12, 89, 22 };

int[] arr2 = new int[3];

int[] arr3 = new int[5];

Console.WriteLine("original array");

foreach (var item in arr)

{

Console.WriteLine(item);

}

// built in class Array

Array.Copy(arr, 0, arr2, 0, 2);

Array.Copy(arr, arr3, 5);

Array.Sort(arr);

Array.Reverse(arr);

Array.Clear(arr3, 3, 2);

Console.WriteLine("sorted array");

foreach (var item in arr)

{

Console.WriteLine(item);

}

Console.WriteLine("reverse array");

foreach (var item in arr)

{

Console.WriteLine(item);

}

Console.WriteLine("print arr 2");

foreach (var item in arr2)

{

Console.WriteLine(item);

}

Console.WriteLine("print arr 3");

foreach (var item in arr3)

{

Console.WriteLine(item);

}

}

static void Main(string[] args)

{

// jagged array

// row size is fixed & col size is vary.. it is also called as array of array

int[][] jarray = new int[3][];

jarray[0] = new int[] { 1, 2, 3, 4, 5 };

jarray[1] = new int[] { 10, 20, 30 };

jarray[2] = new int[] { 100, 200 };

for (int i = 0; i < jarray.Length; i++) //jarray.Length --> row size 3

{

for (int j = 0; j < jarray[i].Length; j++)

{

Console.Write(" "+jarray[i][j]);

}

Console.WriteLine();

}

}

**Homework :**

1. Move zeros to beginning or end of array. e.g. if given array is { 2 , 10 , 0 ,5 ,3, 0, 4 ,1} then output array should be a. { 0,0,2,10,5,3,4,1} b. {2,10,5,3,4,1,0,0}
2. Display all elements of two dimensional array with dimensions 3X4.
3. Find maximum number from each row of two dimensional array with dimensions ( 3 x 4)
4. Find minimum number from a two dimensional array .

**String classes in C#**

static void Main(string[] args)

{

string str1 = "hello";

string str2 = "this is sample text there";

**str2.Trim(); // remove space from L & R side**

**str2.TrimStart();// remove space from left side**

**str2.TrimEnd();// remove space from right side**

Console.WriteLine(str1.Length);

Console.WriteLine(str2.Length);

string str3=**string.Concat(str1," ",str2);**

// string str3 = str1 +" "+ str2;

Console.WriteLine(str3);

**Console.WriteLine(str3.ToUpper());**

**Console.WriteLine(str3.ToLower());**

char[] ch= **str3.ToCharArray();**

foreach (var item in ch)

{

Console.WriteLine(item);

}

**bool res= str3.StartsWith("hello");**

**bool res1= str3.EndsWith("text");**

Console.WriteLine(res);

Console.WriteLine(res1);

}

**StringBuilder class**

**Use using System.Text; namespace to work with StringBuilder class**

static void Main(string[] args)

{

double salary = 33000.56;

StringBuilder sb=new StringBuilder("hello to all");

sb.Append("welcome to C# class");

sb.AppendFormat("salary is {0:c}",salary);

Console.WriteLine(sb);

sb.Insert(12, "Good Morning");

Console.WriteLine(sb);

//sb.Remove(0, 12);

sb.Replace("Good Morning", "Good Day");

Console.WriteLine(sb);

}

[**https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings**](https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings)

String class → immutable

StringBuilder class → mutable

static void Main(string[] args)

{

string str ="hello";

Console.WriteLine(str);

Console.WriteLine(str.GetHashCode());// returns the add loc

str = str + " good morning";

Console.WriteLine(str);

Console.WriteLine(str.GetHashCode());

StringBuilder sb = new StringBuilder("hello");

Console.WriteLine(sb);

Console.WriteLine(sb.GetHashCode());

sb.Append(" good morning");

Console.WriteLine(sb.GetHashCode());

Console.WriteLine(sb);

}

namespace TestProject

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public int Price { get; set; }

}

public class Program

{

static void Main(string[] args)

{

Product[] productlist = new Product[3]

{

new Product{Id=1,Name="mouse",Price=999},

new Product{Id=2,Name="keyboard",Price=1999},

new Product{Id=3,Name="pendrive",Price=2999}

};

foreach (Product p in productlist)

{

Console.WriteLine($"{p.Id} {p.Name} {p.Price}");

}

}

}

}

Book → Id, Name,Publisher,price

Course → Id, Name, Duration, Fees

**Collections in C#**

**What are collections ?**

When we want to process object or collection of objects dynamically, we can use collections

Why ?

Two limitations

1. Array is fixed size
2. Size need to specify at compile time

To process collections / group / list at run time

Two types of collections

1. Non generic collections
2. Generic collections
3. Non -generic collections
   1. In non generic collections, we can process different type of data
   2. In non generic collections data get convert from value type to the reference type
   3. To retrieve data again reference type needs to convert to value type
4. Conversion from value type to the reference type ( object) → boxing → implicitly
5. Conversion from reference type to the value type → unboxing → explicit

Classes

1. ArrayList
2. Stack
3. Queue
4. Hashtable

Interfaces

1. IEnumerable
2. IEnumerator
3. IComparable
4. **ArrayList**

Arraylist is the same as array, as it will also store data in contiguous memory locations, but ArrayList can process data dynamically. Memory get also grow & shrink based on data that we add or remove from it

Add namespace → System.Collections

static void Main(string[] args)

{

ArrayList list = new ArrayList();

list.Add(10); // boxing

list.Add(22.22);

list.Add("hello");

double d = Convert.ToDouble(list[1]);

int a = Convert.ToInt32(list[0]); // unboxing

}

static void Main(string[] args)

{

int[] arr = new int[] { 1, 2, 3 };

ArrayList list = new ArrayList();

list.Add(45); // boxing

list.Add(67);

list.Add(11);

list.Add(88);

list.Insert(1, 15);

// list.Remove("hello");

list.RemoveAt(3);

list.Sort();

list.Reverse();

list.AddRange(arr);

list.RemoveRange(2, 3);

// list.Clear();

foreach (var item in list)

{

Console.WriteLine(item);

}

Console.WriteLine($"Total elements {list.Count}");

}

Stack → LIFO (Last in first out)

static void Main(string[] args)

{

// LIFO (list in first out)

Stack s = new Stack();

s.Push(10);// add to stack

s.Push(20);

s.Push(30);

s.Pop(); // remove the last element from the collections

foreach (var item in s)

{

Console.WriteLine(item);

}

Console.WriteLine("last element "+s.Peek()); // display the last element

Queue // FIFO (first in first out)

Queue q = new Queue();

q.Enqueue(10);

q.Enqueue(20);

q.Enqueue(30);

q.Dequeue(); // remove the first element 10

foreach (var item in q)

{

Console.WriteLine(item);

}

Console.WriteLine("first element in queue "+q.Peek());// return the first element

}

**Hashtable**

**Is used to store data in the form of key-value pair**

**Country id country name**

**Mobile num - name**

**Roll no name**

**Hashtable internally uses hash algorithm to store & manage data, at the time of add it will sort the data & then store**

**We can not insert in between in the Hashtable**

**Key should be unique in the hashtable**

**We can remove data from hashtable using key**

**To read data from hashtable we will use built in struct DictionaryEntry**

static void Main(string[] args)

{

Hashtable ht = new Hashtable();

ht.Add(1, "USA");

ht.Add(2, "UK");

ht.Add(63, "AUS");

ht.Add(91, "IND");

ht.Remove(2);

foreach (DictionaryEntry item in ht)

{

Console.WriteLine(item.Key+" "+item.Value);

}

}

**Generic Collections :**

**What is generic ? → similar type of objects, that need to be processed**

1. **Type safe data can be process dynamically(run time) using generic collections**
2. **Generic collections does not need boxing & unboxing**
3. **Save the time of boxing & unboxing ( when we will manage large amount of data)**

**List<T>**

**Stack<T>**

**Queue<T>**

**Dictionary<T,T>**

**T - it is a type for placeholder**

**Type → int, string, float, double, class,**

**Need to add namespace → using System.Collections.Generic**

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

**{**

**List<int> ilist = new List<int>();**

**ilist.Add(10);**

**ilist.Add(20);**

**ilist.Add(30);**

**ilist.Insert(2,15);**

**List<string> slist = new List<string>();**

**slist.Add("C#");**

**slist.Add("MVC");**

**Employee e1 = new Employee("test2", 22000);**

**Employee e2 = new Employee("test3", 22000);**

**List<Employee> emplist = new List<Employee>();**

**emplist.Add(new Employee("test1", 32000));**

**emplist.Add(new Employee("test4", 32000));**

**// or**

**emplist.Add(e1);**

**emplist.Add(e2);**

**// or**

**List<Employee> emplist2 = new List<Employee>()**

**{**

**new Employee("test1", 32000),**

**new Employee("test2", 22000),**

**new Employee("test3", 22000)**

**};**

**foreach (Employee emp in emplist)**

**{**

**Console.WriteLine(emp);**

**}**

**List<Product> prodlist = new List<Product>()**

**{**

**new Product{Id=1,Name="mouse",Price=999},**

**new Product{Id=1,Name="mouse",Price=999},**

**new Product{Id=1,Name="mouse",Price=999},**

**new Product{Id=1,Name="mouse",Price=999},**

**new Product{Id=1,Name="mouse",Price=999},**

**};**

**foreach (Product item in prodlist)**

**{**

**Console.WriteLine($"{item.Id} {item.Name} {item.Price}");**

**}**

**foreach (int item in ilist)**

**{**

**Console.WriteLine(item);**

**}**

**}**

**public class Product**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**public int Price { get; set; }**

**}**

static void Main(string[] args)

{

Dictionary<int, string> dictionary = new Dictionary<int, string>();

dictionary.Add(91, "IND");

foreach (KeyValuePair<int,string> item in dictionary)

{

Console.WriteLine(item.Key+item.Value);

}

}

using System;

using System.Collections;

using System.Collections.Generic;

namespace TestProject

{

// Streams --> CS,IT, Elec

// in each Stream --> no of studs

public class Stud

{

public int Rollno { get; set; }

public string Name { get; set; }

}

public class Stream

{

public int Id { get; set; }

public string Name { get; set; }

public List<Stud> Students = new List<Stud>();

}

public class Program

{

static void Main(string[] args)

{

List<Stream> streams = new List<Stream>() {

new Stream

{

Id=1,

Name="CS",

Students =

{

new Stud{Rollno=1,Name="stud1"},

new Stud{Rollno=2,Name="stud2"},

new Stud{Rollno=3,Name="stud3"},

new Stud{Rollno=4,Name="stud4"},

}

},

new Stream

{

Id=2,

Name="IT",

Students =

{

new Stud{Rollno=5,Name="stud11"},

new Stud{Rollno=6,Name="stud22"},

new Stud{Rollno=7,Name="stud33"},

new Stud{Rollno=8,Name="stud44"},

}

}

};

foreach (Stream st in streams)

{

Console.WriteLine($"{st.Id} {st.Name}");

foreach (Stud s in st.Students)

{

Console.WriteLine($"\t {s.Rollno} {s.Name}");

}

}

}

}

}

**Generic class**

**Generic method**

**Partial class**

**Partial method**

**using System;**

**using System.Collections.Generic;**

**namespace TestProject**

**{**

**// dept --> employess -> show list of dept & working emp from that dept**

**public class Emp**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**public class Dept**

**{**

**public int Did { get; set; }**

**public string DName { get; set; }**

**public List<Emp> Employees = new List<Emp>();**

**}**

**public class Program**

**{**

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

**{**

**List<Dept> deptList = new List<Dept>()**

**{**

**new Dept**

**{**

**Did = 101,**

**DName = "HR",**

**Employees =**

**{**

**new Emp{Id=1,Name="user1"},**

**new Emp{Id=2,Name="user2"},**

**}**

**},**

**new Dept**

**{**

**Did = 102,**

**DName = "Development",**

**Employees =**

**{**

**new Emp{Id=3,Name="user3"},**

**new Emp{Id=4,Name="user4"},**

**new Emp{Id=5,Name="user5"},**

**new Emp{Id=6,Name="user6"},**

**}**

**},**

**new Dept**

**{**

**Did = 104,**

**DName = "Development",**

**Employees =**

**{**

**new Emp{Id=7,Name="user33"},**

**new Emp{Id=8,Name="user43"},**

**new Emp{Id=9,Name="user56"},**

**new Emp{Id=10,Name="user67"},**

**}**

**}**

**};**

**foreach (Dept d in deptList)**

**{**

**Console.WriteLine(d.Did+" "+d.DName);**

**foreach (Emp e in d.Employees)**

**{**

**Console.WriteLine($"\t {e.Id} {e.Name}");**

**}**

**}**

**}**

**}**

**}**

**Homework**

**Branch → students**

**CS → s1,2,3,4,5,6,**

**IT → s10,11,12,**

**Account → Customer → Loan**

**Saving → c1**

**Home**

**Car**

**→ c2**

**Home**

**→ c3**

**Car**

**Personal**

**→ c4**

**Personal**

**Current**

**-> c5,c6,....business loan**

If we want to process a value from the class -> set , get the value

Int, string, emp….

**Generic class**

using System;

using System.Collections;

namespace TestProject

{

public class MyGeneric<T> // T - placeholder for the type

{

private T data;

public string value;

public void Accept(T data)

{

this.data = data;

}

public T GetData()

{

return data;

}

}

public class Program

{

static void Main(string[] args)

{

MyGeneric<int> myGeneric1 = new MyGeneric<int>();

myGeneric1.Accept(10);

MyGeneric<Employee> myGeneric2=new MyGeneric<Employee>();

myGeneric2.Accept(new Employee("test", 20000));

MyGeneric<string> s = new MyGeneric<string>();

s.Accept("Hello");

}

}

}

**Generic method**

public class Program

{

static void Swap<T>(T a, T b)

{

T c;

c = a;

a = b;

b = c;

Console.WriteLine(a);

Console.WriteLine(b);

}

static void Main(string[] args)

{

Employee e1 = new Employee("test1", 2000);

Employee e2 = new Employee("test1", 2000);

Swap<int>(10, 20);

Swap<double>(45.67, 78.66);

Swap<string>("Hello", "hi");

Swap<Employee>(e1, e2);

}

}

Interface → IEnumerable, IEnumerator, IComparable

IEnumerable<T> , IEnumerator<T>

using System;

using System.Collections.Generic;

using System.Collections;

namespace TestProject

{

public class Player

{

private int runs;

private string name;

public Player(int runs, string name)

{

this.runs = runs;

this.name = name;

}

public override string ToString()

{

return "Player :"+name +" "+runs;

}

}

public class Team:IEnumerable

{

private Player[] players;

public Team()

{

players=new Player[3];

players[0] = new Player(12000, "Virat");

players[1] = new Player(13000, "Sachin");

players[2] = new Player(11000, "Dhoni");

}

public IEnumerator GetEnumerator()

{

return players.GetEnumerator();

}

}

public class Program

{

static void Main(string[] args)

{

Team team= new Team();

foreach (var item in team)

{

Console.WriteLine(item);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Collections;

namespace TestProject

{

public class Player:IComparable

{

private int runs;

private string name;

public Player(int runs, string name)

{

this.runs = runs;

this.name = name;

}

public int CompareTo(object obj) // obj --> rohit details

{

// type casting

Player p1 = (Player)obj; // p1 --> rohit

if (this.runs > p1.runs) // this --> refer to current object (Virat)

{

return 1;

}

else if (this.runs < p1.runs)

{

return -1;

}

else

{

return 0;

}

}

public override string ToString()

{

return "Player :"+name +" "+runs;

}

}

public class Team:IEnumerable

{

private Player[] players;

public Team()

{

players=new Player[3];

players[0] = new Player(12000, "Virat");

players[1] = new Player(13000, "Sachin");

players[2] = new Player(11000, "Dhoni");

}

public IEnumerator GetEnumerator()

{

return players.GetEnumerator();

}

}

public class Program

{

static void Main(string[] args)

{

Team team= new Team();

foreach (var item in team)

{

Console.WriteLine(item);

}

Player virat = new Player(13000, "Virat");

Player rohit = new Player(13000, "Rohit");

int result=virat.CompareTo(rohit);

if (result == 1)

{

Console.WriteLine("Virat have more runs than Rohit");

}

else if (result == -1)

{

Console.WriteLine("Rohit have more runs than Virat");

}

else

{

Console.WriteLine("Virat & Rohit have equal runs");

}

}

}

}

**Implement IComparable interface with Product class & check which product have more price than other**

**public class TestCompare : IComparer**

**{**

**public int Compare(object x, object y)**

**{**

**Player p1 = (Player)x; // virat**

**Player p2 = (Player)y; // rohit**

**if (p1.Runs > p2.Runs)**

**{**

**return 1;**

**}**

**else if (p1.Runs < p2.Runs)**

**{**

**return -1;**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**}**

Player virat = new Player(13000, "Virat");

Player rohit = new Player(13000, "Rohit");

// int result=virat.CompareTo(rohit);

TestCompare t = new TestCompare();

int result = t.Compare(virat, rohit);

**Partial class→**

It is also called incomplete class

Partial Class A{

M1();

M2()

}

**Partial Class A{**

M3()

M4()

**}**

public partial class Calculator

{

public int Add(int a,int b)

{

return a + b;

}

}

public partial class Calculator

{

public int Sub(int a,int b)

{

return a - b;

}

}

**Windows form applications**

**Design + Code**

**Public partial class Login{**

**// design**

**}**

**Public partial class Login{**

**// code**

**}**

**Partial Method example**

using System;

using System.Collections.Generic;

using System.Collections;

namespace TestProject

{

public partial class Calculator

{

public int Add(int a,int b)

{

return a + b;

}

partial void Multiply(int a, int b);

}

public partial class Calculator

{

public int Sub(int a,int b)

{

return a - b;

}

partial void Multiply(int a, int b)

{

Console.WriteLine(a\*b);

}

public void Test(int a,int b)

{

Multiply(a, b);

}

}

public class Program

{

static void Main(string[] args)

{

Calculator c = new Calculator();

c.Test(4, 6);

}

}

}

**Exception Handling :**

What is exception ? → the errors which comes at run time of program is called exception

Run time error is called exception

Why run time errors occur in the program

1. Due to wrong input → enter 2 no for div → 4 , 0
2. Incorrect logic -> loop, expression int i=1 while(i>10){ —-}
3. Permission
4. Access the resources → File

What is exception handling

To handle run time errors (exception) is called exception handling

CLR will manage exception handling in C#

Why to handle exception

1. Normal termination of program
2. Save the application data
3. Provides meaningful message to the user

How to implement in C#

**Try**  → put the code which lead to run time error

**Catch →** is used to handle the exception which is raised by try block

**Finally →** code that you want to execute even exception occur in the program or not occur in the program

1. It is used to free the resources like File or DB
2. Close the file
3. Close the DB connection

**Throw →** when we want to throw an exception manually or explicitly

We can throw system exceptions & user defined exception also

Method1(){

Try{

Method2();

}

catch(Exception ex)

{

}

…………………..

