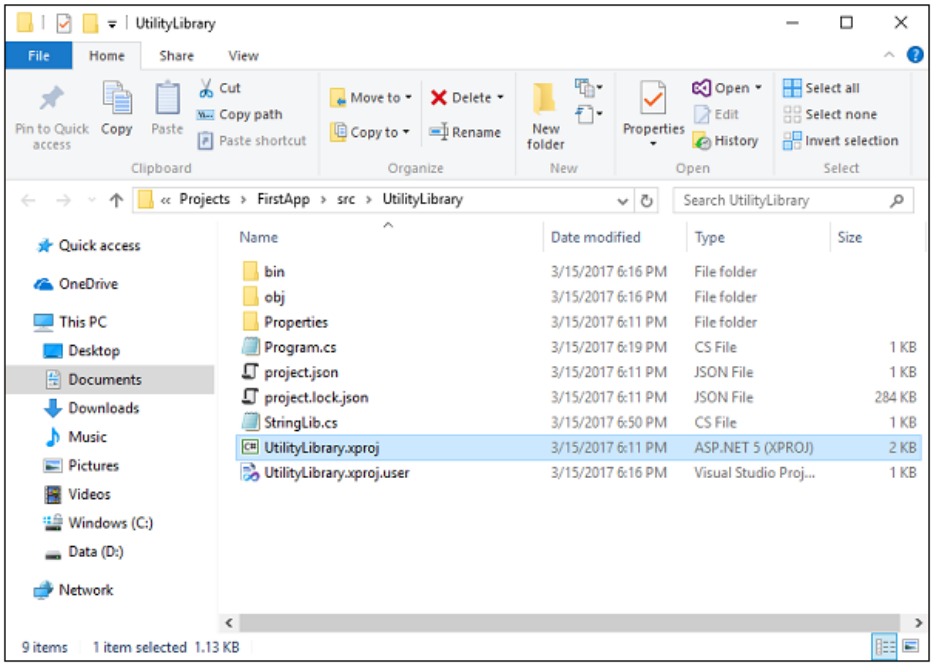
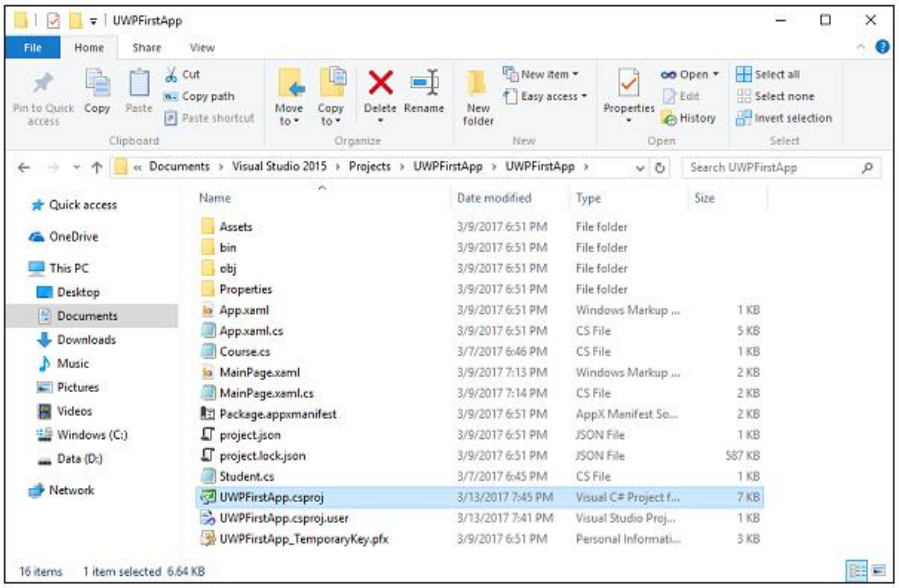
کتابخانه ی کلاسی پرتابل .Net Core

در این بخش کتابخانه ی کلاسی پرتابل (PCL) را توضیح می دهیم و می گوییم که چرا به آن نیاز داریم. جهت درک این مطلب پوشه ی پروژه ی کتابخانه ی کلاسی که در بخش قبل ایجاد کردیم را باز کنید.



