

# ***Pomiar jakości kodu, testy jednostkowe, TDD i BDD***

## **1. Pomiar jakości kodu**

### **Co warto testować:**

- a) czy kod działa poprawnie w normalnych przypadkach
- b) wydajność kodu niezależnie od wielkości danych
- c) czytelność kodu, tj. czy nazwy zmiennych/metod/klas są dobrze opisane oraz czy kod zawiera niezbędne komentarze
- d) podzielność kodu, łatwiej modyfikować małą część kodu
- e) jak często kod się powtarza
- f) bezpieczeństwo
- g) ilość oprogramowania pokrytego testami
- h) dokumentacja oprogramowania
- i) podatność kodu na błędy
- j) ilość nieużywanego kodu
- k) i zapewne wiele innych

### **Zalety mierzenia jakości kodu:**

- a) mamy świadomość jakości kodu, a jego dobra jakość ułatwia wprowadzanie modyfikacji oraz dodawanie nowych funkcjonalności
- b) dobra jakość kodu powoduje, że kod jest zrozumiały dla wszystkich
- c) łatwiej i szybciej można nad nim pracować

### **Przykładowe narzędzia do oceny jakości kodu:**

- a) Quality by Code Climate

Quality stworzone przez firmę Code Climate pozwala na dokonanie statycznej analizy kodu (tzn. bez uruchamiania programu), co pozwala na wykrycie powtórzeń oraz identyfikację skomplikowanych metod oraz klas. Quality sprawdza również pokrycie kodu testami.

Obsługiwane języki programowania:

JavaScript

PHP

Python

Ruby

Java

TypeScript

Go

Swift

Scala

Kotlin



Cennik:

#### OPEN SOURCE

\$0

Free forever

- Unlimited public repositories
- Unlimited users
- GitHub pull request integration

Sign up with GitHub

#### STARTUP

\$0

Up to 4 seats

- Unlimited private repositories
- Everything in the Team plan for up to 4 seats

Sign up with GitHub

#### TEAM

\$16.67/mo

per seat per month  
billed annually

\$20 billed monthly

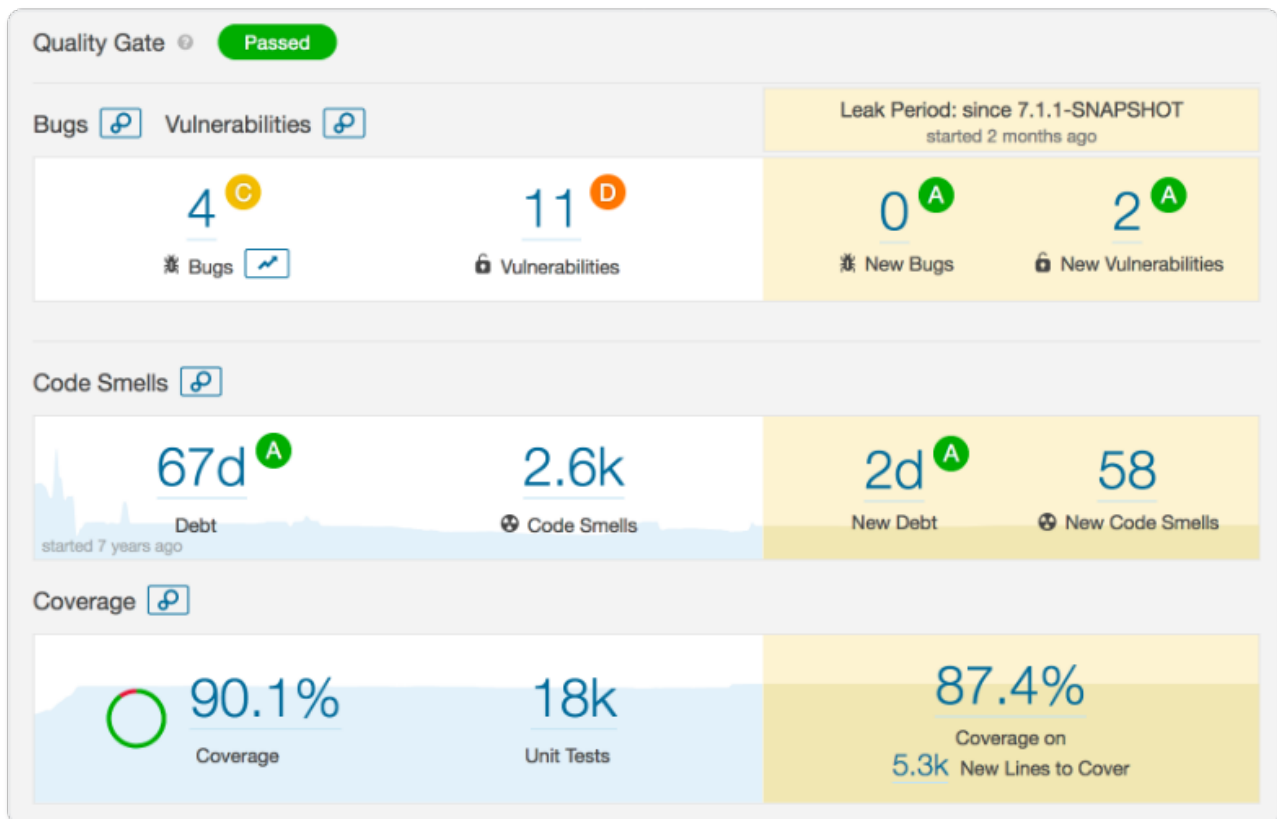
- Unlimited private repositories
- Priority Support
- Discounts for teams of 25+

