پاسخنامه یاسمین آشوری

1- برای خارج شدن از بلوک switch – case از چه عباراتی می توان استفاده کرد ؟

عبارات return و break .

عبارت break باید در انتهای دستور case نوشته شود تا وقتی دستور case اجرا شد از بلوک خارج شویم و در غیر این صورت بقیه دستورات case نیز اجرا می‌شود و جایگزین دیگر آن عبارت return است که بعد از اجرای آن از بلوک خارج می‌شویم.

2- در چه شرایطی نیازی به دستور Console.ReadKey(); نمی باشد و بدون آن می‌توان خروجی برنامه را مشاهده کرد؟

1- در حالت Debug (عیب یابی): یکی از روش هایی که می توانیم خروجی را مشاهده کنیم و عیب یابی برنامه را نیز انجام دهیم استفاده کردن از متدهای دیباگ است که بدون آنکه نیاز به ران کردن پروژه کنسولی و باز نگه داشتن صفحه با Console.ReadKey داشته باشیم، خروجی برنامه را مشاهده و چاپ می کنیم. یا از منوی برنامه وارد حالت دیباگ شویم و خروجی را مرحله به مرحله چک کنیم.

3- ورژن های جدید تر C# : در ورژن 12 که تست کردم نیازی به نوشتن ReadKey نبود و بدون استفاده کردن از متدهایی شبیه آن مانند ReadLine، صفحه کنسول بسته نمی شود. گرچه هنوز می توان از این متد استفاده کرد اما الزامی نیست.

Scheduled Tasks - 2

می‌توان برنامه ای نوشت که وظایف برنامه‌ریزی‌شده که در فواصل زمانی مشخص اجرا می‌شوند،

بدون نیاز به دخالت کاربر، عملیات خودکار را انجام دهد.

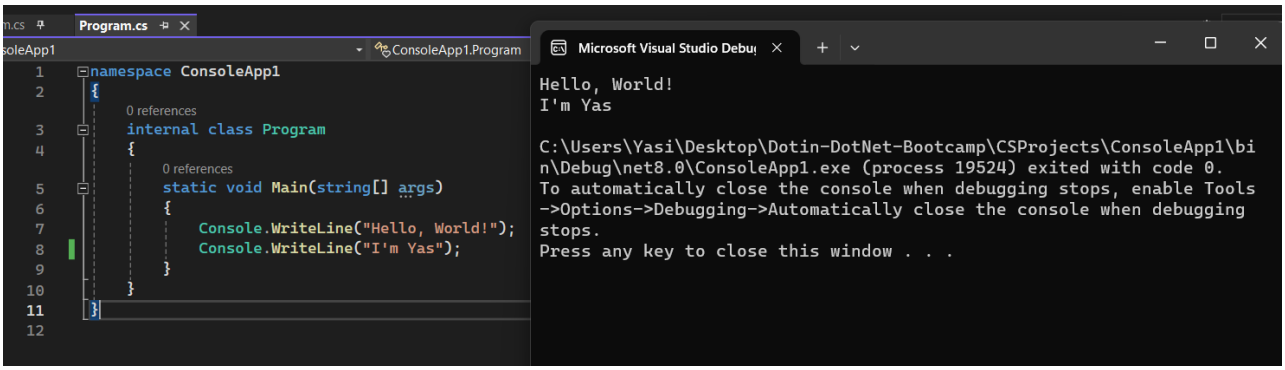
3- برنامه های غیر تعاملی

3- سه پیاده سازی برای ایجاد شرایط سوال قبل را نام ببرید (تکه کد کوتاه).