در این پوشه علاوه بر project.json و فایل های CS می توانید فایل xproj.* را نیز ببینید، به این دلیل که نوع پروژه ی .NET Core ویژوال استودیو به جای csproj.* ، xproj.* است.

همان طور که توسط ماکروسافت بیان شده است، xproj.* در حال از بین رفتن است، اما با این حال همچنان در تجهیز پیش نمایش 2 وجود دارد. همان طور که قبلا نیز بیان کردیم، برنامه ی UWP از csproj.* استفاده می کند.



حقیقت امر این است که نمی توان csproj.* را به عنوان مرجع قرار داد و این قابلیت بنا نیست که اجرا شود، زیرا xproj.* در نهایت این عرصه را ترک خواهد کرد.

بنابراین در عوض ما به کتابخانه ی کلاسی ای نیاز داریم که بتوان آن را بین برنامه ی UWP و برنامه ی کنسول به اشتراک گذاشت. در همین نقطه است که PCL وارد می شود.

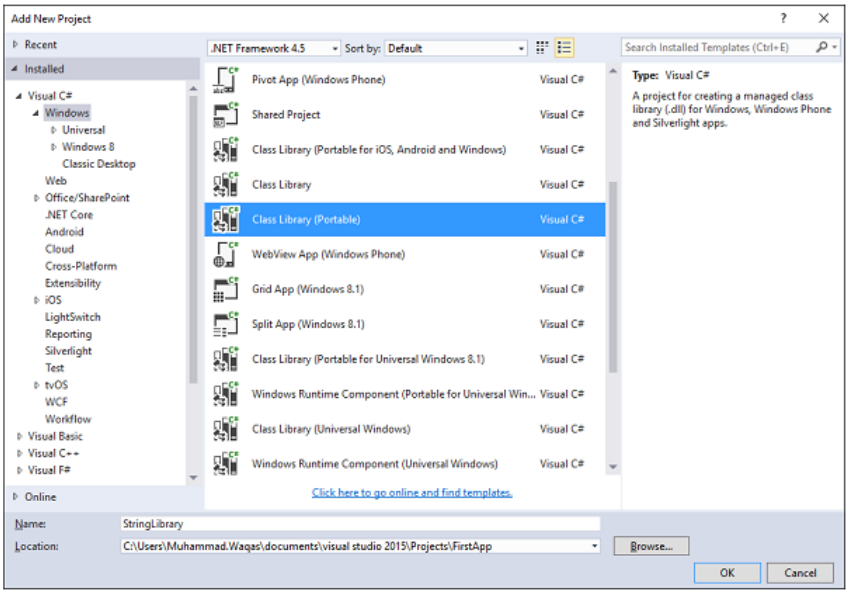
PCL چیست؟

اکنون به تعریف PCL می پردازیم:

- **با کمک پروژه ی PCL می توان اسمبلی های مدیریت شده ای که در بیش از یک پلتفرم .NET Framework کار می کنند را ساخت و نوشت.**
- **می توان کلاس هایی را ایجاد کرد که از کدی تشکیل شده باشند که بخواهیم آن ها را در میان چندین پروژه مانند shared business logic به اشتراک بگذاریم و پس از آن چندین نوع از پروژه های مختلف به این کلاس ها اشاره کنیم.**
- **با کمک PCL می توانید کتابخانه ها و برنامه های چند پلتفرمی را به صورت سریع و آسان برای پلتفرم های ماکروسافت بنویسید.**
- **با کمک PCL می توانید زمان و هزینه ی نوشتن و آزمایش کد را کاهش دهید.**

