**بسمه تعالی**

**Stock Market Engine:**

دو صف خرید و فروش جهت انجام معاله سهام یک نماد وجود دارد. در صورتی که قیمت سفارش خرید مساوی یا بیشتر از قیمت سفارش فروش باشد معامله با قیمت فروش انجام میشود و یا قیمت سفارش فروش مساوی یا کوچکتر از قیمت سفارش خرید باشد معامله با قیمت فروش انجام میشود.

**Priority Queue:**

در سی شارپ نسخه 10 صف با الویت معرفی شد که الگوریتم مرتب سازی آن براساس درخت باینری می باشد. بدین صورت که صف براساس اولویت که در زمان ایجاد و دارای دو دستور کلیدی برای ورود به صف یا خروج از صف می باشد. و کد زیر نمونه کد برای ایجاد یک صف با الویت می باشد.

به عنوان مثال برای حل مشکل ایجاد صف خرید یا فروش با اولویت قیمت و ای دی میتوان از priority queue استفاده کرد.

var patients = new List<(Patient, int)>()

{

(new("Sarah", 23), 4),

(new("Joe", 50), 2),

(new("Elizabeth", 60), 1),

(new("Natalie", 16), 5),

(new("Angie", 25), 3)

};

var hospitalQueue = new PriorityQueue<Patient, int>(patients); **تعریف صف با اولویت**

hospitalQueue.Enqueue(new Patient("Roy", 23), 5); **ورود به صف**

var highestPriorityPatient = hospitalQueue.Dequeue(); **خروج از صف**

**نکته: در سی شارپ صف الویت دار به صورت min heap مرتب می شود و برای مرتب سازی به صورت max heap باید از اینترفیس Icomperar استفاده کرد.**

**Min heap: درخت براساس کم ارزش ترین اولویت مرتب می شود (نزولی)**

**نمونه مثال برای مرتب سازی max heap**

public class ModifiedOrderPriorityMax : IComparer<Order>

{

public int Compare(Order? x, Order? y)

{

if (x.Price == y.Price)

{

return 0;

}

else if (x.Price < y.Price)

{

return 1;

}

else

{

return -1;

}

}

}

**برای درک صحیح تر می توانید به سایت زیر مراجعه نمایید.**

<https://code-maze.com/csharp-priority-queue/>

**دیزاین پترن State**

**Blocking queue:**

**یک کالکش thread safe هست که پترن producer/consumer را پیاده سازی میکند. بدین معنی که producer دیتا را ایجاد میکند و consumer دیتا را فراخوانی میکند این صف مشکل هم زمانی را حل میکند.**

**به عنوان مثال در مساله صف های فروش و خرید چند سفارش خرید یا فروش هم زمان بخواهد بدون مشکل ثبت شوند میتوانیم از blocking queue استفاده کنیم.**

**Thread safe: چند thread همزمان میتوانند یک قطعه کد اجرا کنند بدون اینکه عملکرد برنامه مختل شود.**

**نحوه ایجاد blocking queue**

**BlockingCollection<int> bCollection = new BlockingCollection<int>();**

**افزودن به صف**

bCollection.Add(1);

**حدف از صف**

bCollection.Take();

**تعریف صف با ظرفیت:**

**BlockingCollection<int> bCollection = new BlockingCollection<int>();**

**نکته: در صورتی که بخواهیم بیشتر از ظرفیت تعریف شده, آیتم اضافه کنیم میتوانیم از مند tryadd استفاده کنیم. بدین صورت که آیتم جدید تا زمانی که یک آیتمی که در صف وجود داشته باشد نمی تواند وارد شود و همچنین می توان برای افزودن آیتم جدید یک زمان معین در نظر گرفت.**

**نکته: در صورتی صف خالی باشد و بخواهیم منتظر بمانیم تا آیتم جدید اضافه شود میتوان از trytask استفاده کرد و همجنین میتوان برای آن زمان معین در نظر گرفت.**

**نکته: در thread producer زمانی که تمامی آیتم ها اضافه کردیم و آیتمی قرار نبود اضافه شود باید متد CompleteAdding صدا زد شود که در thread consumer بدانیم صف خالی شده است و زمانی که متد CompleteAdding صدا بزنیم مقدار IsCompleted مساوی با true میشود.**

**نمونه کد برای ایجاد و خواندن از صف:**

static void Main(string[] args)

{

BlockingCollection<int> bCollection = new BlockingCollection<int>(boundedCapacity: 10);

Task producerThread = Task.Factory.StartNew(() =>

{

for (int i = 0; i < 10; ++i)

{

Thread.Sleep(TimeSpan.FromSeconds(1));

bCollection.Add(i);

}

bCollection.CompleteAdding();

});

foreach (int item in bCollection.GetConsumingEnumerable())

{

Console.WriteLine(item);

}

//Output:

// 0

// 1

// 2

// 3

// 4

// 5

// 6

// 7

// 8

// 9

}

**برای درک صحیح تر میتوانید به سایت زیر مراجعه نمایید.**

<https://dotnetpattern.com/csharp-blockingcollection>

**دیزاین پترن command**

[دیزاین پترن](https://holosen.net/what-is-design-pattern/) Command یکی از زیرشاخه های الگوهای طراحی از نوع Behavioral است بدین صورت که یک درخواست داخل یک ابجکت محصور میشود که اسم آن کامند است و ابجکت فرخوان کننده دنبال ابجکتی میگردد که میتواند این دستور اجرا کند. سپس دستور به ابجکت مورد نظر برای اجرا ارسال میکند.

مثال: در سفارش گذاری می توان از این دیزاین پترن استفاده کرد بدین شکل که در زمان سفارش گذاری میتوانیم سفارش خود را ویرایش کنیم حذف کنیم یا ثبت کنیم این کارا میتوان با ساختن یک کلاس پدر و چند کلاس فرزند انجام داد اما کد دارای خوانایی مناسب نیست به همین دلیل میتوان یک کلاس به عنوان کامند در نظر گرفت که آن کلاس با توجه به دستوری که به آن وارد میشود عملیات مود نظر انجام دهد.

**ConfigureAwait**

Task.ConfigureAwait(bool) باید false باشد علت این است که در برنامه نویس async معمولا thread که تسک ایجاد کرده است ,در صورت داشتن await , همان thread ادامه تسک انجام میدهد و اینکار هزینه زیادی دارد و باعث برخی مشکلات مانند deadlock میشود بنابراین مقدار آن باید false باشد.

**Interlock.Increment**