Unit test

ساخت نرم افزار مورد انتظار کاربران و خروجی با کیفیت تری تحویل بدهیم.

تست تمیز سبب ایجاد یک داکیومنت کامل از نرم افزار خود می شود.

حالت های مختلف متد خود را بررسی می کنیم.

اگر کدی را تغییر بدهیم و جای دیگه را تحت تاثیر قرار دهد متوجه می شویم و اصلاح می کنیم.

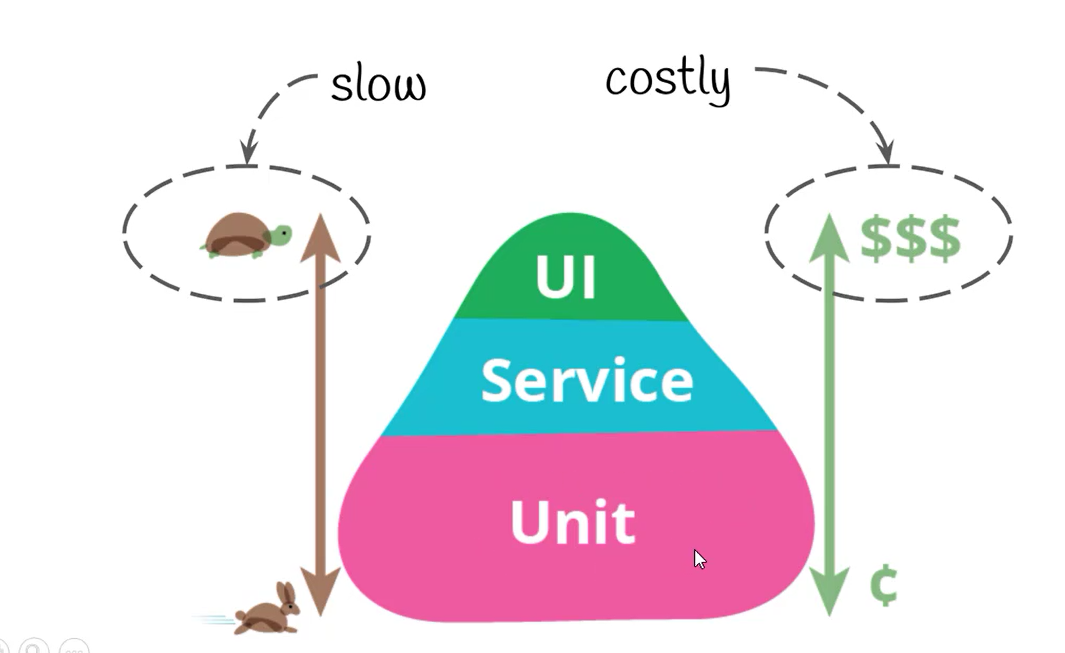
نتیجه تست را سریع و راحت به ما نمایش می دهد.

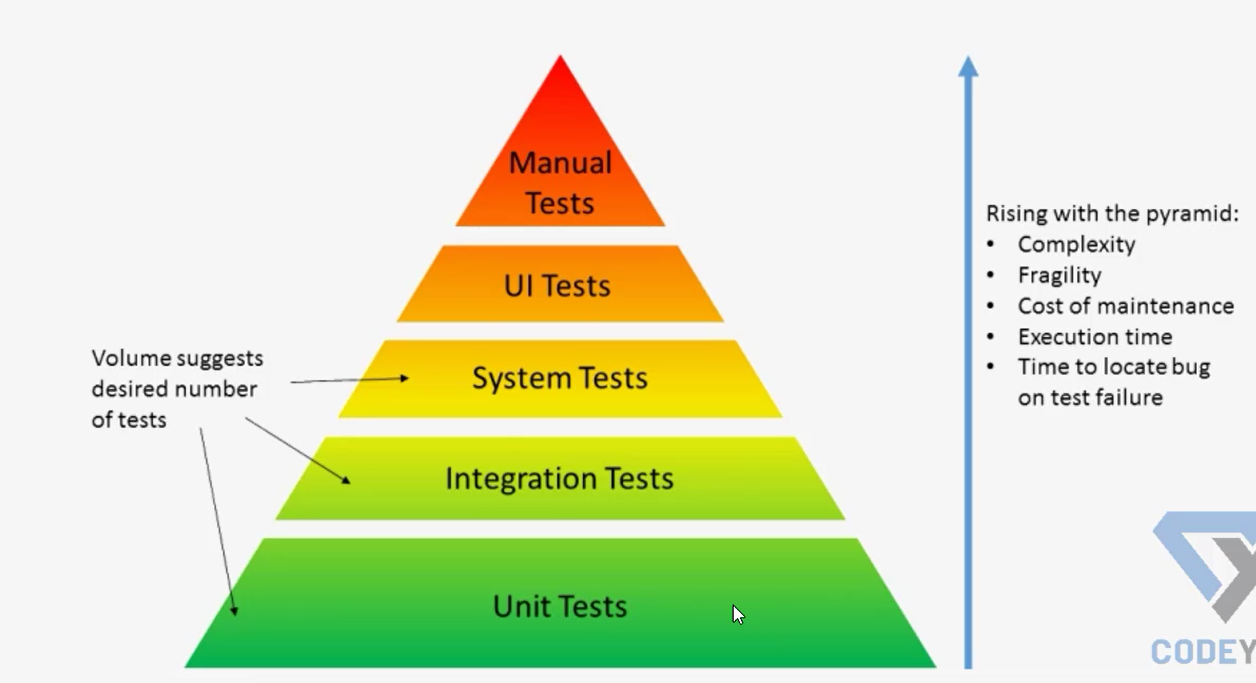
زمان و هزینه را در دراز مدت کاهش می دهد.

unitTest یک واحد کوچک از نرم افزار را تست می کند.