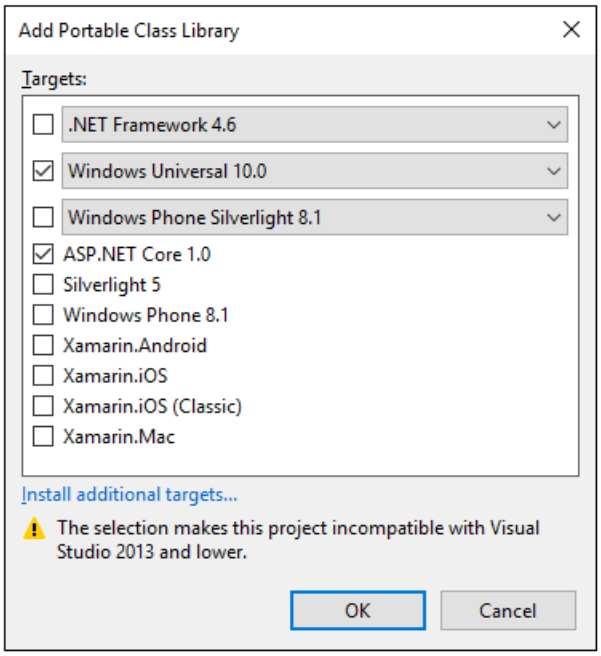
- برای نوشتن و ساخت اسمبلی های پرتابل .NET Framework. این نوع پروژه می تواند کمک زیادی به شما بکند. بعد از این کار تنها کافیست از طریق برنامه هایی که هدف آن ها چندین پلتفرم است (مانند ویندوز و ویندوز فون و غیره)، به این اسمبلی ها اشاره کنید.

حالا کتابخانه ی کلاسی که از طریق Solution Explorer ایجاد کردیم را پاک کنید. به صورت همزمان آن را از پوشه ی Solution حذف کنید و آیتم پروژه ی جدیدی را اضافه کنید.

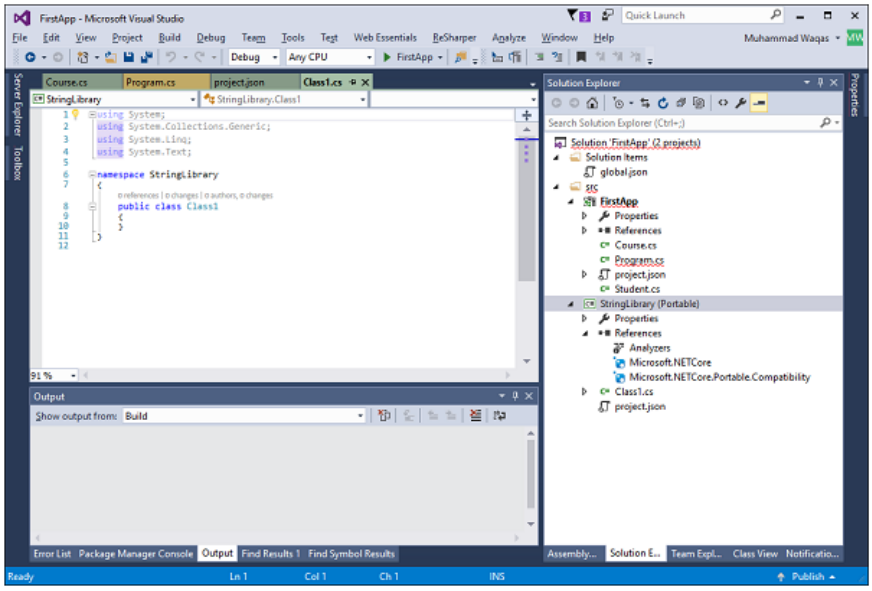


از بخش سمت چپ پنجره Windows → Visual C# را انتخاب کنید و از بخش میانی Class Library (Portable) را انتخاب کنید.