1- دیباگ:

System.Diagnostics,Debug.WriteLine(“H _ H1”);

2- ورژن های جدید



3. Scheduled Tasks

```
using System;
using System.Threading;

class Program
{
    static void Main()
    {
```

```
        Console.WriteLine("Scheduled Task Example: Printing a message every 2 seconds");
        Timer timer = new Timer(PrintMessage, null, TimeSpan.Zero, TimeSpan.FromSeconds(2));
        Console.ReadLine();
    }
    static void PrintMessage(object state)
    {
        Console.WriteLine($"Scheduled Task: Message printed at {DateTime.Now}");
    }
}
```

4- آرایه چیست و چه کاربردی دارد؟

آرایه‌ها (آرایه های کلاسیک)، یک سری خانه های رم حافظه که پشت سر هم هستند به شکل Single Memory Block هستند که به آن Index می‌گوییم و به هر خانه یک مقدار می‌دهیم. Rich API هستند و متد های زیادی دارند، آرایه ها دارای سایز ثابت و Zero based index یعنی ایندکس‌ها بر مبنای 0 هستند و از 0 شروع می‌شود و آرایه‌ها ذاتا Reference Type هستند یعنی وقتی دو رفرنس مساوی یکدیگر می شوند، آدرس آرایه سمت راست با آدرس آرایه سمت چپ یکی می‌شود و نه تنها مقادیر کپی می‌شوند بلکه آدرس‌ها هم در حافظه یکی می‌شوند.

کاربرد آن در مواقعی هست که برای مثال می خواهیم چندین مقدار را در کنار هم داشته باشیم و ساختن تعداد زیادی متغیر برای این کار روش عاقلانه‌ای نیست.

5- متدی را نام ببرید که نتیجه آن به شما یک آرایه باشد.

متد Split ورودی اش از جنس کاراکتر است نه لزوما آرایه کاراکتری ولی خودش کاراکتر ها را به یکدیگر می‌چسباند و به شکل آرایه ای از Separator ها میشه، ورودی‌اش params و خروجی هم آرایه ای از stringها است.

```
str.Split('-',  '=', ',');
```

6- دو روش برای تولید مقادیر تصادفی ؟

به کمک کلاس Random() که دو مدل Overload دارد سپس ساختن شئ از آن و استفاده از متدهای NextDouble, NextBytes(), Next().

به کمک Guid() که Overload های زیادی دارد سپس ساختن شئ از آن و مقدار دهی شئ به کمک **Guid.NewGuid();**

.نکته مهم این است که هیچ وقت تکراری نمی‌شوند

7- جایگاه دستورات break و continue کجاست ؟ عملکرد آنها چگونه است؟

جایگاه استفاده از این دستورات در برنامه هایی است که باید از یک حلقه همیشه درست، خارج شویم.

دستور break برای خارج شدن از حلقه (مثلا حلقه همیشه درست while(true))، (و همینطور در switch-case برای اجرا نشدن بقیه case ها) استفاده می‌شود. اگر یک حلقه while و یک شرط if-else داخل حلقه داشته باشیم، دستور break داخل شرط نه تنها باعث خارج شدن از دستور شرط بلکه باعث خارج شدن از دستور حلقه نیز می‌شود. مثلا در بازی حدس عدد هنگامی که جواب کاربر (شرط برابری حدس و عدد رندوم) درست است و دیگر نیازی به وارد کردن و حدس زدن عدد رندوم نبود با این دستور از حلقه خارج می‌شویم.

دستور continue نیز باعث خارج شدت از حلقه می شود اما تفاوت آن در این است که به جای خارج شدن از حلقه به ابتدای حلقه می رود و شرط حلقه را دوباره چک می کند و بعد از آن خارج می‌شود.

8- Overloading چیست؟

یعنی داشتن چندین متد با اسم های مشترک اما با امضای متد متفاوت. امضای متد یعنی تعداد، ترتیب و نوع پارامترهای ورودی (خروجی متد مهم نیست در امضای متد) است. نکته مهم دیگر نوشتن کلمه کلیدی ref پشت متد، باعث ایجاد Overloading میشود یعنی امضای متد را متفاوت می کند.

9- Overriding چیست؟

Override کردن یک متد یعنی تغییر دادن عملکرد یک متد به مدل دلخواه خودمان یعنی عملکرد متدهای virtual به ارث رسیده را می توان عوض کرد و هر چیزی رو یکبار می تونیم Override کنیم.

به کمک متد virtual انجام می‌شود و به کمک آن به بچه های یک کلاس این مجوز را می‌دهد که متد را Override بکنند و اگر نبود اجازه این کار را نداشتیم.