Integrated test مثلا برای زمانی که یک سرویسی تست شود و کار دیتابیسی نیاز است و دیتا فچ شود.

Ui تست نیز از endpoint شروع می کند و تا انتهای عملیات را تست می کند





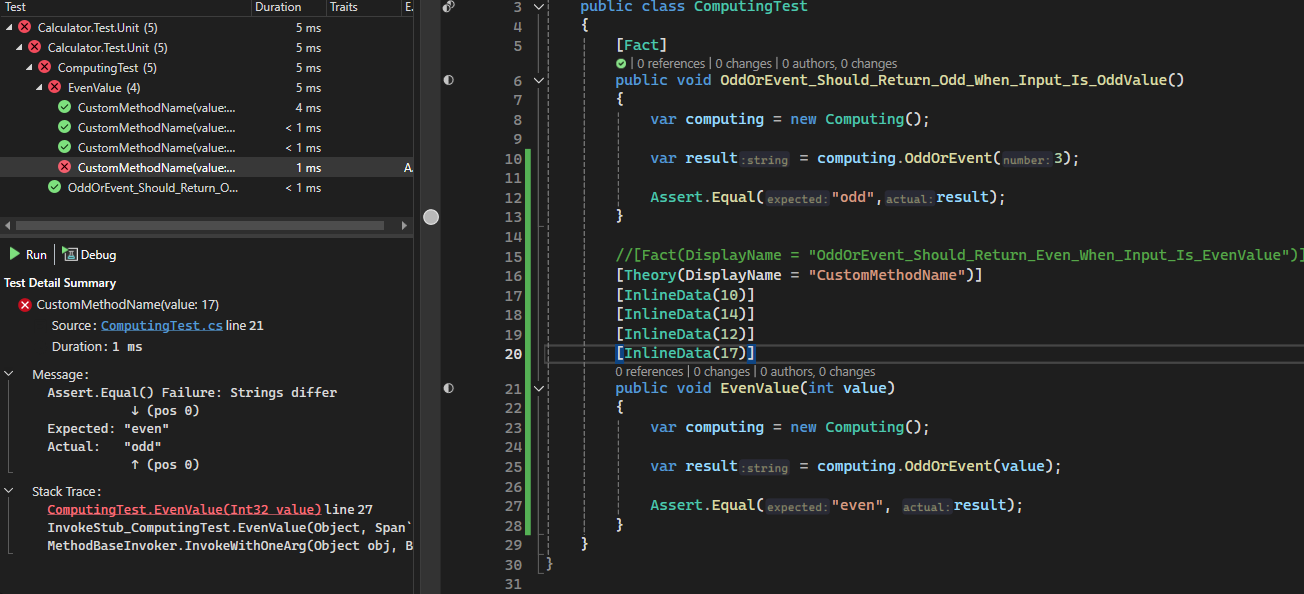
بیشتر تست ها از نوع unit است و هر جه بالا برویم هزینه و پیچیدگی اضافه می شود.

فریم ورک برای تست نوشتن در سی شارپ xunit است و در قدیم nunit بوده است. و اجرای تست ها توسط vs انجام می شود. Visual studio test runner

نکته : اسم متد در xunit باید بیان گر اسم متدی باشد که برای آن تست می خواهیم بنویسیم.

نکته : زمانی که تست خود را کامل کردیم می توانیم تنیجه ای که انتظار داریم برای پاس شدن تست را بدهیم و اگر درست باشد تست pass می شود

Assert.Equal(“Odd”,result);



Ncrunch

یک testRunner به صورت live است که بعد از بروز تغییرات به صورت اتوماتیک تست را انجام می دهد.

قسمت های از کد که بیشتر از حد معمول زمان میبرد را علامت زرد می زند.

Live unit testing

کلیک راست در پروژه و انتخاب گزینه Live Unit testing و آن را پلی کرده. با هر بار ذخیره کردن تست را انجام می دهد مجدد با هر تغییر.

[Fact]

برای تست های ساده و ثابت کاربرد دارد که هیچ پارامتری ندارد و به یک صورت اجرا می شودparameter less

[Theory]

برای تعریف تست های که با داده های مختلف سرکار دارد استفاده می شود و زمانی که متد شما parameter ورودی دارد و اگر داده تعریف نکنید خطا می دهد

Fluent Assertion

فریم ورکی برای افزایش خوانایی کد به جای استفاده از Assert.Equal می توانیم از متد های این کتابخانه استفاده کنیم به طور مثال

Result.Should().Be();

Should.Equal();

Should.BeTypeOf();

**نکته** برای تست Exception

var result = new Action(() =>

{

computing.CalculateAge(0, 2025);

});