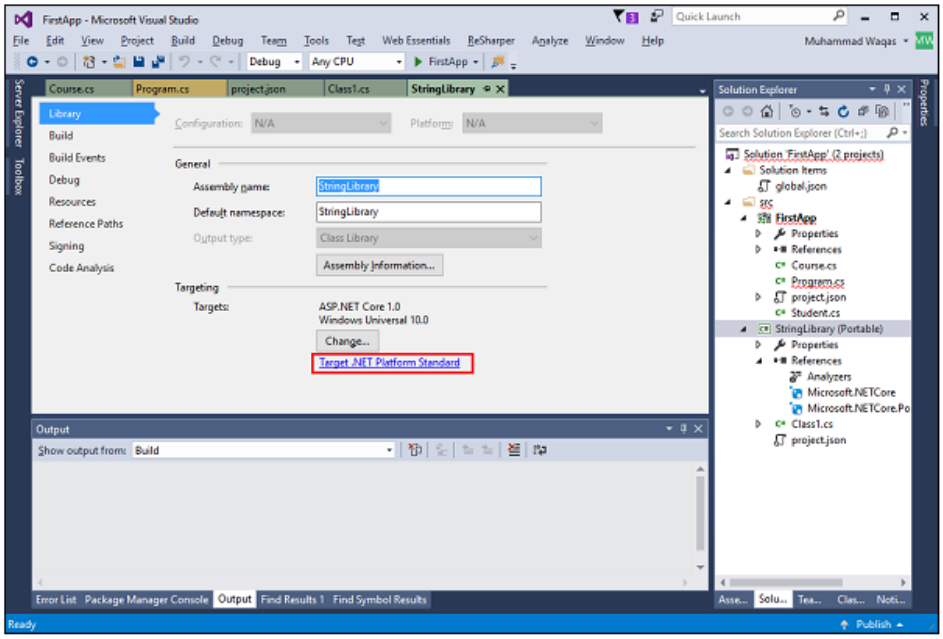
در بخش اسم StringLibrary را وارد کنید و بر روی OK کلیک کنید تا این پروژه ایجاد شود.



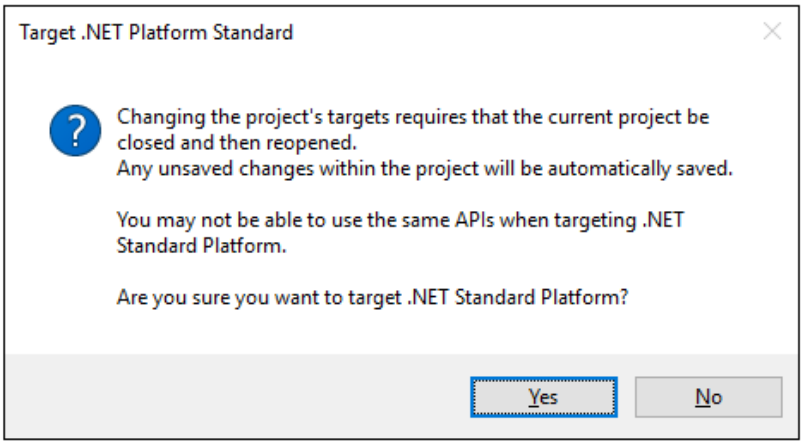
حالا باید فریمورک های هدفی که نیاز است به آن ها اشاره شود را انتخاب کنیم.فعلا بیایید برای مدت کوتاهی Universal و ASP.NET را انتخاب کنیم تا بعدا مجددا هدف را مشخص کنیم. بر روی OK کلیک کنید.