Sign up with GitHub

Includes 14-day free trial

## b) SonarQube wcześniej Sonar

Stała inspekcja – SonarQube umożliwia nie tylko pokazanie kondycji aplikacji, ale także podkreślenie nowo wprowadzonych problemów. Dzięki Quality Gate możesz naprawić wyciek, a tym samym systematycznie poprawiać jakość kodu.



Wykryj podstępne problemy - Analizatory kodu są wyposażone w silnie wrażliwe silniki przepływu danych, które wykrywają trudne problemy, takie jak dereferencje wskaźników zerowych, błędy logiczne, wycieki zasobów





```
139 public static String readFromPreferences(Context context, String preferenceName, String defaultValue) {
140     1 SharedPreferences sharedPreferences = null;
141     try {
142         sharedPreferences = context.getSharedPreferences(PREF_FILE_NAME, Context.MODE_PRIVATE);
143     } catch (Exception e) {
144         e.printStackTrace();
145         new ExceptionHandler(e, "NavigationDrawerFragment - readFromPreferences()");
146     }
147     return 2 sharedPreferences.getString(preferenceName, defaultValue);
148 }
```

A "NullPointerException" could be thrown; "sharedPreferences" is nullable here. ... 11 months ago ▾ L147 🔗

🐛 Bug 🚨 Major 🔵 Open ▾ Not assigned ▾ 10min effort 💬 Comment 🔗 cert, cws ▾

## Cennik

### Choose your plan

 <b>Community Edition</b> Used and loved by 100,000+ companies <ul style="list-style-type: none"><li>✓ SonarQube &amp; 60+ plugins</li><li>✓ SonarLint</li><li>✓ 15 languages ⓘ</li></ul> <hr/> <b>Free &amp; Open Source</b> <a href="#">SEE MORE</a>	 <b>Developer Edition</b> Built for developers by developers <i>Community Edition plus:</i> <ul style="list-style-type: none"><li>✓ Branch Analysis</li><li>✓ Pull Request Analysis</li><li>✓ Detection of injection flaws</li><li>✓ SonarLint notifications</li><li>✓ 21 languages ⓘ</li></ul> <hr/> <b>Starts at €120</b> <a href="#">SEE MORE</a>	 <b>Enterprise Edition</b> Designed for the Enterprise that values Quality <i>Developer Edition plus:</i> <ul style="list-style-type: none"><li>✓ Portfolio management</li><li>✓ Executive reporting</li><li>✓ Project transfer</li><li>✓ 26 languages ⓘ</li></ul> <hr/> <b>Starts at €15,000</b> <a href="#">SEE MORE</a>	 <b>Data Center Edition</b> Designed for High Availability <i>Enterprise Edition plus:</i> <ul style="list-style-type: none"><li>✓ Component redundancy</li><li>✓ Data integrity</li></ul> <hr/> <b>Starts at €100,000</b> <a href="#">SEE MORE</a>
---	---	---	--

