
بسم الله الرحمن الرحيم

کلاس LoadFileForChart

نویسنده

محمدحسین کشاورزباحقیقت

گزارش مرداد ماه

۱۴۰۱

فهرست مطالب

عنوان **صفحه**

فهرست مطالب . . . . . سه

فصل اول : توضیحات کد

۱-۱ داده‌های ورودی و متدهای سازنده ۱

۱-۲ ویژگی‌ها ۲

۱-۳ متدها ۳

۱-۴ الگوی پیاده‌سازی شده ۵

فصل اول

توضیحات کد

هدف از پیاده‌سازی این کلاس دریافت یک فایل ورودی و پردازش داده‌های موجود در آن برای رسم داده‌های دریافتی می‌باشد.

۱ - ۱ داده‌های ورودی و متدهای سازنده

در این کلاس دو متد سازنده پیاده سازی شده است. تفاوت دو متد پیاده سازی شده در دریافت مقدار مشخصی برای بافر مورد استفاده برای بارگذاری داده‌ها از فایل اولیه و یا مقداردهی خودکار برای این امر می‌باشد. اولین داده ورودی مورد استفاده در این کلاس آدرس فایل مورد نظر می‌باشد. دومین داده ورودی تعیین می‌کند که نمونه گیری مجدد با چه درصدی نسبت به داده‌های اولیه صورت گیرد. سومین داده ورودی حجم داده خروجی اولیه را مشخص می‌نماید. همچنین در صورت تمایل می‌توان اندازه بافر مورد استفاده در عملیات بارگذاری و تبدیل فایل را مشخص نمود. در صورتی که مقدار داده مذکور تعیین نشود، به طور خودکار یک مقدار اولیه به آن اختصاص داده خواهد شد.

Code 1-1: First Constructor(without BufferSize)

```
public LoadFileForChart(string filePath,int reSamplePercentage,int
    examplePercent)
        {
            try
            {
                FileInfo fileInfo = new FileInfo(filePath);
                this.FileName = fileInfo.FullName;
                this.FileSize = fileInfo.Length;
                this.ReSamplePercentage = reSamplePercentage;
                this.ExamplePercent = examplePercent;
                this.ExampleSize = FileSize * ExamplePercent / 100;
                this.LoadBufferSize = 1024 * 1024;
                this.ReSampleRate = 100/reSamplePercentage;
            }
            catch
            {
                Console.WriteLine("couldnt get file data!");
            }

        }
```

Code 1-2: Second Constructor(with BufferSize)

```
public LoadFileForChart(string filePath,int reSamplePercentage,int
    examplePercent)
        {
            try
            {
                FileInfo fileInfo = new FileInfo(filePath);
                this.FileName = fileInfo.FullName;
                this.FileSize = fileInfo.Length;
                this.ReSamplePercentage = reSamplePercentage;
                this.ExamplePercent = examplePercent;
                this.ExampleSize = FileSize * ExamplePercent / 100;
                this.LoadBufferSize = 1024 * 1024;
                this.ReSampleRate = 100/reSamplePercentage;
            }
            catch
            {
                Console.WriteLine("couldnt get file data!");
            }

        }
```

۲-۱ ویژگی‌ها

در این کلاس از ۷ ویژگی استفاده می‌شود. در این بخش به معرفی کارکرد این ویژگی‌ها خواهیم پرداخت.

- FileName

این ویژگی برای ذخیره‌سازی آدرس و نام فایل استفاده می‌شود.

- LoadBufferSize

این ویژگی اندازه بافر مورد استفاده در بارگذاری فایل را مشخص می‌نماید.

- FileSize

این ویژگی اندازه فایل ورودی را نگهداری می‌کند.

- ReSamplePercentage

این ویژگی تعیین می‌کند که نمونه گیری مجدد با چه درصدی نسبت به داده‌های اولیه صورت گیرد.

- ReSampleRate

این ویژگی که توسط متد سازنده مقداردهی می‌شود، درصد دریافتی در ویژگی ReSamplePercentage را در قالب یک عدد ثابت ذخیره‌سازی می‌کند.

- ExamplePercent

این ویژگی مشخص کننده نسبت فایل خروجی اولیه به فایل خروجی اصلی بر حسب درصد می‌باشد.

- ExampleSize

این ویژگی اندازه عددی فایل خروجی اولیه را در خود ذخیره‌سازی می‌کند.

۳-۱ متدها

در این کلاس از چند متد برای دریافت و تبدیل فایل اولیه استفاده شده است که تفاوت این متدها در شیوه بارگذاری فایل اولیه می‌باشد. هریک از این متدها فایل اولیه را در قالب یک نوع(Double،Int،Byte،.....) مشخص می‌خوانند.