…………………

}

Method2(){

—-----------

—---------

Throw new Exception(“...........”);

}

1. We can write multiple try & catch block in C#
2. We can write multiple catch for a single try block
3. We can nest try within try block
4. **But we can not nest catch block**
5. We can not use only try in C#
6. Either we need catch or finally with try block
7. Finally is not mandatory to write with try & catch

**Exception**

**SystemException**

**ArithmaticException**

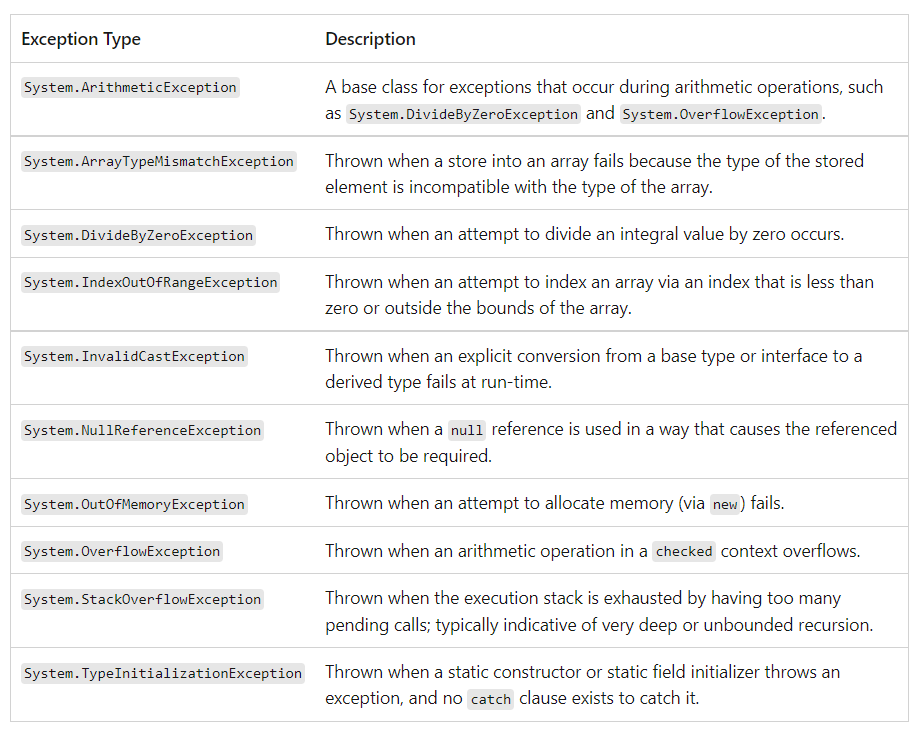
**DivideByZeroException**

**ArgumentExcetion**

**FileNotFoundException**

**IndexOutOfRangeException**

**ApplicationException**



static void Main(string[] args)

{

int a = 0,b = 0;

try

{

a = Convert.ToInt32(Console.ReadLine());

b = Convert.ToInt32(Console.ReadLine());

int c = a / b;

Console.WriteLine(c);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Console.WriteLine("in the finally block");

}

}

**User defined Exception / Custom Exception**

In C# we can create our own exception class to handle any particular business need or to implement any business logic

User age > 18 -> valid

Password -> length should be min 8 chars → exception

Marks -> max -100

using System;

namespace TestProject

{

// create custom exception to handle Age

public class AgeException: Exception

{

public AgeException(string message):base(message)

{

}

}

public class User

{

public void AcceptAge(int age)

{

if (age < 18)

{

throw new AgeException("You are not eligible");

}

}

}

public class Program

{

static void Main(string[] args)

{

User user = new User();

try

{

user.AcceptAge(15);

}

catch(AgeException ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

**In the user class accept name from constructor & if name is null or empty then raise an exception “name is required”**

using System;

namespace TestProject

{

// create custom exception to handle Age

public class AgeException: Exception

{

public AgeException(string message):base(message)

{

}

}

public class NameException : Exception

{

public NameException(string message) : base(message)

{

}

}

public class User

{

private string name;

public User(string name)

{

if (string.IsNullOrEmpty(name))

{

throw new NameException("name is required");

}

else

{

this.name = name;

}

}

public void AcceptAge(int age)

{

if (age < 18)

{

throw new AgeException("You are not eligible");

}

}

}

public class Program

{

static void Main(string[] args)

{

try

{

User user = new User("test");

user.AcceptAge(25);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

**Multithreading:**

**Process →** which have collection of task

MS-Word document , VS, VLC media

**Multiprocessing →** collection of multiple process (OS)

**Thread →** allocation of single task

Word document

Auto correct the word

Spl & grammar check

Auto save

**Multi threading →** we can concurrently execute the multiple threads , which are not dependant on each other

**Why multithreading ?**

* Save the app time
* The task which are not dependent can run concurrently

**How to implement it ?**

**Thread Life cycle**

1. **Unstarted**
2. **Ready to run**
3. **Runnable / Running**
4. **Not Running**
5. **Terminate**
6. **CPU** → exe
7. **Memory →** RAM (store app data)
8. **IO devices →** keyboard , mouse, monitor , speaker

**Context switch between the process**

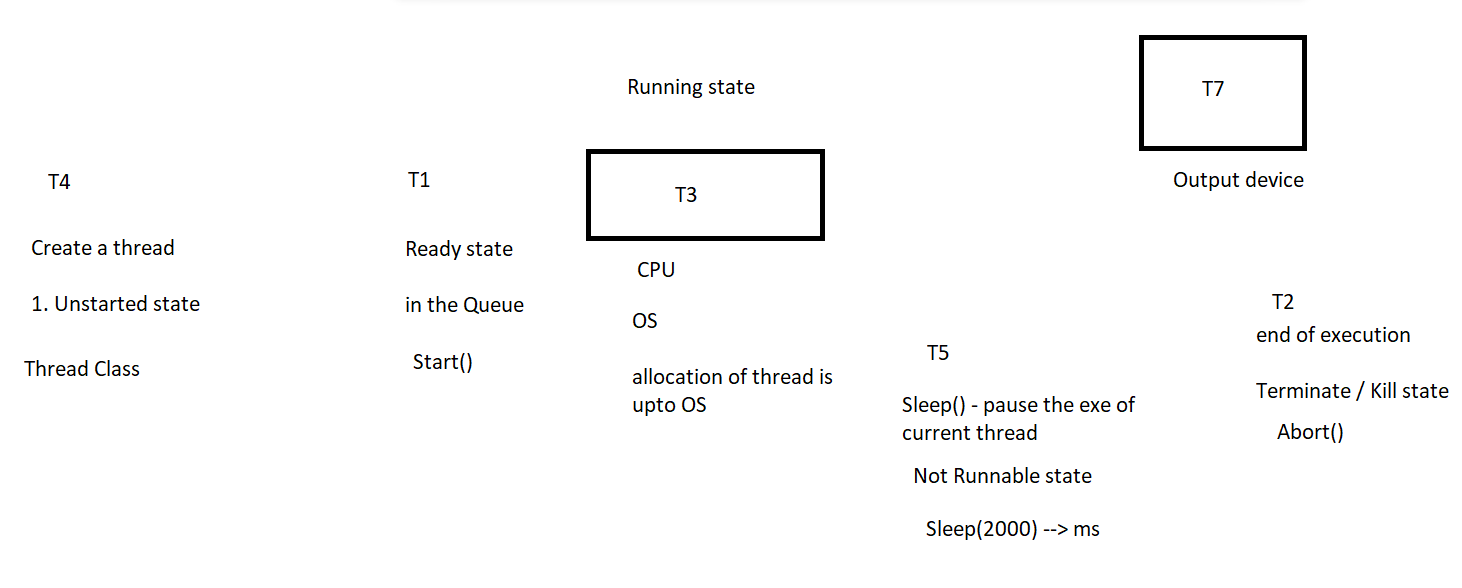
**VS 1ms (time slice) P2**

**Word 1ms P2**

**Zoom 1ms P2**

**Whats app 1ms P3**

**Antivirus 1ms P1**

****

**Namespace → System.Threading**

using System;

using System.Threading;

namespace TestProject

{

public class Test

{

public void Run1()

{

Thread currentThread = Thread.CurrentThread;

Console.WriteLine(currentThread.Name);

for (int i= 1;i<=5; i++)

{

Console.WriteLine(i);

Thread.Sleep(2000);

}

}

public void Run2()

{

Thread currentThread = Thread.CurrentThread;

Console.WriteLine(currentThread.Name);

for (int i = 6; i <= 10; i++)

{

Console.WriteLine(i);

Thread.Sleep(2000);

}

}

}

public class Program

{

static void Main(string[] args)

{

Test test = new Test();

// create a thread

// Thread class --> to create a thread

// ThreadStart delegate -> hold method reference

Thread t1 = new Thread(new ThreadStart(test.Run1));

Thread t2 = new Thread(new ThreadStart(test.Run2));

t1.Name = "Thread 1";

t2.Name = "Thread 2";

t1.Priority = ThreadPriority.Highest; // request to OS not order

t2.Priority = ThreadPriority.Lowest;

t1.Start();

t2.Start();

}

}

}

**\*\* Join() of Thread →** to complete the execution of current thread & then start the execution of other threads

**t1.Join();**

**Thread Synchronization**

When threads approach the critical section of a program, then that code needs to be executed in a sequence , then we can use thread synchronization.

**Lock** keyword will be used to access the block synchronously

public void Run2()

{

lock (this) // this refers to the current thread

{

Thread currentThread = Thread.CurrentThread;

Console.WriteLine(currentThread.Name);

for (int i = 6; i <= 10; i++)

{

Console.WriteLine(i);

}

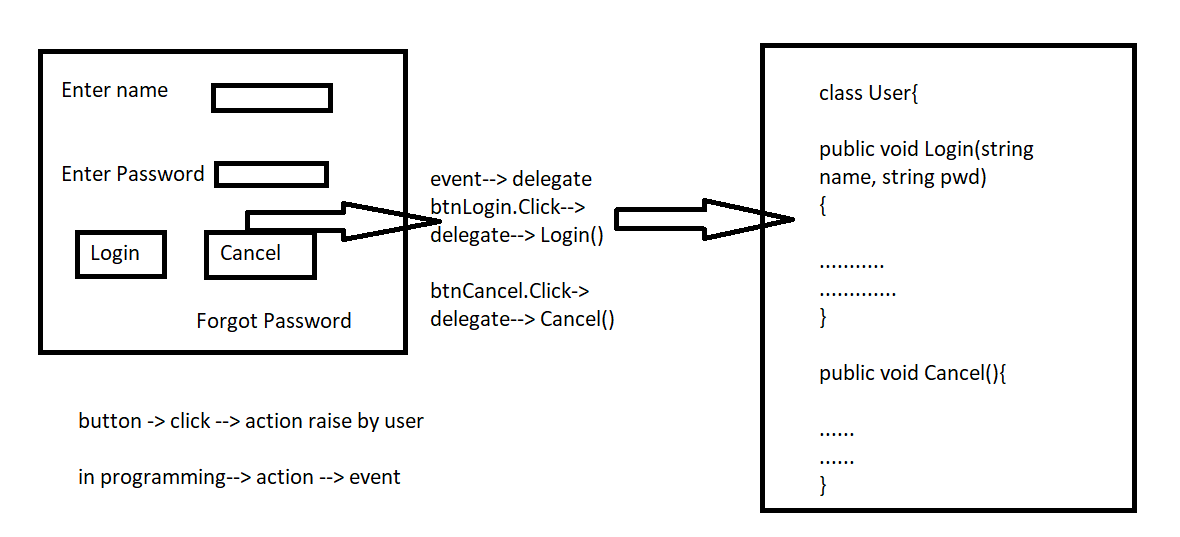
}

}

**Delegates & Events in C#**

Delegates & events are the foundation of GUI (graphical user interface) programming

Graphical elements → visual elements like (textbox, image, button, label….., icons)



Delegate in C#

1. Delegate is a reference type in C#
2. Delegate is used to hold method reference (address)
3. **Delegate also called as type safe pointer in C#**
4. Delegate syntax & method syntax needs to be same (parameters & return type)
5. Delegate needs to be declared in the namespace

Types of Delegate

1. Single cast delegate → single delegate hold single method reference
2. Multicast delegate → single delegate hold multiple method references

Methods get added in the invocation list

Methods get execute in the order which is in list

1. When we want to execute methods in a sequence / order we use multicast delegate

Laptop – >start -> a. RAM b. HDD c. OS

namespace TestProject

{

// declaration of delegate

public delegate int MyDelegate(int a,int b);

public class Calc

{

public int Add(int a,int b)

{

return a + b;

}

}

public class Program

{

static void Main(string[] args)

{

Calc calc = new Calc();

MyDelegate myDelegate = new MyDelegate(calc.Add);// added the method ref

int sum=myDelegate.Invoke(45, 23);

Console.WriteLine(sum);

}

}

}

namespace TestProject

{

// declaration of delegate

public delegate int MyDelegate(int a,int b);

public class Calc

{

public int Add(int a,int b)

{

return a + b;

}

public int Sub(int a, int b)

{

return a - b;

}

public int Multiply(int a, int b)

{

return a \* b;

}

}

public class Program

{

static void Main(string[] args)

{

Calc calc = new Calc();

MyDelegate myDelegate = new MyDelegate(calc.Add);// added the method ref

myDelegate += new MyDelegate(calc.Multiply);

myDelegate += new MyDelegate(calc.Sub); // adding ref to the list

myDelegate -= new MyDelegate(calc.Add); // remove the method from the list

// working with invocation list of delegate

Delegate[] list= myDelegate.GetInvocationList();

foreach (Delegate d in list)

{

Console.WriteLine(d.Method);

Console.WriteLine(d.DynamicInvoke(45,34));

}

}

}

}

**Homework** 👍

Create a class which should have two methods UpperCase & LowerCase which accept the string & converts it in to the respective case & return the string again.

Hold method references using delegate & call & exe using invocation list

Events : an action raised by user is called an event in the program

Like → click on the button

Change value from textbox

Move the mouse

Press a key

To work with event following steps needs to do

1. Declaration of event
2. Raise an event
3. Bind event with delegate to invoke a method

namespace TestProject

{

public delegate void MyDelegate();

public class Stud

{

public event MyDelegate Pass; // declare an event

public event MyDelegate Fail;

public void AcceptMarks(int marks)

{

if (marks < 40)

{

Fail(); // raise an event

}

else

{

Pass(); // raise an event

}

}

}

public class Program

{

static void PassMsg()

{

Console.WriteLine("You are pass");

}

static void FailMsg()

{

Console.WriteLine("You are Fail");

}

static void Main(string[] args)

{

Stud s = new Stud();

s.Pass+=new MyDelegate(PassMsg);

s.Fail += new MyDelegate(FailMsg);

s.AcceptMarks(65);

}

}

}

Assignment

1. Create class Bank with two methods debit & credit
2. Credit method will add amount to the account balance, & then show the updated balance
3. Debit method will debit the amount from the account & display the current balance
4. If debit amount > balance then raise an event “insufficient balance”
5. If balance is 0 then raise an event zero balance
6. If balance is < 3000 then raise an event low balance

**SQL (Structure query Language)**

DBMS - Database management system

What is Data ? →

In database the smallest portion of information is called data

E.g. stud rollno emp id , name, ….