### c) Codacy

<https://www.youtube.com/watch?v=oxqTu2ouxaw>

Codacy zawiera wsparcie dla statycznej analizy, duplikacji kodu, złożności kodu, pokrycia testami

Codacy obsługuje 28 języków programowania

Codacy można zintegrować z wieloma narzędziami: GitHub, Slack, Jira, Git, YouTrack, Bitbucket, Hipchat



## Startup

For teams up to 4 users <sup>?</sup>

\$0

\$0 / Forever

[Get started](#)

✓ Everything in the Pro plan

### Open source

Free for all Open source teams. Forever.

## Pro

Most popular

For growing teams to ship well and on time

\$15

per user / month

[Try for free](#)

30-day trial

- ✓ Private and public repositories
- ✓ Higher priority and faster analysis
- ✓ Priority support

## Enterprise

First-class security on your servers

Custom

Tailored to your business

[Request a demo](#)

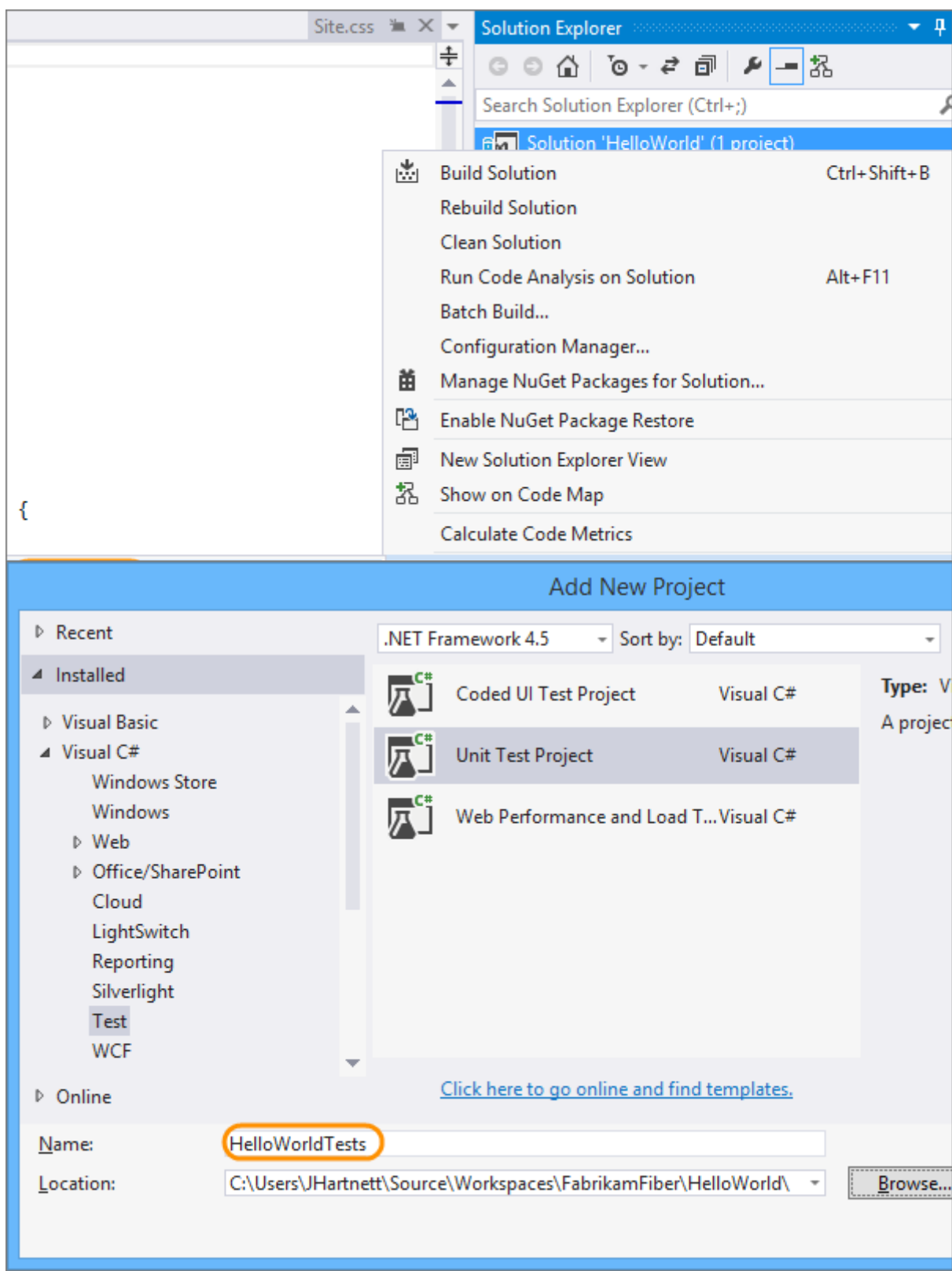
- ✓ Feature updates included
- ✓ For Github enterprise, Bitbucket server and Gitlab
- ✓ Extended security analysis