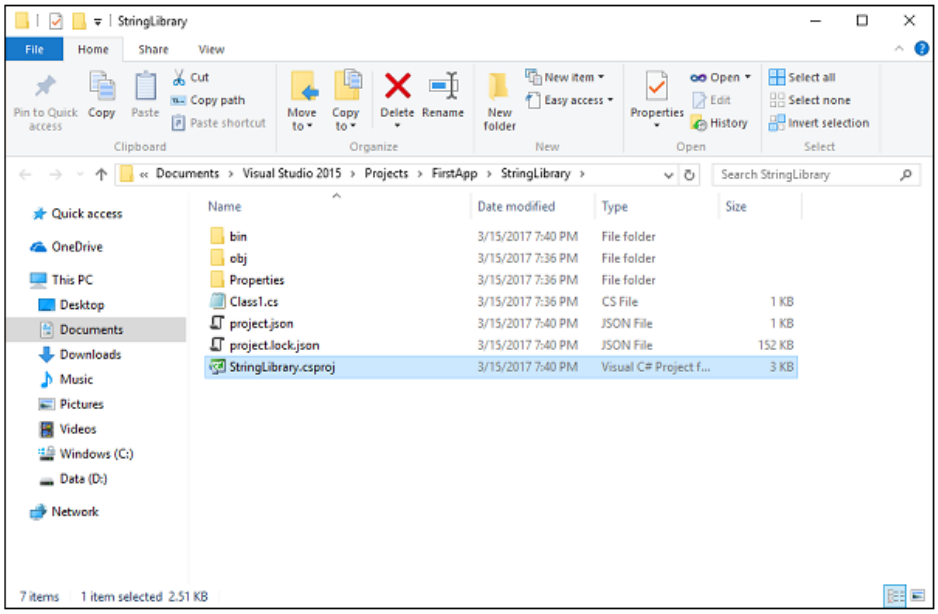
حالا همان طور که می بینید پروژه ی جدیدی به فرمت PCF ایجاد شده است. حالا در Solution Explorer بر روی پروژه ی StringLibrary کلیک کنید و Properties را انتخاب کنید.



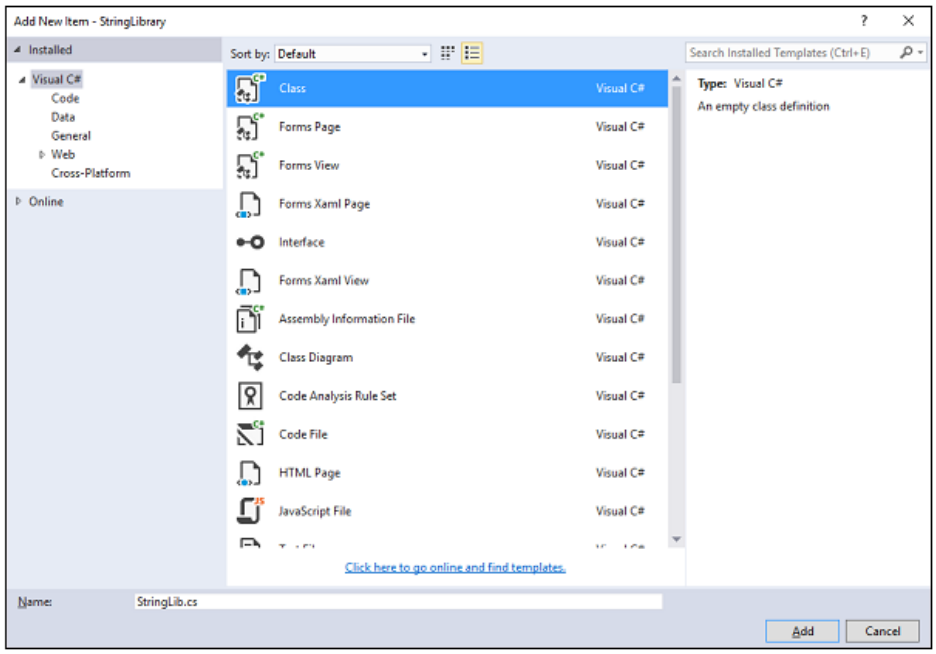
بر روی Target .NET Platform Standard کلیک کنید.