می‌توانیم برای مثال یک شرط بیزینس را داخل متد Override چک کنیم سپس دیتا ها را insert کنیم.

معروف ترین آن ها ToString(); است که می توانیم عبارت دلخواه خودمان رو بنویسیم.

10- Helper Method چیست ؟

یا متدهای سودمند متدهایی هستند که یک چیزی رو بر می‌گردونند و متدهایی هستند که در پروژه پرکاربرد هستند، پس با یک بار نوشتن این متد ها از اصل DRY به معنی تکرار نکردن کدها پیروی می کنیم و هر پروژه ای که به این متدها نیاز داشتیم از آن‌ها استفاده می‌کنیم. و اگر نیاز به تغییر هر کدام باشد فقط یکبار در داخل پروژه عوض میشود. برای ساختن این متدها یک روش توصیه شده وجود دارد که یک فولدر به اسم Utility درست کنید و کلاس با عملکردهای دلخواه به آن اضافه کنید برای مثال String Helper که عملکردهای مرتبط با

string ها را انجام می دهد. using ها را پاک می کنیم و یک متن مناسب کامنت بگذاریم و برای اینکه در خارج از اسمبلی و در پروژه های دیگر قابل استفاده شود کلاس را از internal به public تغییر می دهیم و نحوه استفاده از آن به این صورت است که در پروژه مورد نظر روی Add Reference کلیک کرده و کل فولدر Common که شامل Class Library و متدهای سودمند حاوی آن است را add می کنیم و با فراخوانی متد و using کردن نام Tools از متدها استفاده می کنیم.

11- Extension Method چیست ؟

کاربرد آن این است که کار تکراری کمتری انجام بدهیم و ورودی و خروجی متدها را کنترل کنیم، متد هایی که خودمان می نویسیم را به نوع داده ای که می خواهیم اضافه می کنیم. یعنی برای نوع داده ای که برای خودمان نیست متدها را اضافه کنیم و برای هر نوع داده ای یا هر Interface دلخواهی می تونیم متد هامون رو extend کنیم به تایپی که میخوایم و به عبارتی متد دلخواه نوشت. روش انجام دادن آن به این صورت است که روی پروژه CLass Library خودمان کلیک کنیم و یک فولدر به اسم Extension و یک کلاس ایجاد می کنیم برای مثال IEnumExt و DateTimeExt این کلاس برخلاف قبلا حتما باید استاتیک باشه (non-generic و public) و میخوام تبدیل به تاریخ شمسی برای نوع داده ای DateTime اضافه بشه this DateTime date و برای استفاده کردن ازش باید اول فولدر Extension رو use کرد و متد را فراخوانی کرد. اول کار Extension Method ها را می نویسیم و هر جا نیاز داشتیم اضافه اش می‌کنیم. برای انواع داده ای مثل IOrderedEnumerable که شبیه مجموعه هستن راه بهتر استفاده از extension method ها است به جای utility.

12- انواع داده یی value type و reference type چه تفاوتی دارند ؟ از هرکدام مثالی بزنید .

انواع داده‌ای value type در ساختمان داده Stack و reference type ها در ساختمان داده Heap ذخیره می‌شوند. انواع داده‌ای رفرنس، به صورت پیش فرض nullable هستند یعنی می‌توانیم مقادیر null داشته باشیم اما انواع value ها اینگونه نیستند و non nullable هستند.

انواع value شامل int, float, double, string, Array, Enum می شود و انواع struct مانند DateTime) با اینکه Complex هست ولی عملکردش Struct است) ولی Class, Function از نوع Reference هستند. در انواع value مقدار کپی می شود و اگر مقدار بعدا عوض شود دیتاها با وجود تغییر با یکدیگر مساوی نمی شوند چون هر دو به یک خانه حافظه اشاره نمی کنند! ولی در Reference ها، آدرس خانه حافظه یکی می شود پس هر خانه ای که آدرسش عوض شود، آدرس خانه حافظه شان یکی می شود چون هر دو به یک جا اشاره می کنند.