DBMS → a database is an organized collection of data , so that it can be easily accessed & managed

Features:

1. DBMS will provide the interface to perform various operations like insert, update , delete
2. Allows to create DB based on the client / users requirement
3. Provides specific data
4. Provides the security

Advantage of database

1. Data sharing→ an authorized user can share data among multiple users (database admin)
2. Backup→ provides backup & recovery system create automatic data from system failure

RDBMS → relational database management system

Proposed by E F Codd

Store & manage data from the table (row & cols)

Terms used in RDBMS

1. Instance / table → table with row & col
2. Schema → defines the structure like name of the table, each col in table….
3. Field → field is a smaller entity of the table which contains specific information
4. Column → vertical entity in the table
5. Null values → null value of the table specifies that the field has been blank (not a zero , not a space)

Terms

1. Data integrity → it specifies that there should be no duplicate rows in a table
2. Domain integrity → valid entries for a given column by restricting type, format or range

Mo no…, email id , 18 -60

1. Referential integrity → row can not be deleted which is used by the other record

Dept (table)

deptId name

1 HR

2 Sales

Emp (table)

Empid empname salary deptid

101 abc 100 1

102 xyz 100 2

1. User defined integrity → enforces some business rule that defined by the user

Ac bal < 5000 saving

Ac bal < 10000 current

Age > 18

SQL

Types of SQL queries

1. DDL (data definition lang)
2. DML (data manipulation lang)
3. DQL (data query lang)
4. TCL (transactional control lang)
5. DCL (data control lang)
6. DDL ( data definition language)
7. Sql command deal with description of database schema
8. Used to create , modify or delete the structure of database

Create -> DB , table, index, function, view, stored procedure, triggers

Drop → delete the DB or table

Alter → change the structure of Db or table

Truncate → remove all the records from the table, **including space**

--create database Tesla

-- is comment in sql

--use Tesla

create table Person

(

id int,

fullname varchar(20)

)

-- add new col

alter table person add city varchar(25)

-- modify the size & datatype

alter table person alter column city varchar(30)

-- in oracle

alter table person modify city varchar(30)

alter table person drop column city

-- rename the col name

sp\_rename 'person.fullname', 'name'

-- drop the table

drop table Person

-- drop the database

drop database Tesla

-- remove all records from the table

truncate table perosn

**Constraints:**

Are used to define some rules on the column, constraints can be applied with create & alter

1. Not null
2. Unique
3. Primary key
4. Foreign key
5. Check
6. Default
7. Index
8. Not null→ required field ( column can not have null value) like fname, email, pwd, contact
9. Unique → the column should not have duplicate value, but unique accept null value

create table emp(

email varchar(30) not null,

contact varchar(10) not null

)

alter table emp alter column email varchar(30) not null

alter table emp add unique(email)

--or

alter table emp add unique(email,contact)

-- or

alter table emp add constraint un\_emp unique(email)

--or

alter table emp add constraint unique\_emp unique(email,contact)

alter table emp drop constraint un\_emp

sp\_help emp

1. **Primary key →**  combination of not null & unique constraint

Column value should not be duplicate & should not be null

E.g. Id , RollNo, EmpId

create table emp(

Id int primary key,

email varchar(30) not null,

contact varchar(10) not null

)

--or

create table emp(

Id int,

email varchar(30) not null,

contact varchar(10) not null,

constraint PK\_emp primary key(Id)

)

alter table emp add constraint PK\_emp primary key(Id)

alter table emp drop constraint PK\_emp

1. **Foreign Key →** a key which uniquely identifies key in another table

A primary key which is refer in another table is called FK

**Purpose of FK constraint is i will allow to insert the value which is present in the primary key table**

create table Dept(

deptid int primary key,

deptname varchar(10)

)

create table emp(

Id int primary key,

email varchar(30) not null,

contact varchar(10) not null,

deptid int,

constraint FK\_emp\_dept foreign key(deptid) references dept(deptid)

)

alter table emp add constraint FK\_emp\_dept foreign key(deptid) references dept(deptid)

alter table emp drop constraint FK\_emp\_dept

1. **Check constraint → is used to apply certain validation on the column , if the given condition is true then data will be added to column otherwise not**

E.g. age >18

create table emp(

Id int primary key,

email varchar(30) not null,

contact varchar(10) not null,

age int check(age>18)

)

--or

create table emp(

Id int primary key,

email varchar(30) not null,

contact varchar(10) not null,

age int,

constraint check\_age check(age>18)

)

alter table emp add constraint check\_age check(age>18)

alter table emp drop constraint check\_age

1. **Default Constraint →**  to set the default value

E.g default state is Maharashtra

Default country is india

Max score → 100

create table emp(

Id int primary key,

email varchar(30) not null,

contact varchar(10) not null,

country varchar(20) default('india')

)

alter table emp add constraint default\_country default 'india' for country

alter table emp drop constraint default\_country

1. **Index constraint →**  are used to improve the search result. To fetch records quickly

-- allow duplicate data

create index in\_emp on emp (email,contact)

-- it will not allow duplicate data

create unique index in\_emp on emp (Id)

drop index emp.in\_emp

**Identity Column →**  When you want column value should be auto increment based on value provided to it e.g. Id → Id should be auto generate

create table Employee(

id int identity(1,1)

)

1 -> start value

1 -> increment by

**DML (data manipulation language)**

1. **Insert**
2. **Update**
3. **Delete**

select \* from Person

alter table perosn add city varchar(20)

drop table person

create table Person(

personid int primary key identity(1,1),

name varchar(20) not null,

city varchar(20) not null

)

insert into Person values('Amol','Mumbai')

insert into Person values('Ajay','Pune')

insert into Person(name,city) values('Tushar','Delhi')

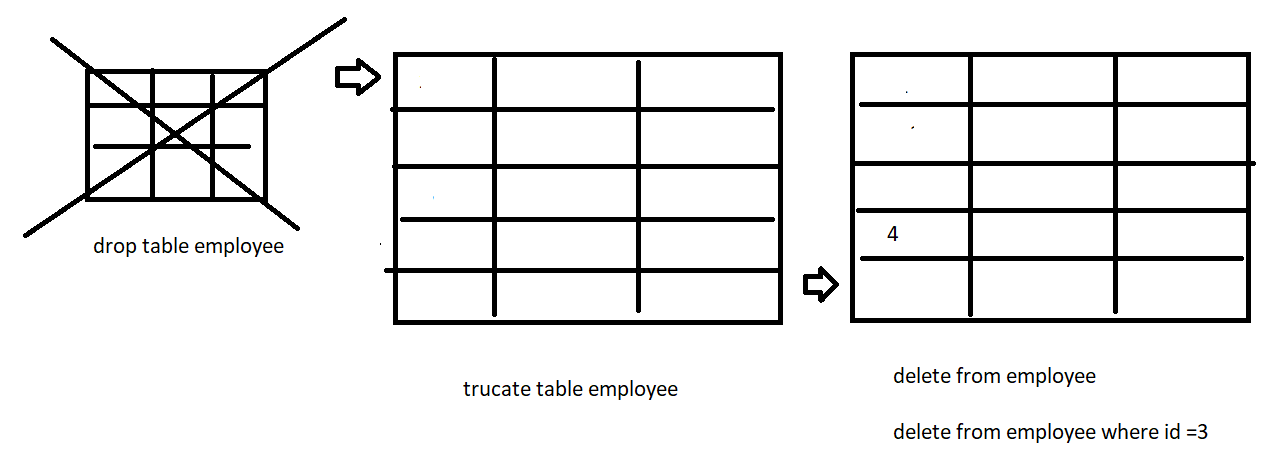
update Person set city='pune' , country='india' where personid=1

update person set country='india' where personid=2

delete from Person where personid=4

delete from Person

Difference between truncate , drop & delete



DQL → Data Query Language

1. Select

To fetch the data from the table we will use select command

* Where clause → to filter the data
* Order by → sort the data

select \* from Person

select name,city from Person

select \* from Person where personid=2

select name,city,age from Person where personid=5

--alise to col name

select name as FullName,city, age as personAge from person where personid=8

--relational operator

select \* from Person where age <27

select \* from Person where age >=27

select \* from Person where age <>30 -- <> not equal

--range using between & and operator

select \* from Person where age between 24 and 27

--in operator & not in operator

-- in is used to select the possible selection of values

select \* from Person where age in (24,27,30)

select \* from Person where city in('pune','nagpur')

select \* from Person where city in('mumbai','nagpur')

select \* from Person where personid in (1,2,8,6)

select \* from Person where age not in (24,27,30)

select \* from Person where city not in('mumbai','nagpur','Delhi')

select \* from Person where city not in('pune','Delhi')

select \* from Person where personid not in (1,2,8,6)

select distinct city from Person

select distinct age from Person

**--search pattern for text using like clause**

select \* from Person where name like 'A%'

select \* from Person where name like '%H'

select \* from Person where name like '%u%'

select \* from Person where name like 'A\_\_\_'

select \* from Person where name like '\_\_\_y'

select \* from Person where name like '\_u\_\_\_'

select \* from Person where name like '[ap]%'

select \* from Person where name like '%[ap]%'

select \* from Person where name like '[h-s]%'

select \* from Person where name not like '[h-s]%'

select \* from Person where name not like '[ap]%'

-- and , or , not operator

select \* from Person where city='pune' and age >25

select \* from Person where city='pune' or city='delhi'

select \* from Person where not city='pune'

update Person set country=null where personid in(9,10)

-- null values --> is null and is not null operator

select \* from Person where country is null

select \* from Person where country is not null

select \* from Person

order by age

select \* from Person

order by age desc

select \* from Person where city='pune'

order by age

select \* from Person where city='pune'

order by personid

select \* from Person where city='pune'

order by personid desc

--aggregate function in sql

-- count, sum, avg, max,min

select count(personid) as TotalCount from Person

select COUNT(\*) as TotalCount from Person -- \* consider the duplicate & null values

select COUNT(distinct city) as totalCity from Person

select COUNT(distinct age) as countage from Person

select sum(age) from Person

select avg(age) as avgAge from Person

select max(age) as maxAge from Person where city='pune'

select min(age) as minAge from Person

**--group by clause**

-- group by statement groups the rows that have same values in to summary

-- group by statement can be used with aggregate function to group the result

-- the name of column that we put in group by that column can be used with select

--statment

select city,count(personid) as TotalPersons from person

group by city

**Having clause**

* Having clause added in SQL to apply condition with aggregate function
* Where clause can not be used with aggregate function
* Having clause can include the column which is exists in group by clause

--group by clause

-- group by statement groups the rows that have same values in to summary

-- group by statement can be used with aggregate function to group the result

-- the name of column that we put in group by that column can be used with select

--statment

select city,count(personid) as TotalPersons from person

group by city

having count(personid)>4

select name,count(personid) as TotalPersons from person

group by name

having avg(age)<27

select city,count(personid) as TotalPersons from person

group by city

having city in('pune','delhi')

select city,count(personid) as TotalPersons from person

group by city

having city in('pune','delhi')

order by COUNT(personid) desc

select city,count(personid) as TotalPersons from person

group by city

having city in('pune','delhi')

order by city desc

**Homework :**

1. Write a statement that will select the City column from the Customers table
2. Select all the *different* values from the Country column in the Customers table.
3. Select all records where the City column has the value "London
4. Use the NOT keyword to select all records where City is NOT "Berlin".
5. Select all records where the CustomerID column has the value 23.
6. Select all records where the City column has the value 'Berlin' *and* the PostalCode column has the value 121110.
7. Select all records where the City column has the value 'Berlin' or 'London'.
8. Select all records from the Customers table, sort the result alphabetically by the column City.
9. Select all records from the Customers table, sort the result *reversed* alphabetically by the column City.
10. Select all records from the Customers table, sort the result alphabetically, first by the column Country, then, by the column City
11. Select all records from the Customers where the PostalCode column is empty.
12. Select all records from the Customers where the PostalCode column is NOT empty.
13. Set the value of the City columns to 'Oslo', but only the ones where the Country column has the value "Norway".
14. Delete all the records from the Customers table where the Country value is 'Norway'.
15. Use the MIN function to select the record with the smallest value of the Price column.
16. Use an SQL function to select the record with the highest value of the Price column.
17. Use the correct function to return the number of records that have the Price value set to 20
18. Use an SQL function to calculate the average price of all products.
19. Use an SQL function to calculate the sum of all the Price column values in the Products table
20. Select all records where the value of the City column starts with the letter "a".
21. Select all records where the value of the City column *ends* with the letter "a".
22. Select all records where the value of the City column contains the letter "a".
23. Select all records where the value of the City column starts with letter "a" and ends with the letter "b".
24. Select all records where the value of the City column does NOT start with the letter "a".
25. Select all records where the *second* letter of the City is an "a".
26. Select all records where the first letter of the City is an "a" or a "c" or an "s".
27. Select all records where the first letter of the City starts with anything from an "a" to an "f".
28. Select all records where the first letter of the City is NOT an "a" or a "c" or an "f".
29. Use the IN operator to select all the records where the Country is either "Norway" or "France".
30. Use the IN operator to select all the records where Country is NOT "Norway" and NOT "France".
31. Use the BETWEEN operator to select all the records where the value of the Price column is between 10 and 20
32. Use the BETWEEN operator to select all the records where the value of the Price column is NOT between 10 and 20.
33. Use the BETWEEN operator to select all the records where the value of the ProductName column is alphabetically between 'Geitost' and 'Pavlova'.
34. When displaying the Customers table, make an ALIAS of the PostalCode column, the column should be called Pno instead.
35. When displaying the Customers table, refer to the table as Consumers instead of Customers.
36. List the number of customers in each country.
37. List the number of customers in each country, ordered by the country with the most customers first.
38. Write the correct SQL statement to create a new database called testDB.
39. Write the correct SQL statement to delete a database named testDB
40. Add a column of type DATE called Birthday in Persons table
41. Delete the column Birthday from the Persons table

Create database testDB

Drop database testDB

Alter table persons add Birthday date

create table dept(

deptid int primary key identity(1,1),

deptname varchar(20)

)

create table employee(

id int primary key identity(1,1),

empname varchar(20),

city varchar(20),

salary decimal,

deptid int,

constraint fk\_employee\_dept foreign key(deptid) references dept(deptid)

)

insert into dept values('HR')

insert into dept values('Development')

insert into dept values('Testing')

insert into dept values('Sales')

insert into employee values('Amol','Mumbai',75000,4)

insert into employee values('Chetan','Pune',35000,1)

insert into employee values('Pratik','Mumbai',32000,2)

insert into employee values('Rahul','Pune',45000,3)

insert into employee values('Rohan','Nagpur',25000,1)

insert into employee values('Suraj','Nagpur',37000,2)

insert into employee values('Rajesh','Nashik',23000,3)

insert into employee values('Kishor','Mumbai',78000,2)

insert into employee values('Ajay','Pune',21000,4)

insert into employee values('Tushar','Delhi',56000,4)

alter table employee add managerid int

update employee set managerid =1 where id in(2,3)

update employee set managerid =6 where id in(4,5)

update employee set managerid =8 where id in(7,9,10)

**Joins in SQL →** get the data from multiple tables

**Types**

1. **Inner join** → select the records that having matching values in both tables
2. **Left Join** → returns all the records from the left table & matching records from the another table, rest records will be display as null

Customer → orders

1 101,102

2 103,102

3 null

1. **Right Join** → returns all the records from the right table & matching records from the another table,
2. **Full join / Full outer join** → returns all the records from the both tables, full join is a combination of left join & right join
3. **Self join** → join the table to itself

-- self join

select e.empname as EmployeeName, m.empname as Managername

from employee e, employee m

where m.id=e.managerid

1. **Cross join / cartesian product** → the cross join is used to generate a paired combination of each row of first table with each row of second table

This type of join is also called cartesian product

update employee set deptid=null where id=10

select e.\*,d.deptname from employee e

inner join dept d on d.deptid=e.deptid

select e.\*,d.deptname from employee e

left join dept d on d.deptid=e.deptid

select e.\*,d.deptname from employee e

right join dept d on d.deptid=e.deptid

select e.\*,d.deptname from employee e

full join dept d on d.deptid=e.deptid

Cross join e.g. → **select \* from breakfast cross join drinks**

**Breakfast table**

1 Poha

2. Upma

3. Idali

**Drink table**

1 Tea

2 Coffee

3. Lemon tea

1 poha tea

2 upma tea

3 idali tea

1 poha coffee

2 upma coffee

3 idali coffee

1 poha lemon tea

2 upma lemon tea

3 idali lemon tea

Flipkart

Red t-shirt M

Red t-shirt L

Red t-shirt XL

Red t-shirt XXL

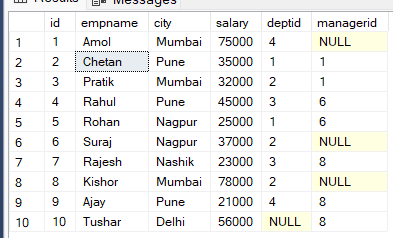
Black t-shirt M

Black t-shirt L

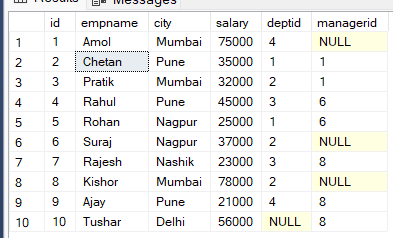
Black t-shirt XL

Black t-shirt XXL

Employee e



Employee m



M.id =e.managerid

**Sub query in SQL**

* **Display the emp who belongs to hr dept**

**Select \* from employee where deptid =(select deptid from dept where deptname=’HR’)**

1. Sub query get executed first & its result will be passed to the outer query / main query

select \* from employee where salary=

(select max(salary) from employee)

select \* from employee where salary=(

select max(salary) from employee where salary <>

(

select max(salary) from employee

)

)

**Homework:**

-- display emp details whose salary is > avg salary of all emps

-- display the emp who is taking higher salary than empid =5

-- display emp who have salary > avg salary of sales dept

-- display emp who salary > avg salary of each dept

**View in SQL**

1. View is a virtual table based on result set of an sql statement
2. View contains rows & columns same as the original table

create view EmpFromHr as

Select \* from employee where deptid =

(select deptid from dept where deptname='HR')

create view EmpFromDev as

Select \* from employee where deptid =

(select deptid from dept where deptname='Development')

-- execute the view / call the view

select \* from EmpFromHr

select \* from EmpFromDev

**Stored Procedure**

**Function**

**Triggers**

**Normalization**

**Stored Procedure:**

1. SP is a prepared query
2. Also called prepare SQL code
3. Access the RDBMS
4. SP uses data dictionary
5. Can accept the input parameters & return the multiple values
6. I.e. with the help of output parameters