- متد ProcessDataByte

این تابع داده موجود در فایل را بر مبنای byte خوانده و عملیات مورد نظر ما را بر روی آن اعمال می‌کند و یک آرایه نهایی برای ترسیم در اختیار ما قرار می‌دهد. در این متد در ابتدا یک شئ از جنس BinaryReader ایجاد می‌گردد که وظیفه خواندن داده از فایل اولیه را برعهده دارد. همچنین دو آرایه fileContent و converteData نیز به وجود می‌آیند. متد مذکور در هر مرحله از یک چرخه که بر مبنای اندازه فایل ابتدایی صورت می‌گیرد، به اندازه بافر تعریف شده در ویژگی LoadBufferSize از داده موجود در فایل اولیه را بصورت byte خوانده و در متغییر fileContent ذخیره می‌کند. در مرحله بعدی داده‌های موجود در متغییر fileContent را با احتساب نسبت تعیین شده برای نمونه‌گیری مجدد داده‌ها، فیلتر کرده و در متغییر convertedData ذخیره می‌نماید. همچنین در هر مرحله با محاسبه میزان پیشرفت عنصر خواننده نسبت به حجم فایل ابتدایی و ذخیره‌سازی آن در متغییر progress نسبت به ترسیم یک bar progress برای نمایش میزان پیشرفت اقدام می‌شود. تعداد دفعات اجرای این چرخه برحسب نسبت اندازه فایل ورودی به اندازه بافر تعریف شده در کلاس محاسبه می‌شود.

- متد processDataDouble

عملکرد کلی این تابع همانند تابع قبل می‌باشد. با این تفاوت که این تابع پس از ایجاد شی از Bi-naryReader نیاز به استفاده از یک حلقه for برای خواندن داده بصورت نوع داده double خواهد داشت، چراکه کلاس binaryreader توانایی خواندن یک آرایه از نوع داده double را ندارد و این تابع بصورت مرحله ای باید پیاده‌سازی شود

- متد processDataInt۳۲

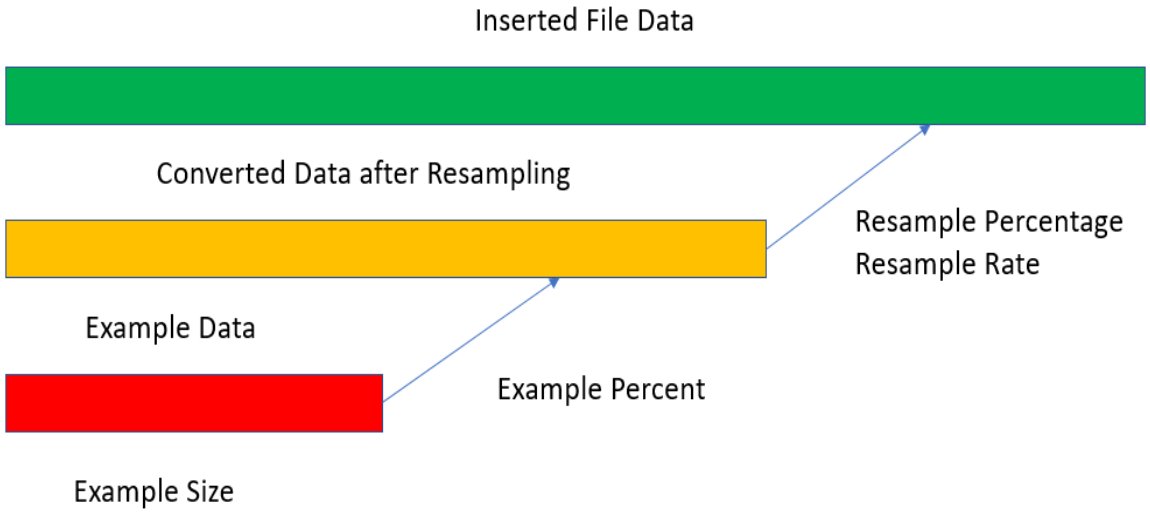
عملکرد این متد همانند متد processDataDouble می‌باشد.

- متد processDataInt۱۶

عملکرد این متد همانند متد processDataDouble می‌باشد.

۱ - ۴ الگوی پیاده‌سازی شده

این بخش با هدف درک بهتر مباحث توضیح داده شده در بخش‌های قبلی ایجاد شده است. شکل زیر ارتباطات میان داده اولیه و نتایج خروجی نمونه برداری مجدد و همچنین داده خروجی اولیه را نمایش می‌دهد. همانطور که در شکل شماهده می‌شود، مقدار ورودی Percentage Sample نسبت داده دریافتی از فایل ورودی و داده خروجی عملیات نمونه برداری مجدد را تعیین می‌کند. همچنین به واسطه مقدار ورودی Percent Example مشخص می‌شود که چه میزان از داده‌های خروجی عملیات نمونه‌گیری مجدد می‌بایست در خروجی(ترسیم) اولیه به نمایش گذاشته شود(شکل۱ - ۱).



شکل ۱ - ۱: نمودار الگوی نمونه گیری مجدد و خروجی اولیه