از استثناهای این انواع، String است که در عمل مانند Value Type ها عمل می‌کنند و Immutable هستند و وقتی ساخته می‌شوند دیگر تغییر نمی کنند اما نوع Reference مانند آرایه ها (در عمل مانند رفرنس هستند با اینکه در ValueType ها قرار گرفته اند) مقدار آرایه اصلی عوض میشود و به همین علت نیازی به return آرایه جدید نیست چون آرایه اولیه تغییر کرده است.

مثال برای value type:

```
int a = 10;
int b = a  // value 10
a++;  // value = 11
```

مثال برای reference type:

```
int[] numbers1 = new int[3];
numbers1[0] = 10;
numbers1[1] = 20;
numbers1[2] = 30;
```

```
int[] numbers2 = numbers1;

foreach (var item in numbers2)
{
   Console.WriteLine(item);
}

numbers1[1] = 2000;

foreach (var item in numbers2)
{
   Console.WriteLine(item);
}
Console.ReadKey();
```

13- تفاوت ArrayList و Generic List چیست ؟

تفاوت شان در این است که در لیست آرایه ها ما می توانیم هر نوع داده ای را به عنوان ورودی بدهیم پس لازم نیست نوع آن را مشخص کنیم، اما لیست های جنریک نوع داده ای آن را باید مشخص کرد.

لیست آرایه ها عضو کلاس مجموعه (Collection) هستند: System.Collections.Arraylist

و لیست های جنریک: <System.Collections.Generic.List<T

جنریک ها می توان هر نوع داده ای نوشت object اما باید نوعش را مشخص کرد اما کالکشن ها یا غیر جنریک ها همه چیز object هستن ولی نوعش رو نمی‌توانیم مشخص کرد.

همچنین از نظر سرعت، لیست های جنریک زمان کمتری نسبت به لیست آرایه ها برای پیمایش یک آرایه صرف می کنند.

14- LINQ چه کاربردی دارد ؟ یک تکه کد کوتاه و دلخواه از آن بنویسید.

به معنای کوئری ای هست که با زبان برنامه نویسی یکپارچه شده و می توان به کمک آن از هر مجموعه ای، آرایه معمولی یا Collection، کوئری گرفت. LINQ دارای تعداد زیادی Extension Method است که از قبل روی اون نوشته شده و همه اینها IEnummerable هستن مثل: Max, Min, Single

```
int[] numbers = new int[] { 1, 3, 5 };


var linqExample = (from n in numbers
                   select n).Distinct().Count();
```

15- شی گرایی چیست ؟

شئ گرایی یعنی مدل کردن دنیای واقعی به دنیای برنامه نویسی با هر زبانی که از آن پشتیبانی می‌کند، چون یک مفهوم است.

16- اجزای اصلی شی گرایی کدام است ؟

اجزای اصلی OOP:

1-Class 2-Attribute 3-Method 4-Object 5-Encapsulation

6-Inheritance 7-Abstraction 8- Polymorphism 9-Interface

برنامه‌نویسی شئ‌گرا 4 ستون اصلی دارد: یک رابطه is-a هست بین پدر و فرزند، و یک سری اطلاعات مشترک برای کلاس پدر داریم که کلاس های فرزند آنها آیتم های خاص و امکانات بیشتری دارند (property, method های بیشتر). از کلاس های sealed یک کلاس فرزند ایجاد کنیم.

2-Encapsulation

یک کلاس داریم که یک سری آیتم private دارد که از بیرون دیده نمی شود، پیاده سازی‌ها هم حتی قابل مشاهده نیست ولی می توان آنها را فراخوانی کرد.

3-Abstraction

جلوگیری از ایجاد نمونه از کلاس پدر

4- Polymorphism

چند ریختی اجازه می‌دهد یک object چندین نوع داده را ساپورت کند.

و همچنین قوانین SOLID: به ما کمک می کند تا از کدهای tightly coupled و با encapsulation کم به سمت کدهای loosely coupled و کپسوله شده حرکت کنیم.