Benefits of SP

1. Reduce the server / client traffic
2. Stronger security
3. Reuse the code
4. Improve the performance

create proc SP\_insert\_dept(

@deptname varchar(20)

)

as begin

insert into dept values(@deptname)

return

end

-- alter is used to modify the existing proc

alter proc SP\_insert\_dept(

@deptname varchar(20)

)

as begin

insert into dept values(@deptname)

return

end

-- call the SP

exec SP\_insert\_dept

@deptname='Test'

select \* from employee

create proc SP\_select\_employee(

@id int

)

as begin

select \* from employee where id=@id

return

end

exec SP\_select\_employee

@id=10

* **Write SP for update the dept**
* **Write SP for delete the dept**

create proc SP\_update\_dept(

@deptname varchar(20),

@deptid int

)

as begin

update dept set deptname=@deptname where deptid=@deptid

return

end

exec SP\_update\_dept

@detpname='test2'

@deptid=5

create proc SP\_delete\_dept(

@deptid int

)

as begin

delete from dept where deptid=@deptid

return

end

exec SP\_delete\_dept

@deptid=5

Difference between View & SP

View is a simple showcasing the data from table, where as SP is a group of statements that can be executed

A view is faster as it display data from the table, where as SP executes the statements

**Functions in SQL**

1. Built in functions
2. User defined functions

-- substring

select SUBSTRING(empname,1,3) as Tempname from employee where id=1

--concat

select CONCAT(empname,' ',city) as tempcol from employee

--len

select empname,len(empname) as LengthOFName from employee

--upper

select UPPER(empname) from employee

select LOWER(empname) from employee

--' Amol'

select LTRIM(empname) from employee

--'Amol '

select RTRIM(empname) from employee

--' Amol '

select TRIM(empname) from employee

select ROUND(salary,2) as Salary from employee

User defined functions

2 types

1. Which returns the scalar value
2. Which returns the result set as table ( multi value )

alter function check\_name\_emp

( @empname varchar(20)

)

returns varchar(5) as

begin

declare @return\_value varchar(5);

if(len(@empname)<5) set @return\_value='false';

else set @return\_value ='true';

return @return\_value

end

select empname ,dbo.check\_name\_emp(empname) from employee

alter function add\_salary\_employee

(

@salary decimal

)

returns decimal

as begin

declare @newSalary decimal;

set @newSalary =@salary +1350;

return @newSalary

end

select empname, salary , dbo.add\_salary\_employee(salary) from employee

Product → price → pass to function & 10 % discounted value

100 → function → 100 -> 10% → 90

Prodname price discounted price

Mouse 100 90

**Normalization**

<https://www.javatpoint.com/dbms-normalization>

<https://www.geeksforgeeks.org/normalization-process-in-dbms/>

<https://www.tutorialspoint.com/dbms/database_normalization.htm>

1NF → all the column should have atomic value

2NF→ table should be in 1NF, all the non key columns should have dependency on PK col

3NF → table should be in 2NF , remove all the non key col which are dependent on non key col

BCNF→ all the function dependency (all the col ) should dependant on PK col

Userid(pk) user name email contact addressid

101 abc email 1223 1

Address id userid zipid

1 101 1

Fixed ( master table in sql)

City id city name state id

1 pune 1

2 nagar 1

3 Abad 1

4 Mumbai 1

5 Suraj 2

State id state name

1 maha

2 Gujrat

3 Delhi

4 MP

5 Raj

Id Zipid stateid

1 431001 1

Student , Trainer → Skills

Sid sname

101 Amol

102 Suraj

Trainrerid trainer name

1001 Tushar

1002 Ajay

Skill id skill name

1 C#

2 java

3 MVC

4 angular

Trainer\_skill\_map

Id skillid trainer id

10 1 1001

20 2 1002

30 4 1001

Stud\_trinaer\_map

Id studid trainer id

1 101 1001

2 102 1002

**Homework:**

Create a table structures for e-comm website

Users

Add to cart

Orders

Roles -> customer / admin

Products

**Triggers**

1. Trigger also called as special SP
2. That will automatically executes in response

Types

1. DML
2. DDL
3. Logon trigger

2 types of triggers

1. After trigger
2. Instead trigger

create trigger tr\_emp\_forinsert

on emp for insert

as begin

declare @id int

select @id=id from inserted

insert into empaudit values('new emp with id '+ cast(@id as varchar(5))

+ ' is added at '+ cast(getdate() as varchar(20)))

end

create trigger tr\_emp\_fordelete

on emp for delete

as begin

declare @id int

select @id=id from deleted

insert into empaudit values('new emp with id '+ cast(@id as varchar(5))

+ ' is removed at '+ cast(getdate() as varchar(20)))

end

create table emp

(

id int primary key identity(1,1),

name varchar(20),

contact varchar(10)

)

--to log emp table

create table empaudit

(

auditid int primary key identity(1,1),

description varchar(100)

)

**Update trigger**

CREATE trigger [dbo].[tr\_employee\_forup]

on [dbo].[employeeTemp] after update

as begin

declare @id int

declare @oldname varchar(20),@newname varchar(20)

declare @olddesignation varchar(20), @newdesignation varchar(20)

declare @oldsalary decimal,@newsalary decimal

declare @auditstr varchar(1000)

select \* into #TempTable from inserted

while(Exists(select Id from #TempTable))

begin

set @auditstr=''

select Top 1 @id=Id,@newname=Name,@newdesignation=Designation,@newsalary=Salary

from #TempTable

select @oldname=Name,@olddesignation=Designation,@oldsalary=Salary from deleted

where Id=@id

set @auditstr='emp with id '+ cast(@id as varchar(5))+'change'

if(@oldname<>@newname)

set @auditstr=@auditstr+' old name '+@oldname +'new name '+@newname

if(@olddesignation<>@newdesignation)

set @auditstr=@auditstr+' old designation '+@olddesignation +'new designation '+@newdesignation

if(@oldsalary<>@newsalary)

set @auditstr=@auditstr+' old salary '+cast(@oldsalary as varchar(50)) +'new salary '+cast(@newsalary as varchar(50))+''

insert into employeeaudit values(@auditstr)

delete from #TempTable where Id=@id

end

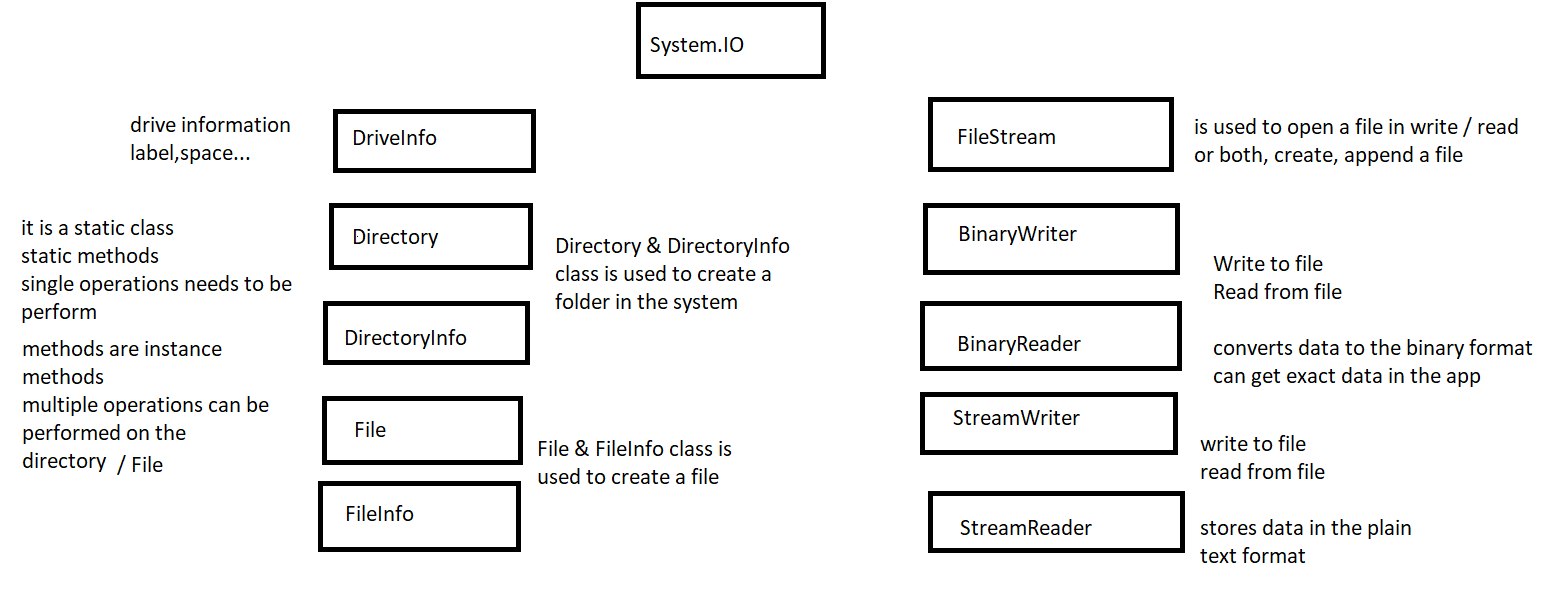
End

File IO -> input / output

Write

Read

ReadWrite

****

**using System.IO;**

private void btnCreateFolder\_Click(object sender, EventArgs e)

{

try

{

string path = @"D:\Tesla";

if (Directory.Exists(path))

{

MessageBox.Show("Folder exists");

}

else

{

Directory.CreateDirectory(path);

MessageBox.Show("Folder created");

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnCreateFile\_Click(object sender, EventArgs e)

{

try

{

string path = @"D:\Tesla\sample.txt";

if (File.Exists(path))

{

MessageBox.Show("File exits");

}

else

{

File.Create(path);

MessageBox.Show("File created");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

FileStream fs;

BinaryWriter bw;

BinaryReader br;

private void btnBinaryWrite\_Click(object sender, EventArgs e)

{

try

{

fs=new FileStream(@"D:\Tesla\emp.dat",FileMode.Create,FileAccess.Write);

bw=new BinaryWriter(fs);

bw.Write(Convert.ToInt32(txtId.Text));

bw.Write(txtName.Text);

bw.Write(Convert.ToDouble(txtSalary.Text));

MessageBox.Show("Data added to file");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

bw.Close();

fs.Close();

}

}

private void btnBinaryRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\emp.dat", FileMode.Open, FileAccess.Read);

br = new BinaryReader(fs);

txtId.Text = br.ReadInt32().ToString();

txtName.Text = br.ReadString();

txtSalary.Text = br.ReadDouble().ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

br.Close();

fs.Close();

}

}

StreamWriter sw;

StreamReader sr;

private void btnStreamWrite\_Click(object sender, EventArgs e)

{

try {

fs=new FileStream(@"D:\Tesla\testFile.txt",FileMode.Create,FileAccess.Write);

sw = new StreamWriter(fs);

sw.WriteLine(richTextBox1.Text);

MessageBox.Show("Done !");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

sw.Close();

fs.Close();

}

}

private void btnStreamRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\testFile.txt", FileMode.Open, FileAccess.Read);

sr = new StreamReader(fs);

richTextBox1.Text= sr.ReadToEnd();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

sr.Close();

fs.Close();

}

}

**Serialization in C#**

**Definition:**  it is a process if converting an object into a stream of bytes store the object or transmit it or store in memory, or database or file

.net app —-> serialization → java app

React java

Angular .net

4 types of serialization

1. Binary
2. XML
3. SOAP (simple object access protocol)
4. JSON (javascript object notation)
5. Binary Serialization:
6. Data will be store in binary format
7. Binary serialization maintain exact copy of your data
8. All the members can serialized by using
9. Limitation of Binary serialization is data can not be transmit through firewall

2. XML serialization:

1. XML → extensible markup language
2. It uses user defined tags to store data
3. Data will be stored in text format
4. Only public members can be serialized using XML serialization
5. We can transmit the XML file over the network
6. App needs to typecast the data by their own

<department>

<id>1</id>

<name> HR</name>

<location>pune</location>

<department>

3. SOAP → simple object access protocol

1. SOAP will maintain the exact copy of data
2. All members can be serialized using SOAP

<department>

<id>1</id>

<name> HR</name>

<location>pune</location>

<department>

4. JSON → javascript object notation

1. Accepted by the universe to transmit data over the net
2. It is a lightweight format
3. Maintain the exact copy of data
4. All members be serialized

dept={

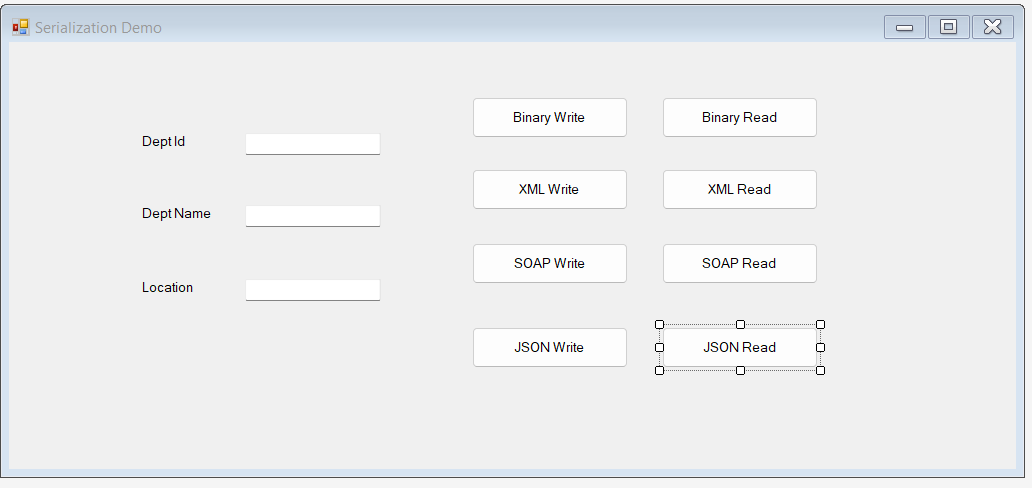
“deptid”:101,

”deptname”:”HR”,

”location”:”Pune”

}

**Deserialization :** get the data from file /DB /memory from stream of bytes to object



using System;

using System.Windows.Forms;

using System.Runtime.Serialization.Formatters.Binary;

using System.Runtime.Serialization.Formatters.Soap;

using System.Text.Json;

using System.IO;

using System.Xml.Serialization;

namespace FileIO

{

public partial class Form2 : Form

{

FileStream fs;

public Form2()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void btnBinaryWrite\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.dat", FileMode.Create, FileAccess.Write);

BinaryFormatter binaryFormatter = new BinaryFormatter();

Department dept = new Department();

dept.DeptId = Convert.ToInt32(txtId.Text);

dept.DeptName = txtName.Text;

dept.Location = txtLocation.Text;

binaryFormatter.Serialize(fs, dept);

MessageBox.Show("Done !");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnBinaryRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.dat", FileMode.Open, FileAccess.Read);

BinaryFormatter binaryFormatter = new BinaryFormatter();

Department dept = new Department();

dept=(Department)binaryFormatter.Deserialize(fs);

txtId.Text = dept.DeptId.ToString();

txtName.Text = dept.DeptName;

txtLocation.Text = dept.Location;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnXMLWrite\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.xml", FileMode.Create, FileAccess.Write);

XmlSerializer xmlSerializer = new XmlSerializer(typeof(Department));

Department dept = new Department();

dept.DeptId = Convert.ToInt32(txtId.Text);

dept.DeptName = txtName.Text;

dept.Location = txtLocation.Text;

xmlSerializer.Serialize(fs, dept);

MessageBox.Show("Done !");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnXMLRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.xml", FileMode.Open, FileAccess.Read);

XmlSerializer xmlSerializer = new XmlSerializer(typeof(Department));

Department dept = new Department();

dept = (Department)xmlSerializer.Deserialize(fs);

txtId.Text = dept.DeptId.ToString();

txtName.Text = dept.DeptName;

txtLocation.Text = dept.Location;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnSOAPWrite\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.soap", FileMode.Create, FileAccess.Write);

SoapFormatter soapFormatter = new SoapFormatter();

Department dept = new Department();

dept.DeptId = Convert.ToInt32(txtId.Text);

dept.DeptName = txtName.Text;

dept.Location = txtLocation.Text;

soapFormatter.Serialize(fs, dept);

MessageBox.Show("Done !");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnSOAPRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.soap", FileMode.Open, FileAccess.Read);

SoapFormatter soapFormatter = new SoapFormatter();

Department dept = new Department();

dept = (Department)soapFormatter.Deserialize(fs);

txtId.Text = dept.DeptId.ToString();

txtName.Text = dept.DeptName;

txtLocation.Text = dept.Location;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnJSONWrite\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.json", FileMode.Create, FileAccess.Write);

Department dept = new Department();

dept.DeptId = Convert.ToInt32(txtId.Text);

dept.DeptName = txtName.Text;

dept.Location = txtLocation.Text;

JsonSerializer.Serialize<Department>(fs, dept);

MessageBox.Show("Done !");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

private void btnJSONRead\_Click(object sender, EventArgs e)

{

try

{

fs = new FileStream(@"D:\Tesla\dept.json", FileMode.Open, FileAccess.Read);

Department dept = new Department();

dept =JsonSerializer.Deserialize<Department>(fs);

txtId.Text = dept.DeptId.ToString();

txtName.Text = dept.DeptName;

txtLocation.Text = dept.Location;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

fs.Close();

}

}

}

}

**Homework:**

Student details (rollno, name, percentage )

Use 4 serialization to write & read data from file

**ADO.NET**

**Definition:** it is a component of .net framework, which run on the top of CLR , it act as a mediary between application & database to store & manage data.