بر روی Yes کلیک کنید؛ با در نظر داشتن این مطلب که این کتابخانه نسبت به قبل تنها یک اختلاف اندک دارد. این تفاوت این است که از این کلاس می توان در کنار UWP نیز استفاده کرد. زیرا در آن به جای xproj.* از csproj.* استفاده شده است.



حالا کلاس جدیدی را اضافه کنید. برای انجام این کار از داخل Solution Explorer بر روی پروژه کلیک راست کنید و Add → Class... را انتخاب کنید.



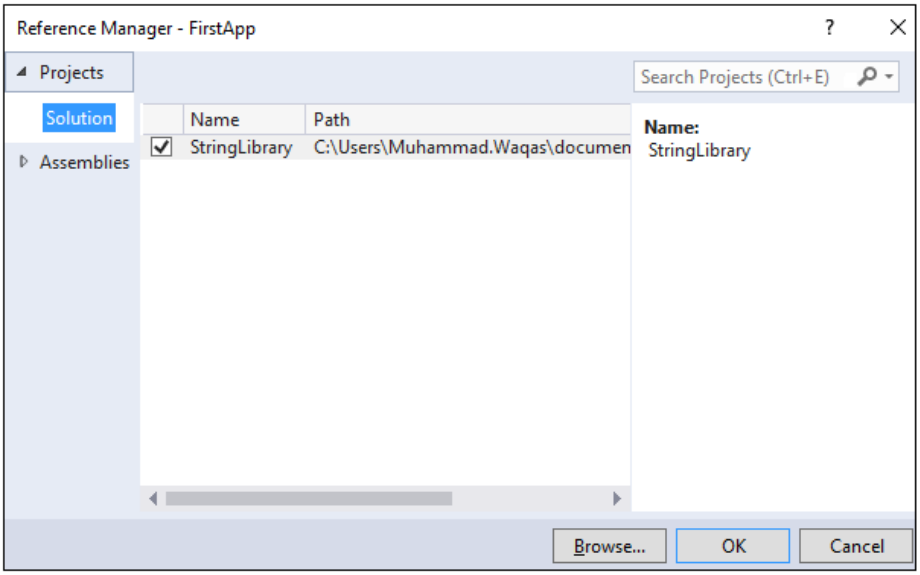
از بخش میانی پنجره class را انتخاب کنید و در قسمت اسم StringLib.cs را وارد کنید و در نهایت بر روی Add کلیک کنید. بعد از اضافه شدن این کلاس کد زیر را در فایل StringLib.cs جایگزین کنید.

```
using System;
```