17- تاثیر کلمه static بر روی کلاس ، خصوصیت و متد چیست ؟

وقتی یک کلاس از جنس static تعریف می‌شود دیگر نمی‌شود از آن شئ ساخت و باید فیلدها و متدهای آن استاتیک شوند و اگر یک کلاس استاتیک شود، property (خصوصیت) آن هم استاتیک می‌شود و باید حتما مقداردهی شود در همان داخل کلاس، تا بتوانیم از آن استفاده کنیم. پس دیگر وابسته به مقدار شئ نمی‌باشد و مستقیما یک خصیصه ای از خود آن کلاس می‌شوند.

فلسفه استفاده از static برای مثال در یک متد این است که وقتی نمی خواهیم جواب یک متد برای مثال وابسته به شئ باشد آن را استاتیک می کنیم تا هنگام فراخوانی متد خروجی به object بستگی نداشته باشد. اما متدهای غیر استاتیک خروجی شان وابسته به شئ ای هست که آن ها را صدا می زند.

متد ها یا کلاس های استاتیک جوابشان وابسته به فراخوانی کننده کلاس، consumer کلاس ندارد و جواب ثابت است. اما غیر استاتیک ها فقط از طریق شئ که از کلاس آن ساختیم، فراخوانی می‌شوند و دیتای آنها وابسته به instance است پس اگر یک متد داریم که فرقی ندارد که توسط چه شئ ای فراخوانی شود بهتر است آن را استاتیک کنیم. و متدهای استاتیک با کلاس شان فراخوانی می‌شوند پس کلا کلاس را هم از نوع استاتیک تعریف می‌کنیم که نتوان از آن object ساخت اگر یک کلاسی استاتیک باشد دیگر نمی‌توانیم از کلاس استاتیک یک شئ بسازیم.

18- اگر نیاز باشد نحوه مقدار دهی یک خصوصیت را محدود کنیم ، آن را چگونه تعریف کنیم؟

property (خصوصیت) محاسباتی یا خصوصیت read-only به این محدودیت در مقدار دهی گفته می‌شود مانند computed column که برای نمایش سن در sql داشتیم. پس به آن دیگر مقدار نمی‌دهیم و خودش مقدارش را بر می‌گرداند. خصوصیتی است که set ندارد و حتی اگر بخواهیم هم، نمی توانیم به آن مقداری بدهیم و فقط get دارد و مقداردهی نمی‌شود. نحوه تعریف کردن آن به شکل زیر است و چون هیچ مقداری به آن داده نمی شود، خودمان مقدارش را return می‌کنیم. ولی اگر نیاز به مقداردهی باشد به کمک تابع سازنده کلاس مقداردهی انجام می‌شود.

```
public string TitileFa
{
        get
        {
        return Gender ? "Mr. " : "Ms. ";
        }
}
```

19- شما به عنوان یک برنامه نویس ارشد چه زمانی از کلاس های Abstract استفاده می‌کنید؟

برای بحث امنیت و به خصوص Encapsulation، اگر بخواهیم جلوی ساخت نمونه object از کلاس پدر را بگیریم و فقط از کلاس های فرزند بتوانیم شئ ایجاد کنیم، می‌توانیم از کلاس های Abstract استفاده کنیم. در مثالی که گفته شد برای محاسبه محیط و مساحت اشکال هندسی مختلف، ما نیاز داریم که اسم های مختلف و ویژگی های مختلف برای محاسبه محیط و مساحت وجود نداشته باشد و یک متد خالی داشته باشیم با اسم مشخص که این کار را انجام می‌دهد و اشکال هندسی مختلف با فرمول خاص خودشان از این کلاس ابسترکت ارث بری کنند و از کلاس خودشان نمونه سازی کنند نه کلاس پدر!

نکات مهم:

نمی‌توان از یک کلاس Abstract شئ درست کرد و داخل کلاس Abstract متدهای Abstract داشت که بدنه دارند و متدهای ابسترکت حتما باید داخل کلاس ابسترکت باشند و پراپرتی هم، می‌تواند ابسترکت باشد و متدهای معمولی virtual نیز داشته باشیم.