.net app MS SQL server, oracle, mysql

C# ADO.NET SQL

2 types of architecture

1. Connected
2. Disconnected
3. Connected → app opens a conn with DB & fires the query, & app closes the conn with DB

To fire queries DML, DQL, (insert,update,delete,select)

When we want data to be up to date

Data is changeable

ADO.NET providers

1. Sql server database → SqlClient
2. Access → Oledb
3. Oracle → Oracle
4. mySql → Odbc

Connected architecture providers

1. Conection
2. Command
3. DataReader
4. Connection → configure the conn with DB

Open a conn with DB

Closes a conn with DB

To set the connection string to connection to SQL server DB using Windows authentication

“Server=THINKQUOTIENT\SQLEXPRESS;Database=TQ;Integrated Security=True;”

To set the connection string to connection to SQL server DB using SQL server authentication

“Server=THINKQUOTIENT\SQLEXPRESS;Database=TQ;User Id= sa; Password=Admin@123 ;”

Server / Data Source

Database / Initial Catalog

User Id / UID

Password / PWD

1. Command → it is responsible for fire the queries in to the DB

Methods

1. ExecuteNonQuery() → insert ,update, delete

Return type – >int → no of row affected by the query

1- success

0- fail

1. ExecuteReader() → select \* from emp / select \* from emp where id=1

Return type → object of DataReader

1. ExecuteScalar() → select max(salary ) from emp

Select with aggregate function

Return type → object

Required type casting explicitly

C. DataReader → to read data from DB , will use DataReader object

1. HasRows → it check the existence of records → bool
2. Read() → it is used to read data from DB & display in the app

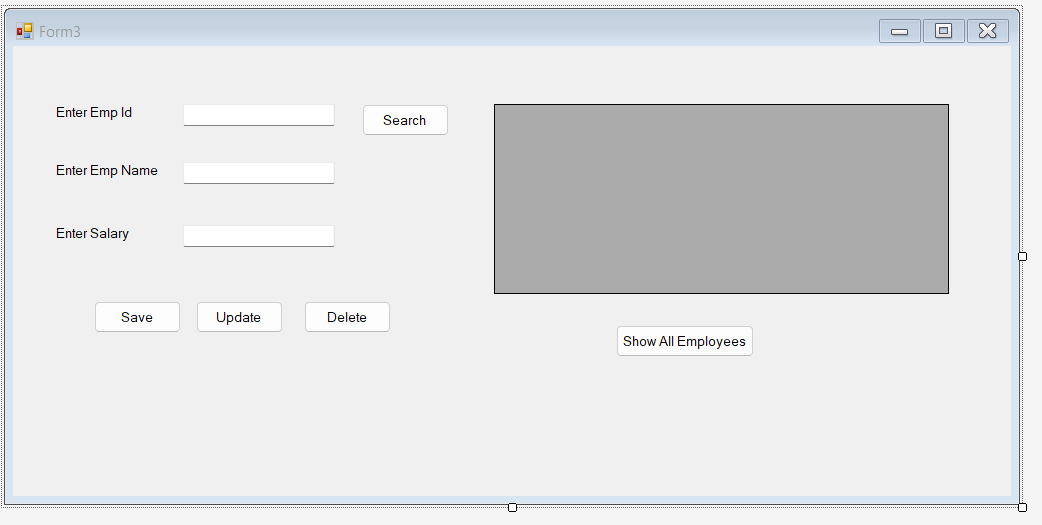
In app.config

<connectionStrings>

<add name="defaultConnection" connectionString="Server=THINKQUOTIENT\SQLEXPRESS;Database=TQ;Integrated Security=True;"

providerName="System.Data.SqlClient"/>

</connectionStrings>



using System;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Data;

namespace FileIO

{

public partial class Form3 : Form

{

SqlConnection con;

SqlCommand cmd;

SqlDataReader dr;

public Form3()

{

InitializeComponent();

string str = ConfigurationManager.ConnectionStrings["defaultConnection"].ConnectionString;

con = new SqlConnection(str);

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

string qry = "insert into emp values(@nm,@sal)";

cmd = new SqlCommand(qry, con);

cmd.Parameters.AddWithValue("@nm", txtName.Text);

cmd.Parameters.AddWithValue("@sal", Convert.ToDecimal(txtSalary.Text));

con.Open();

int result=cmd.ExecuteNonQuery();

if (result == 1)

{

MessageBox.Show("Record inserted");

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

try

{

string qry = "update emp set name=@nm , salary=@sal where id=@id";

cmd = new SqlCommand(qry, con);

cmd.Parameters.AddWithValue("@nm", txtName.Text);

cmd.Parameters.AddWithValue("@sal", Convert.ToDecimal(txtSalary.Text));

cmd.Parameters.AddWithValue("@id", Convert.ToInt32(txtId.Text));

con.Open();

int result = cmd.ExecuteNonQuery();

if (result == 1)

{

MessageBox.Show("Record updated");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

try

{

string qry = "delete from emp where id=@id";

cmd = new SqlCommand(qry, con);

cmd.Parameters.AddWithValue("@id", Convert.ToInt32(txtId.Text));

con.Open();

int result = cmd.ExecuteNonQuery();

if (result == 1)

{

MessageBox.Show("Record deleted");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

try

{

string qry = "select \* from emp where id=@id";

cmd = new SqlCommand(qry, con);

cmd.Parameters.AddWithValue("@id", Convert.ToInt32(txtId.Text));

con.Open();

dr = cmd.ExecuteReader();

if (dr.HasRows)

{

while (dr.Read())

{

txtName.Text = dr["name"].ToString();

txtSalary.Text = dr["salary"].ToString();

}

}

else

{

MessageBox.Show("Record not found");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

private void btnShowAll\_Click(object sender, EventArgs e)

{

try

{

string qry = "select \* from emp";

cmd = new SqlCommand(qry, con);

con.Open();

dr = cmd.ExecuteReader();

if (dr.HasRows)

{

DataTable table = new DataTable();

table.Load(dr);// take dr & convert to table format

dataGridView1.DataSource = table;

}

else

{

MessageBox.Show("Record not found");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

}

}

**Homework:**

Create student table → rollno (PK , identity) , name, city, percentage

Perform CRUD operation

C → create / insert / save

R → Read / select

U → update

D→ delete

**Disconnected Architecture:**

1. In disconnected architecture app opens a conn with DB implicitly fetch the data at application side & closes the conn with DB implicitly
2. Data will be hold at application side in the local object called DataSet
3. DML operations (insert /update/ delete)will be performed at DataSet
4. Application opens a conn with DB implicitly reflect the changes from DataSet to the DB & closes the conn implicitly

Database is not highly changeable in that case we can use disconnected architecture

Online exam →

1000 candidate

Connected

20 questions

1. Fetch → answer → DB → 2 req
2. fetc-> ans submit 2req

40 req \* 1000 → 40000 → 1hr

Disconnected

1000 → start → 1000 req

Submit → DataSet

End exam -> DataSet → DB

1000 -> 1000 req

2000 total req → 1hr

Providers of disconnected architecture

1. Connection
2. DataAdapter → it is bridge between app & DB to fetch & update the data from DB
3. Fill() → to fetch data from DB & store in DataSet
4. Update() → reflect the changes from DataSet to DB

3. CommandBuilder → track the DataSet & generate the SQL queries & pass to the DataAdapter

DataSet:

It is collection of DataTable class

DataRow class

DataColumn class

DataSet

DataTable[0] as emp DataTable[1] DataTable[2]

DataRow DataRow

DataColumn DataColumn

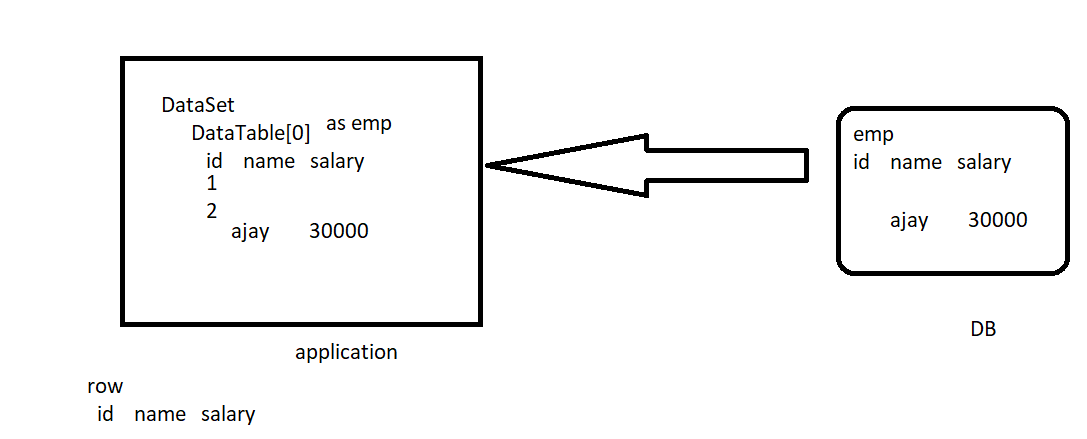
DataSet → DataTable[0] as emp

Id name salary

1 —-------------

2—-----------------

DataRow → id name salary



using System;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Data;

namespace FileIO

{

public partial class Form4 : Form

{

SqlConnection con;

SqlDataAdapter da;

SqlCommandBuilder scb;

DataSet ds; // it is not provider specific

public Form4()

{

InitializeComponent();

string str = ConfigurationManager.ConnectionStrings["defaultConnection"].ConnectionString;

con = new SqlConnection(str);

}

public DataSet GetAllEmps()

{

da = new SqlDataAdapter("select \* from emp", con);

da.MissingSchemaAction = MissingSchemaAction.AddWithKey;

scb = new SqlCommandBuilder(da);

ds = new DataSet();

da.Fill(ds,"emp");// give alias to the DataTable in DataSet

return ds;

}

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

ds = GetAllEmps();

DataRow row = ds.Tables["emp"].NewRow();

row["name"] = txtName.Text;

row["salary"] = txtSalary.Text;

ds.Tables["emp"].Rows.Add(row);

int result = da.Update(ds.Tables["emp"]);// reflect the change to main DB

if (result == 1)

{

MessageBox.Show("Record inserted");

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

try

{

ds = GetAllEmps();

DataRow row = ds.Tables["emp"].Rows.Find(txtId.Text);

if (row != null)

{

row["name"] = txtName.Text;

row["salary"] = txtSalary.Text;

int result = da.Update(ds.Tables["emp"]);// reflect the change to main DB

if (result == 1)

{

MessageBox.Show("Record updated");

}

}

else

{

MessageBox.Show("Id not found to update");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

try

{

ds = GetAllEmps();

DataRow row = ds.Tables["emp"].Rows.Find(txtId.Text);

if (row != null)

{

row.Delete();

int result = da.Update(ds.Tables["emp"]);// reflect the change to main DB

if (result == 1)

{

MessageBox.Show("Record deleted");

}

}

else

{

MessageBox.Show("Id not found to delete");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

try

{

ds = GetAllEmps();

DataRow row = ds.Tables["emp"].Rows.Find(txtId.Text);

if (row != null)

{

txtName.Text = row["name"].ToString();

txtSalary.Text = row["salary"].ToString();

}

else

{

MessageBox.Show("Record not found");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnShowAll\_Click(object sender, EventArgs e)

{

try

{

ds = GetAllEmps();

dataGridView1.DataSource = ds.Tables["emp"];;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

**Responsive Web Development:**

**HTML**

1. Hypertext markup language
2. Hypertext → text with some style
3. Markup → tag (pre defined)
4. Using HTML we can design the static pages
5. HTML tags will be render by the browser

HTML uses pre defined tags

2 types

1. Pair tag <> </>
2. Single tag <>

< />

E.g. paragraph

<p>................................. </p>

<h1>------------------</h1>

<br>

<b>

<i>

<u>

<strong>

<small>

<div>

<marquee direction="right" scrollamount="20" behavior="scroll">

<h1> GOOD MORNING </h1>

</marquee>

<table border="1" cellpadding="10" cellspacing="8">

<caption>Student List</caption>

<thead>

<tr>

<th colspan="2">Roll No / Name</th>

<th>City</th>

<th>Percentage</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td>Ajay</td>

<td rowspan="3">Pune</td>

<td>88.77</td>

</tr>

<tr>

<td>2</td>

<td>Ajay</td>

<td>88.77</td>

</tr>

<tr>

<td>4</td>

<td>Ajay</td>

<td>88.77</td>

</tr>

<tr>

<td>3</td>

<td>Ajay</td>

<td colspan="2">Pune 88.77</td>

</tr>

<tr>

<td>5</td>

<td>Ajay</td>

<td>Pune</td>

<td>88.77</td>

</tr>

</tbody>

</table>

<table border="1">

<tr>

<td colspan="2">1</td>

<td rowspan="2">pune</td>

</tr>

<tr>

<td>2</td>

<td>Test</td>

</tr>

</table>

<a href="Login.html" target="\_blank">

Login

</a>

<a href="https://www.google.com" title="Google Page">

<img src="laptop1.avif" height="20" width="20"/>

</a>

In html <form> </form> will be used to pass data on the server

<input type=”text” />

Radio

Checkbox

Email

Password

Date

Datetime-local

Time

Month

Week

Color

Url

Range

Number

Button

Submit

Reset

**Safari browser**

Datetime

Search

tel

<form action="Login.html">

Enter Name :<input type="text" placeholder="full name" required /> <br />

Country :<input type="text" value="India" readonly /> <br />

Select Date :<input type="date" /> <br />

Select Gender :<input type="radio" name="gender" /> Male

<input type="radio" name="gender" /> Female

<br />

Select Exp : <input type="radio" name="exp" /> 2 years

<input type="radio" name="exp" /> 4 years

<input type="radio" name="exp" /> 6 years

<br />

Select Hobbies : <input type="checkbox" name="hobbies" /> Music

<input type="checkbox" name="hobbies" /> Sports

<input type="checkbox" name="hobbies" />Study

<br />

Select File :<input type="file" multiple />

<br />

Enter Age :<input type="number" min="18" max="60" />

<br />

Enter Email :<input type="email" maxlength="20" />

<br />

Select City :

<select multiple>

<option>--select city--</option>

<option>Pune</option>

<option>Mumbai</option>

<option>Nagpur</option>

</select>

<br />

<input type="button" value="Button" />

<input type="submit" value="Submit" />

<input type="reset" value="Reset" />

</form>

</body>

HTML 5 Features

1. Added new tags -> header,footer,article, mark, nav, section (semantic tags)
2. Added new attributes → placeholder,required, readonly, multiple, min , max, maxlength
3. Added new input types → email, range,

<header></header>

**Homework:**

1. Create register page & login page using html form

Register page → username, email id, phone number new password, confirm password

Login page → email id & password

**CSS → cascading style sheet**

It is used to set the look & feel of html page

**Cascading →** integrate / nest

**Style sheet** → sheet related to look & feel of html page

1. Font / Text → typography
2. Page layout
3. Colors / background , text color
4. Borders

3 types of CSS

1. External css → css which is used to set the look & feel for multiple html pages (menu, title, font size, footer, banner, images resolution, buttons….)

<link href=”name of file” rel=”stylesheet”/>

1. Internal css → set the look & feel for a single page

<style> </style>

1. Inline css → set the look & feel for a single html tag

<div style="background-color: #138D75;height:200px;width:100%">

thi is example of inline css

</div>

<https://htmlcolorcodes.com/>

Selectors in CSS → we can use selectors in external & internal css

1. Element / tag
2. Id → denotes by #
3. Class → common css for diff / multiple html elements, denote using . (dot)

**Basic syntax:**

Selector

{

Css property: value;

property:value

}

**Box model in CSS**

1. Margin
2. Border
3. Padding

T- top , R- right, B- bottom , L - left

Margin : T R B L (all values would be different ) margin: 10px 20px 30px 40px ;

Margin: TB LR (combination of 2 → TB & LR ) margin: 10px 20px ;

Margin: TRBL (same for 4 sides) → margin: 30px;

Margin-top:

Margin-bottom

Margin-left:

Margin-right

(above rules are same for padding)

box -shadow

text-shadow

border-radius: 104px 186px 52px 34px;

TL TR BR BL

**Position property in CSS**

1. Static → it is default position of any html element
2. Relative → follow the immediate parent , and set the position
3. Absolute → follow the root element, i.e. body tag
4. Fixed → position to be fixed, after scroll to the page
5. Sticky → stick the element when its margin value becomes true

.parent{

background-color:red;

width:35%;

height:250px;

position:relative;

}

.child {

background-color: yellow;

height: 57px;

width: 80%;

position: fixed;

bottom:0;

}

<body>

<h1>Think Quotient</h1>

<div class="parent">

3 types of CSS

External css → css which is used to set the look & feel for multiple html pages (menu, title, font size, footer, banner, images resolution, buttons….)

Internal css → set the look & feel for a single page

Inline css → set the look & feel for a single html tag

</div>

<div class="child">

This is child div

</div>

<div class="parent">

3 types of CSS

External css → css which is used to set the look & feel for multiple html pages (menu, title, font size, footer, banner, images resolution, buttons….)