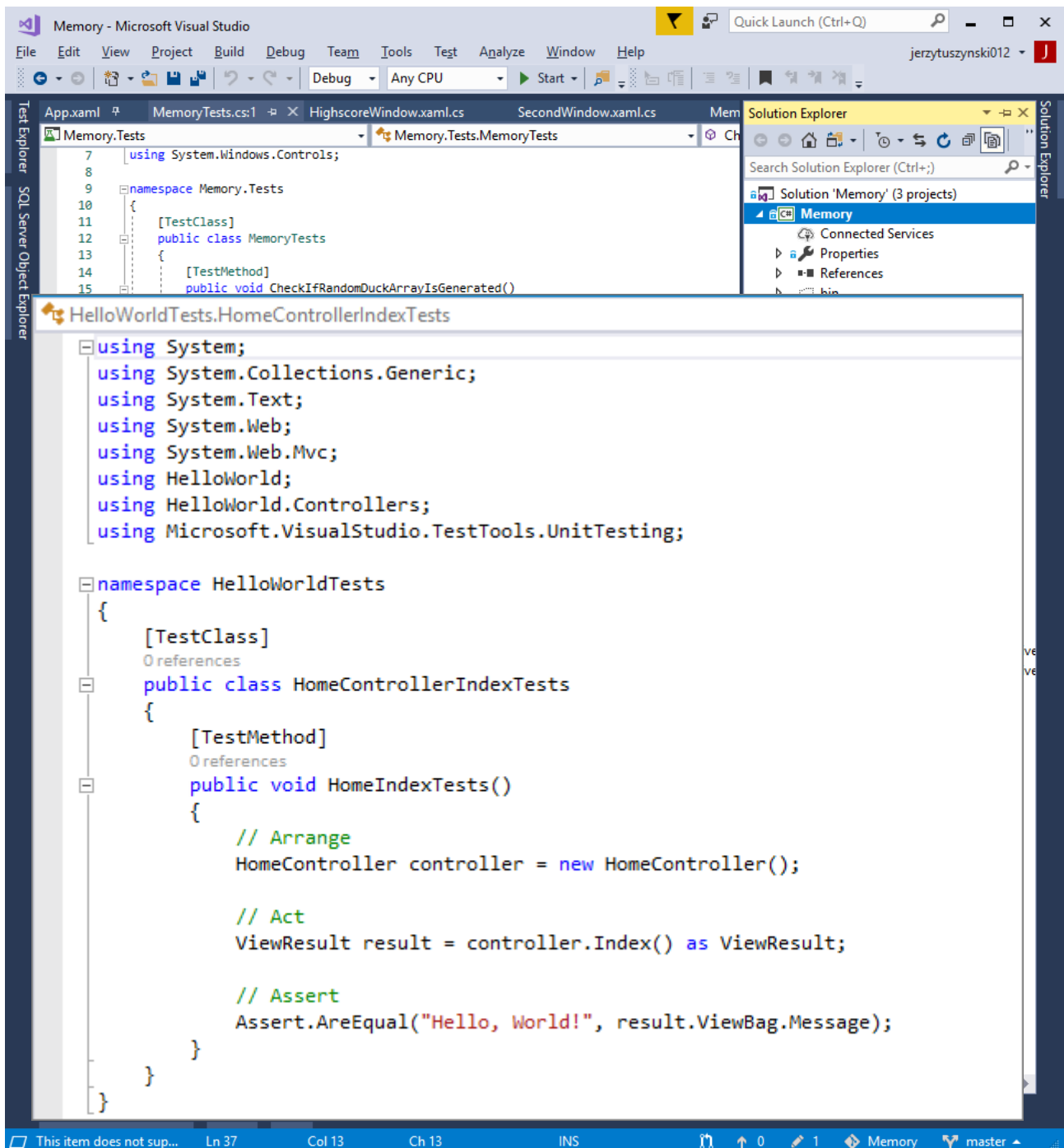
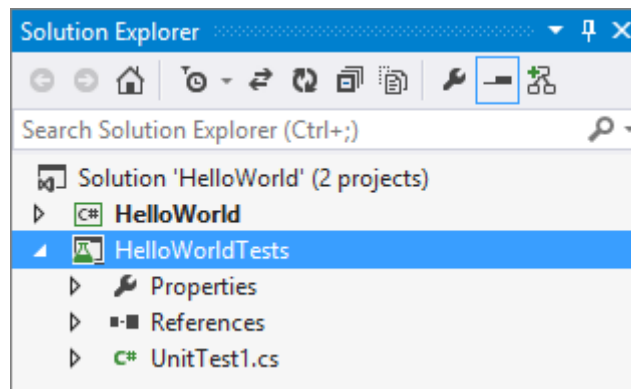
VAT not included