20- شما به عنوان یک برنامه نویس ارشد چه زمانی از Interface ها استفاده میکنید؟

در واقع Interface ها قرارداد هستند برای انجام دادن انواع کارها مثلا: IComparer <T >

وقتی هر نوعی یا کلاسی این رو داشته باشه متوجه میشه چطور نمونه هایی از اون نوع رو با هم مقایسه کنه و هر Interface یک چیزی رو انجام میده و مزایای خوبی دارن چون میشه کدی نوشت که دو ویژگی مهم زیر رو داشته باشه:

1-loosely coupled

2- highly cohesive

باشه یعنی به همدیگر وابستگی نداشته باشد و پیوستگی بالا داشته باشد و کاربرد Interface همین کاهش وابستگی برای کلاس هاست. و چون قرارداد هستن پیاده سازی ندارن.

همچنین نمی‌شود از یک Interface شئ درست کرد.

21- شما به عنوان یک برنامه نویس ارشد چه زمانی از Class Library ها استفاده می‌کنید؟

زمانی که نمی‌خواهیم پروژه های رایج common داشته باشیم که به راحتی قابل اجرا شدن نیستن به تنهایی و به کمک بقیه پروژه ها کار می‌کنه، یک Tools و جعبه ابزار هستند سراغ پروژه های Class Library می‌رویم که یک پروژه است برای ساختن فایل های dll. فایل های exe) dll) فایل هایی نیستن که به تنهایی کاری انجام بدن و بقیه فایل ها از این exe استفاده می کنند. اسم این مدل پروژه ها یا اسم خودمان است یا کمپانی و برند، یک اسم عمومی که بقیه هم بتونن ازش در پروژه ها استفاده کنند و تفاوت محتوای این مدل پروژه ها نداشتن Program.cs و App.config است و با متد های خودمان محتوای این پروژه را ایجاد می‌کنیم و هر برنامه ای که به متدهای آن نیاز داشتیم به این پروژه رفرنس می‌دهیم و از آنها استفاده می‌کنیم. یا برای ViewModel ها باید از نوع کلاس لایبرری تعریف شود که برای نمایش به کار می رود و لزوما در دیتابیس به این شکل وجود ندارد.

22- ارکان اصلی افزودن یک رویداد دلخواه در کلاس کدامند ؟ هرکدام با توضیحی مختصر

برنامه نویسی مبتنی بر رویداد Event Driven تا زمانی که برای تمام المان ها، event ها رو هندل نکنم موقع کلیک کردن اتفاقی نخواهد افتاد. روی المان کلیک می کنیم event به برنامه نویس اجازه می ده که با هندل کردن لحظه های کلیدی یک life cycle در زمان اجرای برنامه پاسخ بدیم و به رویداد جواب بدیم. در وب یا ویندوز کاربر میتونه با ظاهر گرافیکی برنامه تعامل داشته باشه. و هر کدام از اینها یک رویداد رو فراخوانی میکنه و می تونی برنامه ای بنویسیم که به این واکنش ها پاسخ بده پس وقتی روی یک دکمه کلیک می کنیم یک اتفاقی بیافته پس یک رویداد موقع کلیک فراخوانی میشه و متدی رو ایجاد می کنم که به اون رویداد attach میشه.

رویدادها به کمک Delegate انجام میشوند یعنی یک pipeline هستند برای رساندن رویداد به event handler ها.

1- تعریف یک Event:

```
public event WorkPerformedHandler WorkPerformed;
```

رویدادها را می توان در یک کلاس با استفاده از کلمه کلیدی رویداد تعریف کرد.

2- تعریف کردن یک رویداد با Add/Remove accessor:

```
private WorkPerformedHandler _WorkPerformedHandler;

public event WorkPerformedHandler WorkPerformed

{

[MethodImpl(MethodImplOptions.Synchronized)]

add

{

_WorkPerformedHandler = (WorkPerformedHandler)Delegate.Combine(

_WorkPerformedHandler, value);

}

[MethodImpl(MethodImplOptions.Synchronized)]

remove

{

_WorkPerformedHandler = (WorkPerformedHandler)Delegate.Remove(

_WorkPerformedHandler, value);
```

```
}
}
```

Raising Events-3:

رویدادها با فراخوانی رویداد مانند یک متد raised می شوند یا راه حل دیگر دسترسی به event delegate و فراخوانی مستقیم است:

```
if (WorkPerformed != null) {
WorkPerformed(8, WorkType.GenerateReports);
}
```

4- Custom EventArgs Class ساخت

کلاس EventArgs در امضای بسیاری از delegate ها و event handler ها استفاده می شود.

```
public void button_Click(object sender, EventArgs e)
{
// Handle button click
}
```

وقتی که داده های سفارشی باید ارسال شوند، کلاس EventArgs را می توان extend کرد.