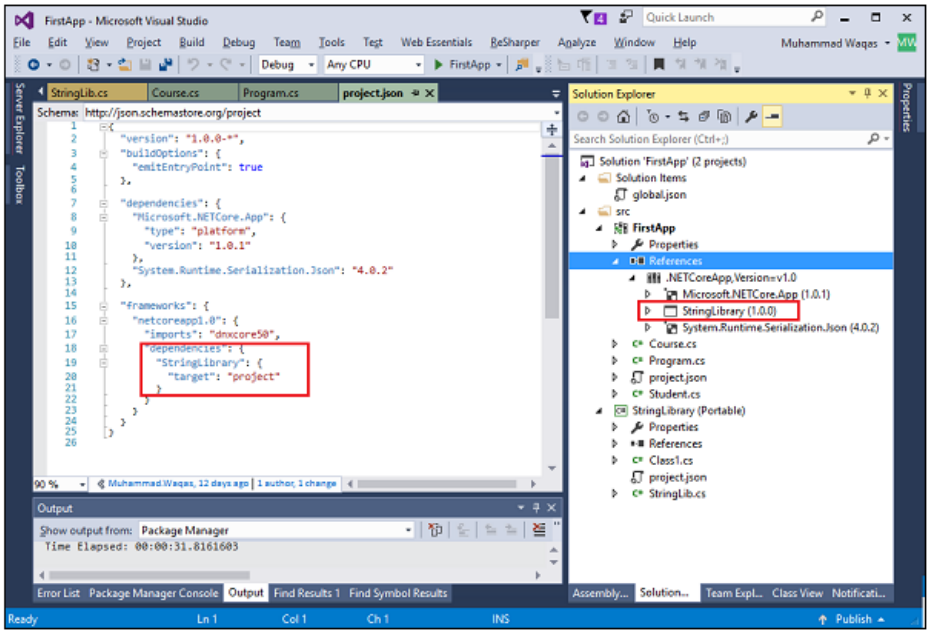
```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StringLibrary {
  public static class StringLib {
    public static bool StartsWithUpper(this String str) {
      if (String.IsNullOrWhiteSpace(str))
        return false;
      Char ch = str[0];
      return Char.IsUpper(ch);
    }
    public static bool StartsWithLower(this String str) {
      if (String.IsNullOrWhiteSpace(str))
        return false;
      Char ch = str[0];
      return Char.IsLower(ch);
    }
    public static bool StartsWithNumber(this String str) {
      if (String.IsNullOrWhiteSpace(str))
        return false;
      Char ch = str[0];
      return Char.IsNumber(ch);
    }
  }
}
```

حالا پروژه ی کتابخانه ی کلاس پرتابل را بسازید. بعد از انجام این کار این پروژه باید بدون هیچ مشکلی کامپایل شود. سپس مرجع این کتابخانه را در پروژه ی کنسول اضافه کنید. بنابراین FirstApp را گسترش دهید، بر روی References کلیک راست کنید و ...Add Reference را انتخاب کنید.

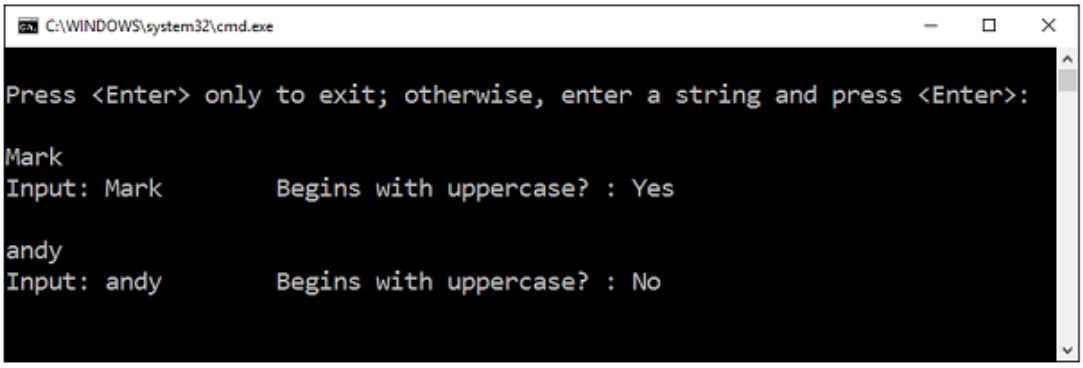


StringLibrary که همان پروژه ی کتابخانه ی کلاسی پرتابل ما است را از پنجره ی Reference Manager انتخاب کنید و بر روی OK کلیک کنید.



همان طور که می بینید، مرجع StringLibrary به پروژه ی کنسول اضافه شده است که می توان آن را در فایل project.json نیز مشاهده کرد.

حالا مجددا برنامه را اجرا کنید تا همان نتیجه ی قبلی نمایش داده شود.



حالا بیایید در پروژه ی خود از متدهای افزونه ای دیگری از کتابخانه ی کلاسی پرتابل استفاده کنیم. همین کتابخانه در برنامه ی UWP شما نیز استفاده خواهد شد.