Internal css → set the look & feel for a single page

Inline css → set the look & feel for a single html tag

</div>

<div style="height:500px"></div>

</body>

**Combinators in CSS**

1. Space → descendent
2. > → child
3. + → adjacent sibling
4. ~ → general sibling

<body>

<p>

this is para 1

</p>

<div>

<p>

this is para 2

</p>

<p>

this is para 3

</p>

<section>

<p>

this is para 4

</p>

</section>

</div>

<section>

this is section 1

</section>

<p>

this is para 5

</p>

<p>

this is para 6

</p>

<section>

this is section 2

</section>

</body>

Media query in CSS

To write css for mobile & tab view

h1{

font-size:70px;

}

.my-img {

position: absolute;

top: 10px;

left: 495px;

height: 200px;

}

/\*-------For tab-------\*/

@media only screen and (min-device-width:758px) and (max-device-width:1000px)

{

h1 {

font-size: 30px;

}

.my-img {

left: 224px;

}

}

/\*-------For mobile-------\*/

@media only screen and (min-device-width:320px) and (max-device-width:480px)

{

h1 {

font-size:14px;

}

.my-img{

left:165px;

}

}

<body>

<h1>Think Quotient</h1>

<img src="laptop1.avif" class="my-img"/>

</body>

**Bootstrap**

1. It is front end framework to design the web page
2. It is combination of html + css +javascript
3. Mobile first approach
4. Provides built in component to design forms, menu (navigation), font / typography, dropdowns & so on
5. How to use bootstrap framework
6. Grid system in Bootstrap

<div class="container">

<div class="row">

<div class="col-md-4 col-xs-12 col-sm-6"></div>

<div class="col-md-4 col-xs-12 col-sm-6"></div>

<div class="col-md-4 col-xs-12 col-sm-6"></div>

<div class="col-md-4 col-xs-12 col-sm-6"></div>

</div>

</div>

Sample.html

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<title></title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">

<style>

div{

margin-bottom:20px;

}

.even{

background-color:yellow;

}

.odd{

background-color:aqua;

}

</style>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-1 odd">1</div>

<div class="col-1 even">2</div>

<div class="col-1 odd">3</div>

<div class="col-1 even">4</div>

<div class="col-1 odd">5</div>

<div class="col-1 even">Think Quotient</div>

<div class="col-1 odd">7</div>

<div class="col-1 even">8</div>

<div class="col-1 odd">9</div>

<div class="col-1 even">

Navbar themes are easier than ever thanks to Bootstrap’s combination of Sass and CSS

</div>

<div class="col-1 odd">11</div>

<div class="col-1 even">12</div>

</div>

</div>

<div class="container-fluid">

<div class="row">

<div class="col-3 odd">

Navbar themes are easier than ever thanks to Bootstrap’s combination of Sass and CSS variables. The default is our “light navbar” for use with light background colors, but you can also apply .navbar-dark for dark background colors. Then, customize with .bg-\* utilities.

</div>

<div class="col-3 even">2</div>

<div class="col-3 odd">

<div class="card" style="width: 18rem;">

<img src="laptop1.avif" class="card-img-top" alt="...">

<div class="card-body">

<h5 class="card-title">Card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

<a href="#" class="btn btn-primary">Go somewhere</a>

</div>

</div>

</div>

<div class="col-3 even">4</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-4 odd">1</div>

<div class="col-8 even">2</div>

</div>

<div class="row">

<div class="col-10 odd">1</div>

<div class="col-2 even">2</div>

</div>

<div class="row">

<div class="col-4 odd">1</div>

<div class="col-4 even">2</div>

<div class="col-4 odd">

Navbar themes are easier than ever thanks to Bootstrap’s combination of Sass and CSS variables. The default is our “light navbar” for use with light background colors, but you can also apply .navbar-dark for dark background colors. Then, customize with .bg-\* utilities.Navbar themes are easier than ever thanks to Bootstrap’s combination of Sass and CSS variables. The default is our “light navbar” for use with light background colors, but you can also apply .navbar-dark for dark background colors. Then, customize with .bg-\* utilities.

</div>

</div>

</div>

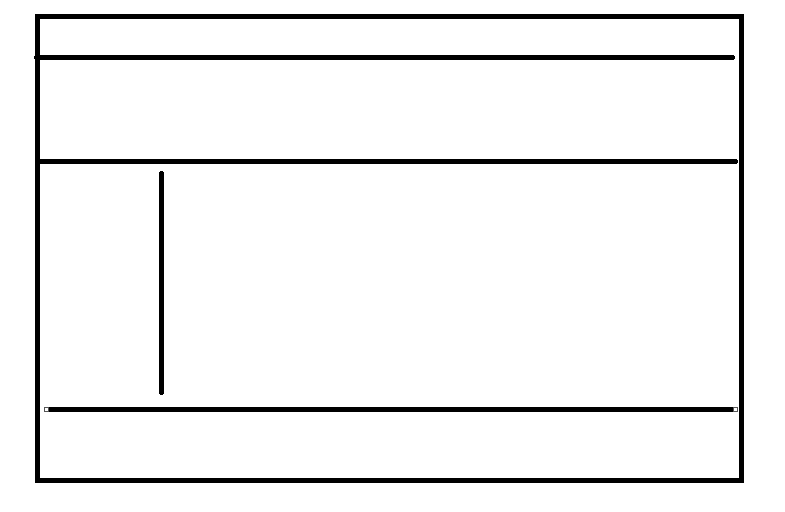
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js" integrity="sha384-oBqDVmMz9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlmh/jp3" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js" integrity="sha384-IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>

</body>

</html>



assignment : design above layout

<link rel="stylesheet"

href="<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css>">

https://fontawesome.com/v4/icons/

**JavaScript:**

1. It is an object oriented programming language.
2. It is most powerful programming to develop web applications or web sites
3. Use Javascript to make HTML interactive / dynamic
4. Manage data, do validations, change the content or form data dynamically, many more..
5. Use at client side & also at server side

Client side scripting

Client side → **browser** → interpreter

HTML + Javascript

React + Javascript

Angular + javascript

Server side scripting

Code will be execute on the server

Browser —-> [www.xyz.com/login](http://www.xyz.com/login) —----------> Server

1. Server OS, (windows NT, 2000, linux)

2. Provides RAM + HDD space

3. Security, backup

4. Database (MS-SQL , mysql….mongoDB)

E.g. providers → AWS , Azure, go daddy

Node.js + Javascript

ASP +Javascript

Php +Javascript

HTML + Javascript

Two ways to use Javascript

1. Using <script> </script>
2. Crate external .js file & integrate .js file in html

E.g.

Test.js

<script src=”Test.js”> </script>

**Variable: →**  rules for variable declaration in C# & javascript are same

Name

Age

Percentage

First name -> not allowed

n@me -> NA

Data Types:

Number → to store the whole number + precision number , 10, 30,1,.... 56.55, 88.99….

String → collection of characters , used with ‘ ‘ or “ “

Boolean → true / false

Array → collection of similar data elements

Object → any user defined object , emp , prod, stud…..

Any → when we are not sure about the type of data

<script>

var age = 18;

var name = "Think Quotient";

var percentage = 88.99;

var nm=1;

var isTrue = true;

console.log("my age is " + age);

console.log("Company name is " + name);

document.write("Company name is " + name);

</script>

Declaration of variables in javascript

Var

Let

Const

Conditional statements : if..else.. Switch case

Loop: for, while & do… while

Syntax of these are same as C#

<script>

var age = 10;

if (age >= 18) {

console.log("Valid user");

}

else {

console.log("Not valid user");

}

for (var i = 1; i <= 10; i++) {

if(i%2==0)

console.log(i);

}

var a = 10;

do {

console.log(a);

a++;

} while (a <= 20);

</script>

**Function in Javascript**

<script>

// 1. function with no parameters & not return type

function Add() {

var a = 10, b = 20;

var c = a + b;

console.log(c);

}

// 2. function with parameters & not return type

function AcceptDetails(rollno, name, per) {

console.log(rollno + " " + name + " " + per);

}

//3. with parameters & return type

function multiply(a, b) {

var res = a \* b;

return res;

}

// call the function

Add();

AcceptDetails(1, "Test", 88.66);

var m = multiply(56, 78);

console.log(m);

</script>

Assignment:

Create a function for each, to calculate add, sub,multiply, div , mod & square

Use parameter + return type

<script>

// **modal dialog box in javascript**

// alert , confirm

var res = confirm("do you want to process for payment ?");

if (res == true) {

// code goes here...

alert("Success !");

}

else {

alert("You cancelled the payment");

}

</script>

**Scope in Javascript:**

<script>

// scope in Javascript

// 1. Global 2. function 3. Local scope

var a = 10; // global

let b = 100; // global scope

const pi = 3.14;

function Test() {

var a = 20; // function scope

let b = 200; // function scope

//let a = 500; // this not allowed in Javascript, as we have already using a with var

//console.log(a);

pi = 3.15; // not alllowed

console.log(pi);

//console.log(a);

//console.log(b);

}

{

var a = 30; // global scope

let b = 300; // local scope

let d = 400;

console.log(d);

}

var pi = 22; // not allowed

console.log(pi);

//console.log(d);

Test();

console.log(a);

console.log(b);

</script>

<script>

**"use strict"**

var a = 10;

console.log(a);

</script>

Array

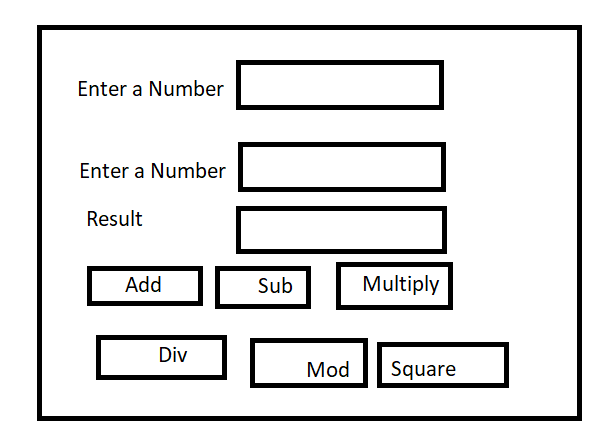
Object

Object + function

Integration with HTML

Homework:

Create a HTML page to perform add, sub,mul,div, mod, sq



**DOM (**Document object Model)

1. It is a programming interface for html & xml document
2. In DOM html elements will be presented as hierarchical order
3. It is logical representation of html elements
4. To access html element /tag, attributes or css in javascript we need DOM

Root → node html

Parents node head body

Child nodes title h1, p, div,table…..

meta

DOM property & method

1. Property → innerHTML
2. Method → getElementById()

With the help of DOM javascript can add / remove or modify the html

Add / remove / modify the css

**Event in javascript**

Onclick

Onmousemove

Onenter

Onleave

Onhover

**Array in Javascript:**

Collection of similar data elements

Array is zero index based

<script>

var names = ["Atul", "Rajesh", "Kishor", "Ajay", "Amol"];

console.log(names[1]);

// names[2] = "Test";

console.log(names);

// print or display using html

var data = "<ol type='i'>";

for (var i = 0; i < names.length; i++) {

data += "<li>" + names[i] + "</li>";

}

data += "</ol>";

console.log(data);

document.getElementById("div1").innerHTML = data;

</script>

<script>

var names = ["Atul", "Rajesh", "Kishor", "Ajay", "Amol"];

// add at the last

names.push("Rohan");

// add at the first

names.unshift("Suraj");

names.pop(); // remove the last element

names.shift(); // remove the first element

names.splice(1, 0, "Tushar", "Anil"); // to add in between + can replace the elemet using delete count

names.sort();

names.reverse();

var newNames = names.slice(1, 6);

console.log(newNames);

var data = "<ol type='i'>";

for (var i = 0; i < names.length; i++) {

data += "<li>" + names[i] + "</li>";

}

data += "</ol>";

console.log(data);

document.getElementById("div1").innerHTML = data;

</script>

Lets create an array of 5 elements in javascript

& use array methods

**Iterate using forEach in Javascript**

<script>

var names = ["Atul", "Rajesh", "Kishor", "Ajay", "Amol"];

var data = "<ol>";

names.forEach(myFunction);

data += "</ol>";

function myFunction(value) {

data += "<li>" + value + "</li>";

}

var nos = [78, 45, 23, 67, 89];

var data2 = "<ol>";

nos.forEach(myfunction2);

data2 += "</ol>";

function myfunction2(value) {

if (value < 50) {

data2 += "<li>" + value + "</li>";

}

}

document.getElementById("div1").innerHTML = data;

document.getElementById("div2").innerHTML = data2;

</script>

<script>

**// working with string in Javascript**

var text = " this,is,sample,text ";

console.log(text.charAt(6));

var text2 = "Hello";

var result = text.concat(" ",text2," ","Good Morning");

console.log(result);

console.log(result.length);

// var txt = text.slice(4);

var txt = text.slice(4, 11);

var data = text.split(",");

//var data = text.split(" ", 3);

console.log(data);

console.log(txt);

console.log(text.toUpperCase());

console.log(text.toLowerCase());

// trim

text.trim();

</script>

<script>

**// working with number in Javascript**

var num = 23.768;

var text = num.toString(); // convert number to string

var no = num.toFixed(2); // currency

console.log(no);

</script>

<script>

**// working with object in javascript**

var employee = {

id: 1,

name: "Amol",

salary: 56000,

print: function () {

return "Employee Details " + this.id + " " + this.name + " " + this.salary;

},

calculateSalary: function () {

var bonus = this.salary \* 0.10;

this.salary += bonus;

return this.salary;

}

};

console.log(employee.print());

console.log(employee.calculateSalary());

</script>

**Homework:**

1. Create an object of student, add rollno, name & 3 subjects
2. Create function to print total marks & percentage;
3. Create an object of Product add id, name , price
4. Create a function to display product
5. Create a function & print discounted price of a product → 10 % discount
6. Create a product list & display using array object

**Array Object**

<script>

// working with object in javascript

var employees =

[

{

id: 1,

name: "Amol", //employees[0]

salary: 56000

},

{

id: 2,

name: "Amol", //employees[1]

salary: 56000

},

{

id: 3,

name: "Amol",

salary: 56000

},

{

id: 4,

name: "Amol",

salary: 56000

}

]

console.log(employees);

var data = "<table border='1'><tr><th>Id</th><th>Name</th><th>Salary</th></tr>";

for (var i = 0; i < employees.length; i++) {

data += "<tr><td>" + employees[i].id + "</td><td>" + employees[i].name + "</td><td>" +

employees[i].salary + "</td></tr>";

}

data += "</table>";

document.getElementById("div1").innerHTML = data;

</script>

**Arrow function in Javascript:**

Arrow function were introduced in ES6 of Javascript

ES → ECMAscript (European computer manufacturer association)

Arrow function allows us to write shorter code / syntax

<script>

// javascript function

function Add(a, b) {

return a + b;

}

// arrow function ()=>{}

let addition = (a, b) => a + b;

var str = (name) => { return "Hello " + name; }

function test() {

return "Hello to all";

}

var test = (a,b) => {

var m = a \* b;

return m;

}

var str1 = () => "Hello to all";

var emp = {

id: 1,

name: "user1",

salary: 34000,

print: function () {

return "Emp details " + this.id + " " + this.name + " " + this.salary;

},

print1: () => {

return "Emp details " + emp.id + " " + emp.name + " " + emp.salary;

}

}

console.log(emp.print1());

console.log(test(20,40));

</script>

JSON.stringify()

object.values()

var emp = {

id: 1,

name: "user1",

salary: 34000

}

console.log(JSON.stringify(emp));

console.log(Object.values(emp));

**entries() in javascript →** to get the key-value pair data

<script>

var fruits = { apple: 200, orange: 250, banana: 100 };

var text="";

for (let [key, value] of Object.entries(fruits)) {

text += key + " " + value+ "<br/>";

}

document.getElementById("div1").innerHTML = text;

</script>

**Call() , apply() & bind() in Javascript:**

<script>

var person = {

print: function (city, country) {

return this.firstname + " " + this.lastname + " " + city + " " + country;

}

}

var emp={

firstname:"fname",

lastname:"lname"

}

var customer = {

firstname: "cfname",

lastname:"clname"

}

//console.log(person.print.call(emp));

//console.log(person.print.call(customer));

console.log(person.print.call(emp, "Pune", "India"));

console.log(person.print.apply(emp, ["Pune", "India"]));

</script>

Call is used to call another object

If we want to pass separate parameters we can use call()

If we want to pass an array to it we need to use apply()

<script>

var person = {

firstname: "pfname",

lastname:"plname",

print: function () {

return this.firstname + " " + this.lastname;

}

}

var emp={

firstname:"fname",

lastname:"lname"

}

var customer = {

firstname: "cfname",

lastname:"clname"

}

var result = person.print.bind(customer); // borrow object one to another

console.log(result());

</script>

**Javascript Callback**

A callback is function that pass as argument to another function

When function wait for a result & run the finish the execution of the first function

<body>

<div id="div1">

</div>

<script>

function add(a, b, mycallbackfunc) {

var sum = a + b;

mycallbackfunc(sum);

}

add(10, 20, display);

function display(result) {

document.getElementById("div1").innerHTML = result;

}

</script>

</body>

<div id="div1">

</div>

<script>

function display() {

document.getElementById("div1").innerHTML = "Hello to all";

}

//display();

// setTimeout(display, 5000);

function showTime() {

let d = new Date();

let res = d.getHours() + " :" + d.getMinutes() + " :" + d.getSeconds();

document.getElementById("div1").innerHTML = res;

}

//showTime();

setInterval(showTime, 10000);

</script>

</body>

Synchronous → sequence

Execution of function one after another

Asynchronous → concurrent

Execution of multiple functions at same time

Asyc code allows the program to be executed immediately where the synchronous code block the further execution

1. Promise
2. Asyc

Promise → promises are used to handle the asynchronous operations in javascript

We can handle multiple asyc calls

Advantages

1. Improve the code readability
2. Handle the multiple asyc call
3. Better error handling

Three states of promises

1. Resolve → success
2. Reject → error / exception/ error in the code
3. Pending → in process / in request

Var promise = new Promise(function(resolve,reject){

resolve();

reject();

});