مشتق شده از کلاس System.EventArgs

```
public class WorkPerformedEventArgs : System.EventArgs
{
public int Hours { get; set; }
public WorkType WorkType { get; set; }
```

```
}
```

5-استفاده کردن از System.EventArgs Class مشتق شده:

برای استفاده از کلاس EventArgs سفارشی، delegate باید به کلاس در امضای خود ارجاع داده شود:

```
public delegate void WorkPerformedHandler(object sender,
WorkPerformedEventArgs e);
```

همچنین می‌توانیم در .NET از جنریک EventHandler<T> استفاده کنیم.

```
public event EventHandler<WorkPerformedEventArgs> WorkPerformed;
```

23- تفاوت و شباهت Func و Action در چیست ؟

شباهت ها: هر دو از نوع Generic Delegate هستند که برای متدهای Callback و کنترل events ها و تزریق وابستگی استفاده می شوند. استفاده می شوند که ارائه دهنده متدهایی با امضای متد خاص هستند: specific signatures

در متدهای Action، می توانیم صفر یا تعدا زیادی پارامتر ورودی داشته باشیم ولی چیزی برگردانده نشود و void هستند ولی در متدهای Func، صفر یا تعداد زیادی پارامتر داریم که یک مقدار برمی گردانند و می تواند هر نوع داده ای داشته باشد.

24- چگونه میتوان یک لیست دلخواه از هر نوعی را با متد Sort مرتب کرد ؟

با کمک لیست های جنریک و Interface متد Sort (چندین overload دارد که یکی از آن ها این Interface است) به اسم IComparer می توانیم مشخص کنیم بر چه اساسی مقایسه انجام بشود. می‌توان هر نوعی از داده را با متد داخلی Sort خودش مرتب کرد زیرا ورودی آن object است و همه انواع داده ای آن را دارند.

ورودی اول آرایه و ورودی دوم Interface است، نحوه استفاده آن به این صورت است که یک کلاس دلخواه داشته باشیم که اینترفیس را پیاده سازی کند و یک متغیر از آن بسازیم و پاس بدهیم.

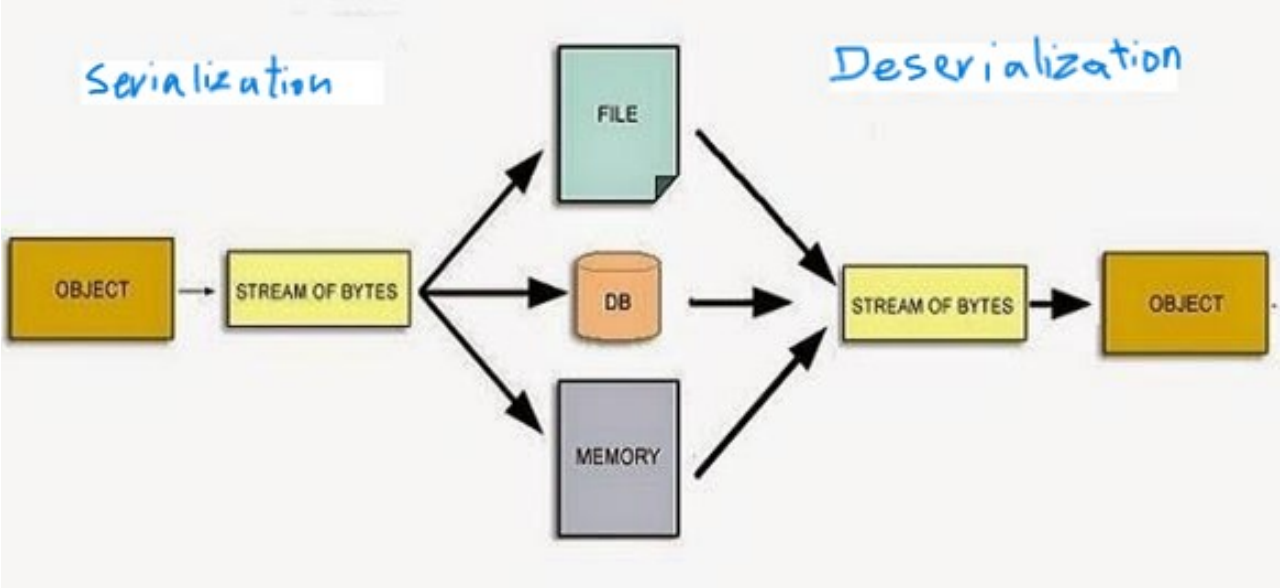
25- متد Peek در ساختمان داده صف ، پشته و همچنین در کلاس TextReader چه عملکردی دارد؟