<script>

var promise = new Promise(function (resolve, reject) {

var a = 10;

if (a == 100) {

resolve("Success");

}

else {

reject("error");

}

});

promise.then(function (value) {

console.log(value);

}, function (val) {

console.log(val);

}

).catch(function (value) {

console.log(value);

});

</script>

1. Promise can be consumed using .then() & exception will be handled by .catch()
2. .then() takes 2 arguments first is resolve & second is reject
3. If promise is resolved it is success
4. If promise returns the reject that means it is failed

<https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/iis/www-administration-management/http-status-code>

200

201

204

400

404

301

500

503

403

<script>

let promise = new Promise(function (resolve, rejct) {

let req = new XMLHttpRequest();

req.open("GET", "https://dummyjson.com/products/1");

req.onload = function () {

if (req.status == 200) {

resolve(req.response);

}

else {

rejct("file not found");

}

};

req.send();

});

promise.then(function (data) {

console.log(data);

}, function (error) {

console.log(error);

});

</script>

<div id="div1">

</div>

<script>

let promise = new Promise(function (resolve, reject) {

let req = new XMLHttpRequest();

// req.open("GET", "https://dummyjson.com/products");

req.open("GET", "https://jsonplaceholder.typicode.com/todos/");

req.onload = function () {

if (req.status == 200) {

resolve(req.response);

}

else {

reject("file not found");

}

};

req.send();

});

promise.then(function (data) {

display(data);

}, function (error) {

console.log(error);

});

var text = "<table border='1'><tr><td>UserId</td><td>Id</td><td>title</td><td>Completed</td></tr>";

function display(data) {

var jsonData = JSON.parse(data);

console.log(jsonData)

for (var i = 0; i < jsonData.length; i++) {

text += "<tr><td>" + jsonData[i].userId + "</td><td>" + jsonData[i].id + "</td><td>" + jsonData[i].title + "</td><td>" +jsonData[i].completed + "</td></tr>";

}

text += "</table>";

// console.log(text)

document.getElementById("div1").innerHTML = text;

}

</script>

</body>

Async

Await

1. Async & await makes promise easier to write
2. Async → to make function which returns the promise
3. Await → makes a function to wait for promise response

<script>

async function myFunction() {

return "Hello from function";

}

myFunction().then(function (value) {

console.log(value);

})

</script>

<body>

<div id="div1">

</div>

<script>

async function getData() {

let promise = new Promise(function (resolve, reject) {

let req = new XMLHttpRequest();

// req.open("GET", "https://dummyjson.com/products");

req.open("GET", "https://jsonplaceholder.typicode.com/todos/");

req.onload = function () {

if (req.status == 200) {

resolve(req.response);

}

else {

reject("file not found");

}

};

req.send();

});

let info = await promise;

display(info);

}

getData();

var text = "<table border='1'><tr><td>UserId</td><td>Id</td><td>title</td><td>Completed</td></tr>";

function display(data) {

var jsonData = JSON.parse(data);

console.log(jsonData)

for (var i = 0; i < jsonData.length; i++) {

text += "<tr><td>" + jsonData[i].userId + "</td><td>" + jsonData[i].id + "</td><td>" + jsonData[i].title + "</td><td>" + jsonData[i].completed + "</td></tr>";

}

text += "</table>";

// console.log(text)

document.getElementById("div1").innerHTML = text;

}

</script>

</body>

**With User input & button event**

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

</head>

<body>

Enter Id :<input type="text" id="t1" />

<input type="button" value="Get" onclick="getDataById()" />

<input type="button" value="Get All" onclick="getData()" />

<div id="div1">

</div>

<script>

async function getDataById() {

var id = document.getElementById("t1").value;

let promise = new Promise(function (resolve, reject) {

let req = new XMLHttpRequest();

// req.open("GET", "https://dummyjson.com/products");

var reqUrl = "https://jsonplaceholder.typicode.com/todos/" + id;

req.open("GET", reqUrl);

req.onload = function () {

if (req.status == 200) {

resolve(req.response);

}

else {

reject("file not found");

}

};

req.send();

});

let info = await promise;

display(info);

}

async function getData() {

let promise = new Promise(function (resolve, reject) {

let req = new XMLHttpRequest();

// req.open("GET", "https://dummyjson.com/products");

var reqUrl = "https://jsonplaceholder.typicode.com/todos/";

req.open("GET", reqUrl);

req.onload = function () {

if (req.status == 200) {

resolve(req.response);

}

else {

reject("file not found");

}

};

req.send();

});

let info = await promise;

display2(info);

}

function display(data) {

var text = "";

text = "<table border='1'><tr><td>UserId</td><td>Id</td><td>title</td><td>Completed</td></tr>";

var jsonData = JSON.parse(data);

text += "<tr><td>" + jsonData.userId + "</td><td>" + jsonData.id + "</td><td>" + jsonData.title + "</td><td>" + jsonData.completed + "</td></tr>";

text += "</table>";

document.getElementById("div1").innerHTML = text;

}

function display2(data) {

var text = "";

text = "<table border='1'><tr><td>UserId</td><td>Id</td><td>title</td><td>Completed</td></tr>";

var jsonData = JSON.parse(data);

for (var i = 0; i < jsonData.length; i++) {

text += "<tr><td>" + jsonData[i].userId + "</td><td>" + jsonData[i].id + "</td><td>" + jsonData[i].title + "</td><td>" + jsonData[i].completed + "</td></tr>";

}

text += "</table>";

document.getElementById("div1").innerHTML = text;

}

</script>

</body>

</html>

**Storage in javascript:**

Http / https protocol

Keep some user data/ we need to manage user data by our own

Http is a state less protocol

State → page information / information which is on the page

Cookies →

Limitations

1. Cookies are included in the req & response
2. Cost the performance of application
3. Security
4. File size / memory

Storage → any page information or user information that you need to store or manage can be done with help of storage

2 types

1. Local storage → permanent storage
2. Session storage → particular time / a duration of login to logout is called a session

setItem(“key”,”value”); – >set the value in the localstorage / session storage

getItem(“Key”) → get the value from local storage / session storage

removeItem(“Key”) → remove the value from local / session storage

Clear () → remove all the values from the storage

<script>

function register() {

var email = document.getElementById("exampleInputEmail").value;

var pass = document.getElementById("exampleInputPassword").value;

if (localStorage) {

localStorage.setItem("userEmail", email);

localStorage.setItem("userPass", pass);

return true;

}

else {

return false;

}

}

</script>

<script>

function authenticate() {

var email = localStorage.getItem("userEmail");

var pass = localStorage.getItem("userPass");

var textEmail = document.getElementById("exampleInputEmail").value;

var textPass = document.getElementById("exampleInputPassword").value;

if (email == textEmail && pass == textPass) {

alert("Success");

}

else {

alert("Login Fail");

}

}

</script>

<https://www.geeksforgeeks.org/differences-between-net-core-and-net-framework/>

**ASP.NET MVC**

(LTS)long term support from the Microsoft

3.1(LTS) & 5(out of support) → VS 2019

Asp.net core 6 using mvc (LTS) → VS 2022

Model view controller → it is design pattern for developing web based application

MVP → model view presenter

Model → is used to manage the application data (classes, collections, array,ado.net,entity framework, LINQ)

View → UI logic / pages / web page

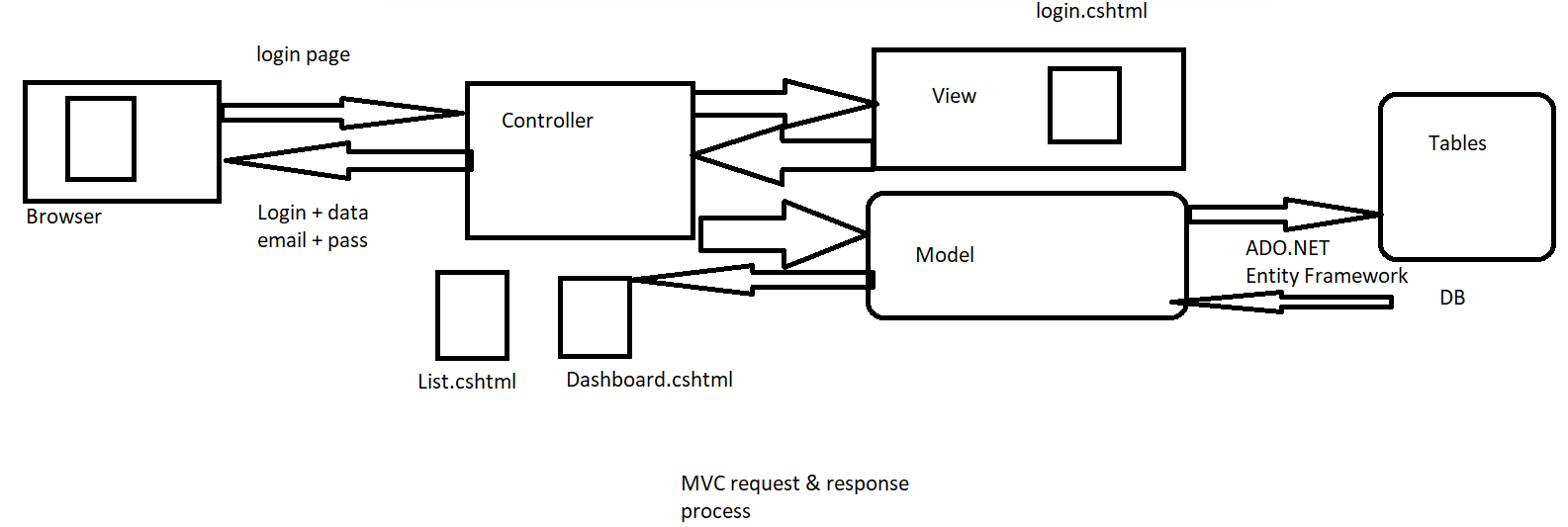
Controller → it is used manage the application request & response

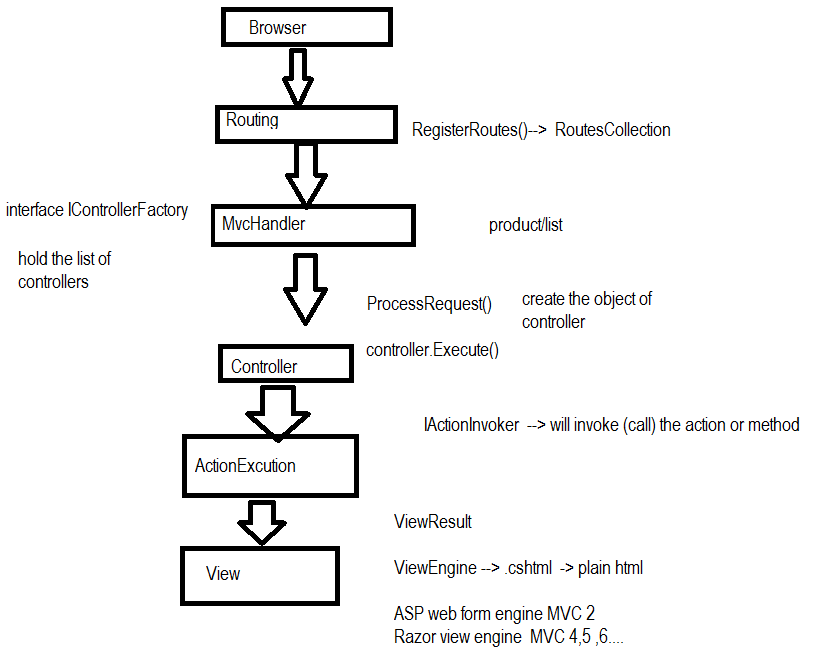
Select the appropriate view to display

Select the appropriate model

Server side programming

Req → made with HTTP protocol





<http://localhost:11249/Home/Privacy>

<http://localhost:11249/Home/AboutUs>

<http://localhost:11249/Home/ContactUs>

<http://localhost:11249/Employee/List>

In mvc user requests are mapped with controller & actions

In mvc action / method returns the IActionResult interface

IActionResult → it represent the response of the view

1. HTML -> View() -> ActionResult
2. JSON → Json() -> JsonResult
3. File → File() -> FileContentResult

localhost:54000/Home/Index

localhost:54000/User/Login

localhost:54000/Home/AboutUs

localhost:54000/Home/ContactUs

**Working with ViewData**

1. When we want to pass data from controller to view we use ViewData
2. ViewData uses Dictionary to hold the information
3. Key → string
4. Value →Object // implicit type casting
5. When we read value from viewdata on cshtml page we need explicit type casting
6. Viewdata needs null value checking
7. Viewdata added in mvc 2

To read data on the View

We need Razor syntax

Razor syntax start with @ sign

Purpose of razor syntax is use C# code on the .cshtml page

public IActionResult Index()

{

ViewData["message"] = "This is sample message from controller";

List<string> colors = new List<string>()

{

"Red","Green","Blue","Yellow","Balck","White","Pink"

};

ViewData["colors"] = colors;

List<Employee> employees = new List<Employee>() {

new Employee{Id=1,Name="emp1",Salary=25000},

new Employee{Id=2,Name="emp2",Salary=26000},

new Employee{Id=3,Name="emp3",Salary=27000},

new Employee{Id=4,Name="emp4",Salary=28000},

};

ViewData["emplist"] = employees;

return View();

}

<p>

@ViewData["message"]

</p>

@if(ViewData["colors"]!=null)

{

<ol>

@foreach (string item in (IEnumerable<string>)ViewData["colors"])

{

<li>

@item

</li>

}

</ol>

}

<table class="table table-bordered">

<tr>

<th>Id</th>

<th>Name</th>

<th>Salary</th>

</tr>

@foreach (Employee emp in (IEnumerable<Employee>)ViewData["emplist"])

{

<tr>

<td>@emp.Id</td>

<td>@emp.Name</td>

<td>@emp.Salary</td>

</tr>

}

</table>

**Working with ViewBag**

1. Viewbag is used to pass data from controller to view
2. Viewbag added in mvc 4
3. Viewbag creates the dynamic property
4. Viewbag does not required type casting
5. Viewbag does not required null value checking

ViewBag.Message=”sample message”;

Public dynamic Message {get;set;}

| ViewData | ViewBag |
| --- | --- |
| MVC 2 | MVC 4 |
| Need type casting | Does not required type casting |
| Need null value checking | Does not required null value checking |
| Stores data in the form of dictionary  Key→string  Value → object | Stores data as dynamic property  Public dynamic MyProperty {get;set;} |

Appsettings.json

"ConnectionStrings": {

"DefaultConnection": "Server=;Database=;Trusted\_Connection=True;TrustServerCertificate=True"

},

Design form in asp.net core

Two ways

1. Using html tags
2. Using html helper methods of mvc application

@HTML interfaces →

@Html.TextBox(“fname”) → <input type=”text” name=”fname” id=”fname”/>

DropDownList

Radio

Checkbox

Html.BeginForm() → <form> </form>

using Microsoft.AspNetCore.Mvc.Rendering;

public class PersonController : Controller

{

[HttpGet]

public IActionResult PersonalDetails()

{

List<string> list = new List<string>();

list.Add("Select Option");

list.Add("Yes");

list.Add("No");

ViewData["option"]=new SelectList(list);

return View();

}

[HttpPost]

public IActionResult PersonalDetails(IFormCollection form,ICollection<string> hobbies)

{

ViewBag.Name = form["fname"];

ViewBag.Gender = form["gender"];

ViewBag.Hobbies = hobbies;

ViewBag.Option = form["option"];

return View("Details");

}

}

<h1>Personal Details</h1>

@using(Html.BeginForm()){

<p>

<label>Enter your name</label> :

@Html.TextBox("fname")

</p>

<p>

<label>Select Gender</label> :

@Html.RadioButton("gender","Male") Male

@Html.RadioButton("gender","Female") Female

</p>

<p>

<label>Select Hobbies</label> :

<input type="checkbox" name="hobbies" value="Sports"/> Sports

<input type="checkbox" name="hobbies" value="Music"/> Music

<input type="checkbox" name="hobbies" value="Study"/> Study

</p>

<p>

<label>Do you want to participate in survey ?</label> :

@Html.DropDownList("option")

</p>

<p>

<input type="submit" value="Save" class="btn btn-primary"/>

</p>

}

Details.cshtml

<h1>Thanks for Survey</h1>

<p>

Your Name : @ViewBag.Name

</p>

<p>

Your Gender :@ViewBag.Gender

</p>

<p>

Hobbies :

<ul>

@foreach (var item in ViewBag.Hobbies)

{

<li>

@item

</li>

}

</ul>

</p>

<p>

Add in Survey : @ViewBag.Option

</p>

Emp name - textbox

Contact - textbox

Email - textbox

Gender - radiobutton

Exp – >dropdownlist

Projects → checkbox

Working with ADO.NET

using System.Data.SqlClient;

namespace CRUDWithAdoNet.Models

{

public class CustomersDAL

{

private readonly IConfiguration configuration;

SqlConnection con;

SqlDataReader dr;

SqlCommand cmd;

public CustomersDAL(IConfiguration configuration)

{

this.configuration = configuration;

con = new SqlConnection(this.configuration.GetConnectionString("defaultConnection"));

}

public IEnumerable<Customers> GetAllCustomers()

{

List<Customers> custlist = new List<Customers>();

string qry = "select \* from Customers";

cmd = new SqlCommand(qry, con);

con.Open();

dr=cmd.ExecuteReader();

if (dr.HasRows)

{

while (dr.Read())

{

Customers c = new Customers();

c.Id = Convert.ToInt32(dr["id"]);

c.Name = dr["name"].ToString();

c.Contact = dr["contact"].ToString();

c.Email = dr["email"].ToString();

c.Password = dr["password"].ToString();

custlist.Add(c);