در ساختمان داده پشته متد peek فقط برای اطلاع دادن مکان آیتم است و مکان بالای پشته Top رو return می‌کند بدون اینکه حذف شان کند.

و همچنین peek در ساختمان داده صف، object را تحویل می‌دهد (تحویل می‌دهد اما return نمی‌کند) بدون اینکه remove کند ولی در ساختمان داده صف object انتهای صف هست برخلاف پشته که آیتم بالای پشته بود.

متد Peek کاراکتر بعدی را بدون تغییر وضعیت reader یا منبع کاراکتر می خواند. یعنی کاراکتر بعدی را بدون خواندن آن از جریان ورودی، نگاه می‌کند و برمی‌گرداند.

26- تصویر زیر را با کلمات مناسب تکمیل کنید



serialization:

عملیات تبدیل کردن یک object داخل stream (جریانی) از بایت هاست در جهت اینکه ذخیره یا منتقل بشه به یک حافظه یا یک دیتابیس یا یک فایل. پس روند تبدیل یک object است که وضعیت اون object رو ذخیره کنیم هر وقت خواستیم و داشته باشیم اش. به روش های مختلفی انجام میشه مثل باینری یا جیسون.

Deserialization:

به عملیات عکس روش بالا گفته می‌شود.

27- اشتباه این تکه کد کدام است ؟

```
1.  Static Void Main(String[] args)
2.  {
3.      const int m = 100;
4.      int n = 10;
5.      const int k = n / 5 * 100 * n ;
6.      Console.WriteLine(m * k);
7.      Console.ReadLine();
8.  }
```

a) ‘k’ should not be declared constant
b) Expression assigned to ‘k’ should be constant
c) Expression (m * k) is invalid
d) ‘m ‘ is declared in invalid format

28- خروجی تکه کد زیر چیست ؟

```
1.   static void Main(string[] args)
2.   {
3.       int i ;
4.       for (i = 0; i < 5; i++)
5.       {
6.           int j = 0;
7.           j += i;
8.           Console. WriteLine(j);
9.       }
10.      Console. WriteLine( i * j);
11.      Console. ReadLine();
12.  }
```

a- 0, 1, 6, 18, 40

b- 0, 1, 5, 20, 30

c- Compile time error

d- 0, 1, 2, 3, 4, 5

29- خروجی تکه کد زیر چیست ؟

```
0 references
static void Main(string[] args)
{
    int i = 0;
    while (i <= 50)
    {
        if (i % 10 == 0)
            continue;
        else
            break;
        i += 10;
        Console.WriteLine(i % 10);
    }
}
```

a- code prints output as 0 0 0 0 0

b- code prints output as 10 20 30 40 50

c- infinite loop but doesn’t print anything

d- Code generate error

30- خروجی تکه کد زیر چیست ؟

```
0 references
static void Main(string[] args)
{
    int[] numbers = new int[] { 2, 4, 6, 8 };
    Console.Write(numbers[numbers.Length]);
}
```

a- 4
b- 8
c- Compile Error
d- Runtime Error

موفق باشید