}

}

con.Close();

return custlist;

}

public Customers GetCustomerById(int id)

{

}

public int AddCustomer(Customers customers)

{

}

public int UpdateCustomer(Customers customers)

{

}

public int DeleteCustomer(int id)

{

}

}

}

LINQ (Language Integrated Query)

Query→ expression which will store & manage data from the data source

Integrated the query in C# lang format (object format)

We can manage data from various data sources

RDBMS (MS-SQL)

XML

Collections

File

* Fetch
* Filter
* Sort
* Group
* Join data from multiple table

Product table→ id, name, price

SQL → select \* from product

LINQ →

Var result=from p in product

Select p;

Lambda expression

Var res=product.ToList();

SQL → select \* from product where price >500

LINQ →

Var result = from p in product

Where p.Price >500

Select p;

Lambda

Var res= product.where(x=>x.Price>500).ToList();

SQL → select \* from product where price >500 order by name

LINQ →

Var res= from p in product

Where p.Price>500

Order by p.Name descending

Select p;

Lambda Expression:

Var res=product.where(x=>x.price>500).Orderby(x=>x.name).ToList();

SQL → select name,price from product where price>500

Var res= from p in product

Where p.price>500

Select new{

Name=p.Name,

Price=p.Price

}

SQL → select e.name,e.salary ,d.name from emp e

Inner join dept d on d.did=e.did

LINQ →

Var res=from e in emp

Join d in dept

On d.did equals e.did

Select new Product{

Name=e.name,

Salary=e.salary,

Dname =d.dname

}

SQL → select \* from product where price=200

Var res= from p in product

Where p.price==200

Select p;

Entity Framework:

(ORM based)

Object relational mapping

EF run on the top of ADO.NET to store & manage data

Object C#

Table: Product

Id Name Price

Class Product{

Public int Id{get;set;}

Public string Name{get;set;}

Public int Price {get;set;}

}

DbContext ← → Database

(dbcontext is responsible for configure the connection string, manage the connection, fetch the data, update the data, insert the data)

Class ← → Table

Property ← → Column

LINQ to Entity Framework → if we want to manage data from RDBMS database we can use LINQ extension as LINQ to EF

**HomeWork:**

Create an asp.net mvc application ,

Configure the connection string

Add Model class Product

Add ProductDAL class with methods

1. ProductList
2. AddProduct
3. UpdateProduct
4. DeleteProduct
5. GetProductById

Create a controller ProductController → select read write actions

Create a View for

1. Create
2. Edit
3. List
4. Delete
5. Details

**Web API:**

**What is a web service ? →** service applications are used to interchange data between applications.

E-comm → payment → **Service** (payment service) → net banking → Bank

.net app → Service → java

SOAP format

Web based application

RESTful service

1. REST → Representational state transfer
2. It is mapped with HTTP request
3. HTTP methods are mapped with the req of Restful service
4. GET -> select
5. POST → insert
6. PUT → modify / update
7. DELETE → delete the data
8. JSON format will be used to send the data
9. Restful service can be consumed in any device
10. If we want to create a restful service then we will use Web API

Web API

1. It does not have views
2. We will create API controller
3. We will use attribute based routing

localhost:5400/api/product/getallproducts

localhost:5400/api/product/addproduct

Dependency injection (DI pattern)

DI is a design pattern where we will inject the dependency in the application with the help of constructor

* We will inject the dependency in the constructor

Interface IEngine

{

EngineType(string name);

NoOfCC();

}

Public class Engine:IEngine{

String name;

EngineType(string name)

{

this.name=name;

}

}

Interface ITire

{

Company();

}

Class Car

{

IEngine e;

ITire t;

Public Car(IEngine e, ITire t){ // inject the dependency

this.e=e;

this.t=t;

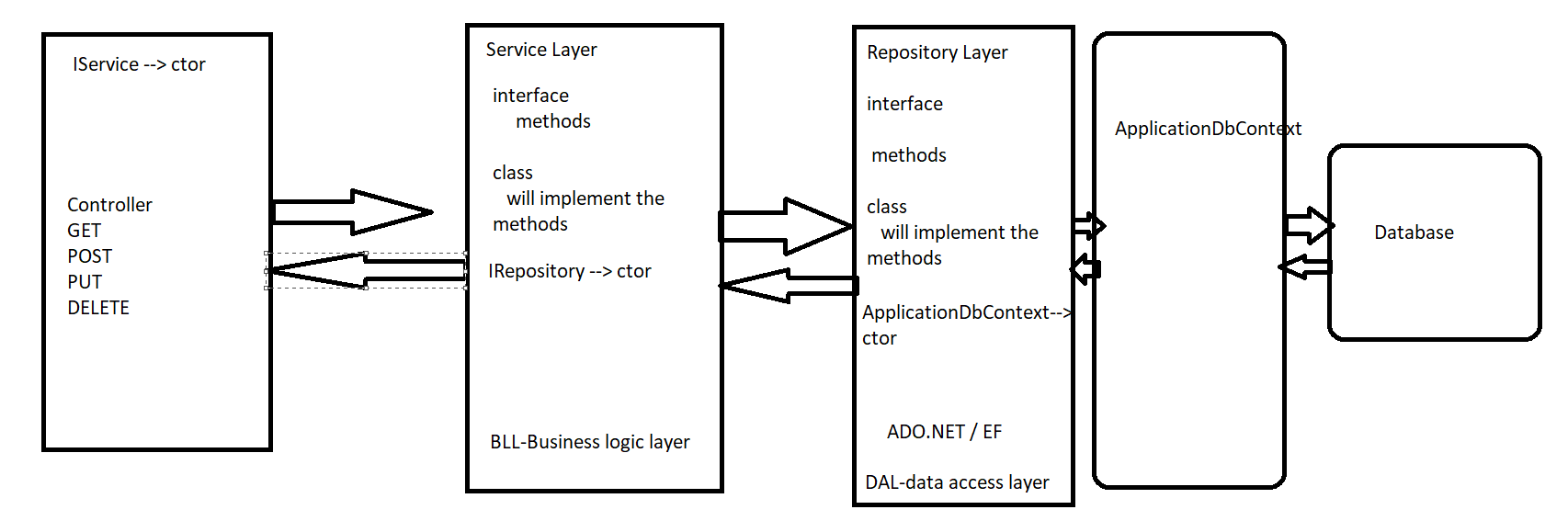
}

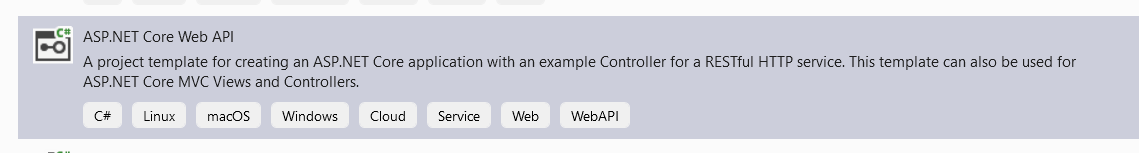
e.EngineType(Petrol);

}

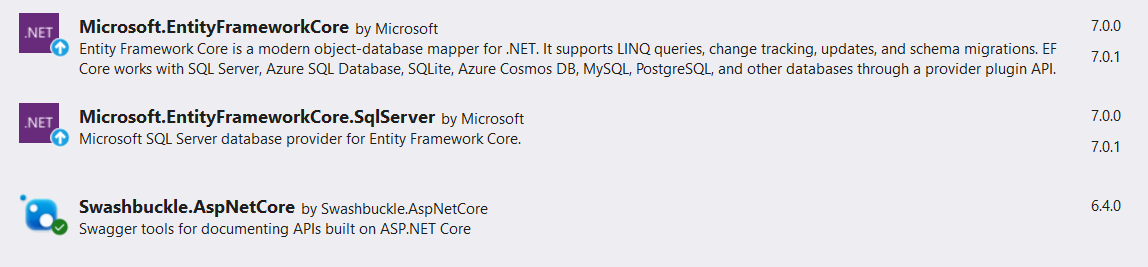
Car c=new Car();

C2





Step 1:



Step 2:



Step 3:

Program.cs

a.

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetSection("ConnectionStrings:DefaultConnection").Value));

B.

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "WebApiDemo", Version = "v1" });

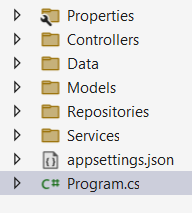
});

C.

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "WebApiDemo v1"));

Step 4:

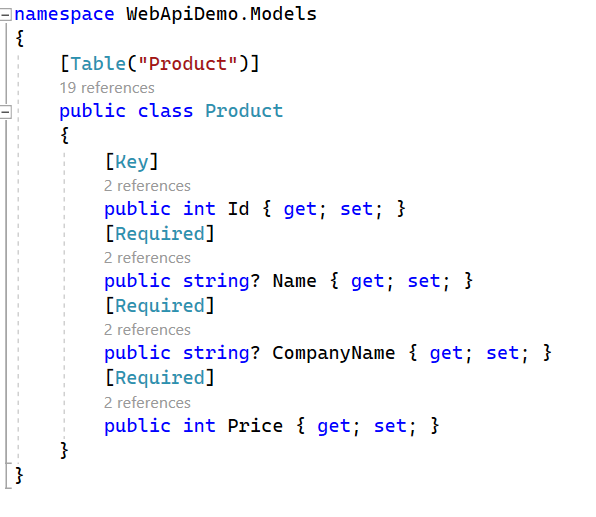
Create folder as below (Services, Repositories)



Step 5:

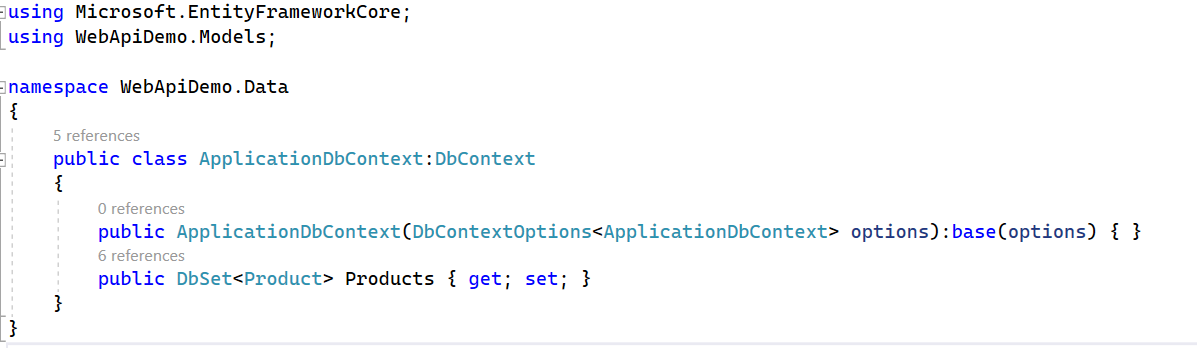
Models folder

Create class



Step 6:

Add below code in Data folder → ApplicationDbContext class



Step 7:

In the repositories folder add one interface & one class



using WebApiDemo.Models;

namespace WebApiDemo.Repositories

{

public interface IProductRepository

{

IEnumerable<Product> GetAllProducts();

Product GetProductById(int id);

int AddProduct(Product prod);

int UpdateProduct(Product prod);

int DeleteProduct(int id);

}

}

using WebApiDemo.Data;

using WebApiDemo.Models;

namespace WebApiDemo.Repositories

{

public class ProductRepository : IProductRepository

{

private readonly ApplicationDbContext \_db;

public ProductRepository(ApplicationDbContext db)

{

\_db = db;

}

public int AddProduct(Product prod)

{

\_db.Products.Add(prod);

int res= \_db.SaveChanges();

return res;

}

public int DeleteProduct(int id)

{

int res = 0;

var prod=\_db.Products.Find(id);

if (prod != null)

{

\_db.Products.Remove(prod);

res=\_db.SaveChanges();

}

return res;

}

public IEnumerable<Product> GetAllProducts()

{

return \_db.Products.ToList();

}

public Product GetProductById(int id)

{

var prod = \_db.Products.Find(id);

return prod;

}

public int UpdateProduct(Product prod)

{

int res = 0;

var p = \_db.Products.Where(x => x.Id == prod.Id).FirstOrDefault();

if (p != null)

{

p.Name = prod.Name;

p.Price = prod.Price;

p.CompanyName = prod.CompanyName;

res = \_db.SaveChanges();

}

return res;

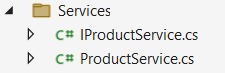
}

}

}

Step 8:

In services folders add one interface & one class



using WebApiDemo.Models;

namespace WebApiDemo.Services

{

public interface IProductService

{

IEnumerable<Product> GetAllProducts();

Product GetProductById(int id);

int AddProduct(Product prod);

int UpdateProduct(Product prod);

int DeleteProduct(int id);

}

}

using WebApiDemo.Models;

using WebApiDemo.Repositories;

namespace WebApiDemo.Services

{

public class ProductService : IProductService

{

private readonly IProductRepository \_repo;

public ProductService(IProductRepository repo)

{

\_repo = repo;

}

public int AddProduct(Product prod)

{

return \_repo.AddProduct(prod);

}

public int DeleteProduct(int id)

{

return (\_repo.DeleteProduct(id));

}

public IEnumerable<Product> GetAllProducts()

{

return \_repo.GetAllProducts();

}

public Product GetProductById(int id)

{

return \_repo.GetProductById(id);

}

public int UpdateProduct(Product prod)

{

return \_repo.UpdateProduct(prod);

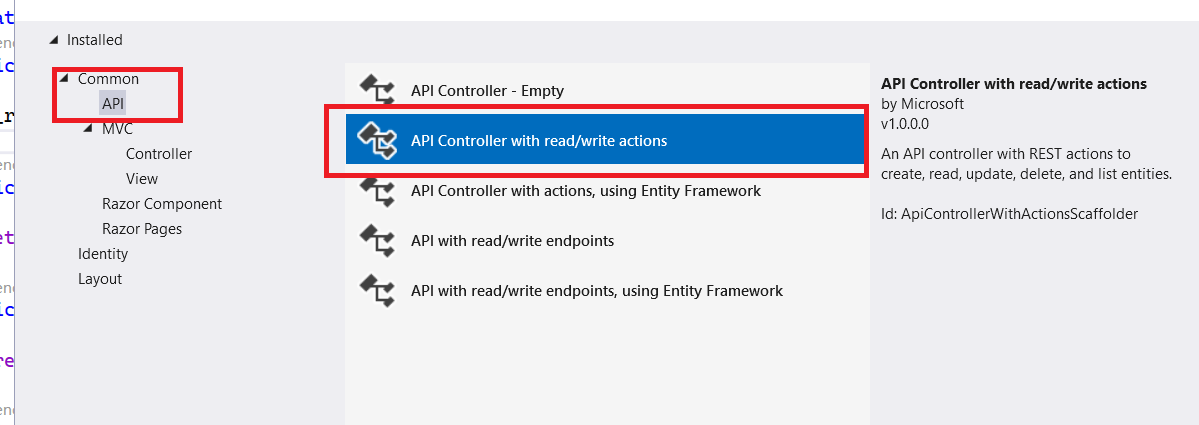
}

}

}

Step 9:

Add Controller



Complete the given code as below

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Models;

using WebApiDemo.Services;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace WebApiDemo.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductService service;

public ProductController(IProductService ser)

{

service = ser;

}

// GET: api/<ProductController>

[HttpGet]

[Route("GetAllProducts")]

public IActionResult GetAllProducts()

{

try

{

return new ObjectResult(service.GetAllProducts());

}

catch(Exception ex)

{

return StatusCode(StatusCodes.Status204NoContent, ex);

}

}

// GET api/<ProductController>/5

[HttpGet]

[Route("GetProductById/{id}")]

public IActionResult GetProductById(int id)

{

try

{

return new ObjectResult(service.GetProductById(id));

}

catch (Exception ex)

{

return StatusCode(StatusCodes.Status204NoContent, ex);

}

}

// POST api/<ProductController>

[HttpPost]

[Route("AddProduct")]

public IActionResult AddProduct([FromBody] Product prod)

{

try

{

int res=service.AddProduct(prod);

if (res == 1)

{

return StatusCode(StatusCodes.Status201Created);

}

else

{

return StatusCode(StatusCodes.Status500InternalServerError);

}

}

catch (Exception ex)

{

return StatusCode(StatusCodes.Status500InternalServerError, ex);

}

}

// PUT api/<ProductController>/5

[HttpPost]

[Route("UpdateProduct")]

public IActionResult UpdateProduct([FromBody] Product prod)

{

try

{

int res = service.UpdateProduct(prod);

if (res == 1)

{

return StatusCode(StatusCodes.Status200OK);

}

else

{

return StatusCode(StatusCodes.Status500InternalServerError);

}

}

catch (Exception ex)

{

return StatusCode(StatusCodes.Status500InternalServerError, ex);

}

}

// DELETE api/<ProductController>/5

[HttpGet]

[Route("DeleteProduct/{id}")]

public IActionResult DeleteProduct(int id)

{

try

{

int res = service.DeleteProduct(id);

if (res == 1)

{

return StatusCode(StatusCodes.Status200OK);

}

else

{

return StatusCode(StatusCodes.Status500InternalServerError);

}

}

catch (Exception ex)

{

return StatusCode(StatusCodes.Status500InternalServerError, ex);

}

}

}

}

Step 10:

Add below code in Program.cs file

builder.Services.AddControllers();

builder.Services.AddScoped<IProductRepository, ProductRepository>();

builder.Services.AddScoped<IProductService, ProductService>();

app.MapControllers();

Step 11:

Run the project & change the URL as below (Note: do not change the port number)