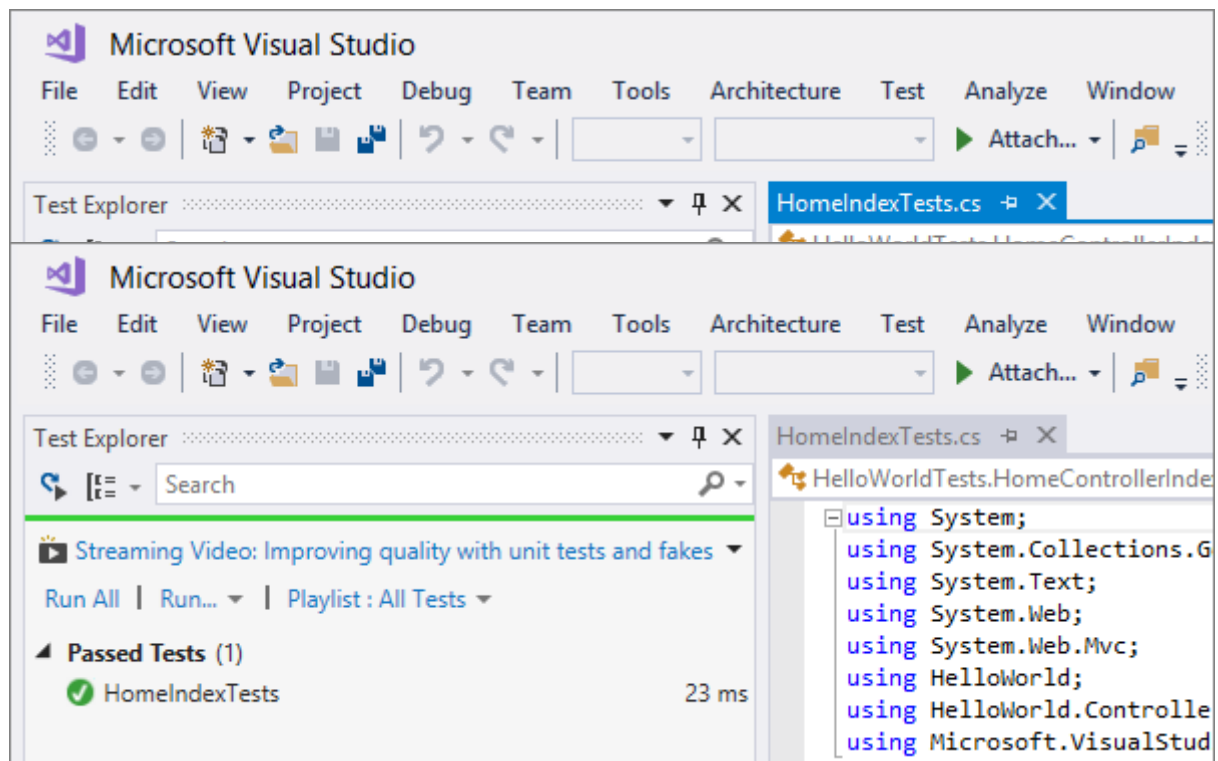
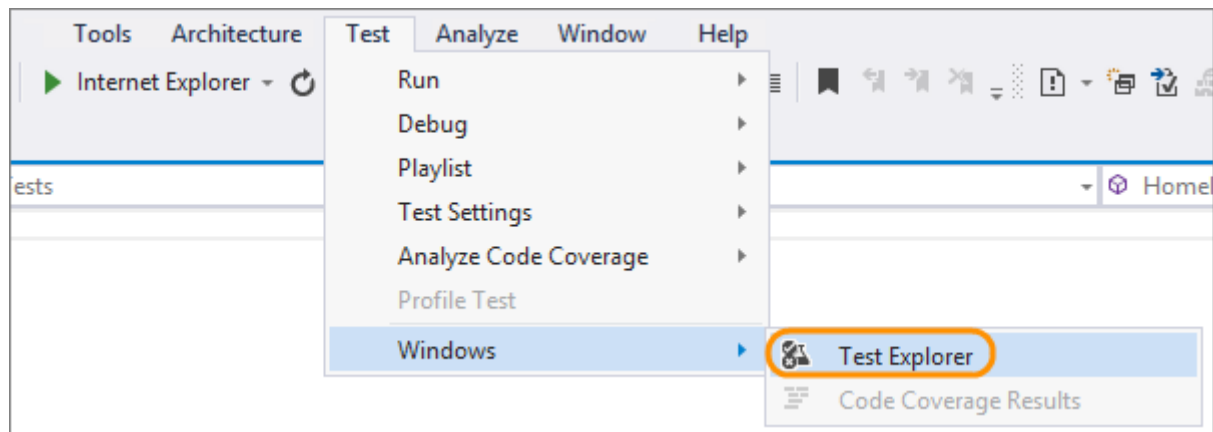
## 2. Testy jednostkowe (ang. unit tests)

Wikipedia: Test jednostkowy – metoda testowania tworzonego oprogramowania poprzez wykonywanie testów weryfikujących poprawność działania pojedynczych elementów (jednostek) programu – np. metod lub obiektów w programowaniu obiektowym lub procedur w programowaniu proceduralnym. Testowany fragment programu poddawany jest testowi, który wykonuje go i porównuje wynik (np. zwrócone wartości, stan obiektu, zgłoszone wyjątki) z oczekiwanymi wynikami.

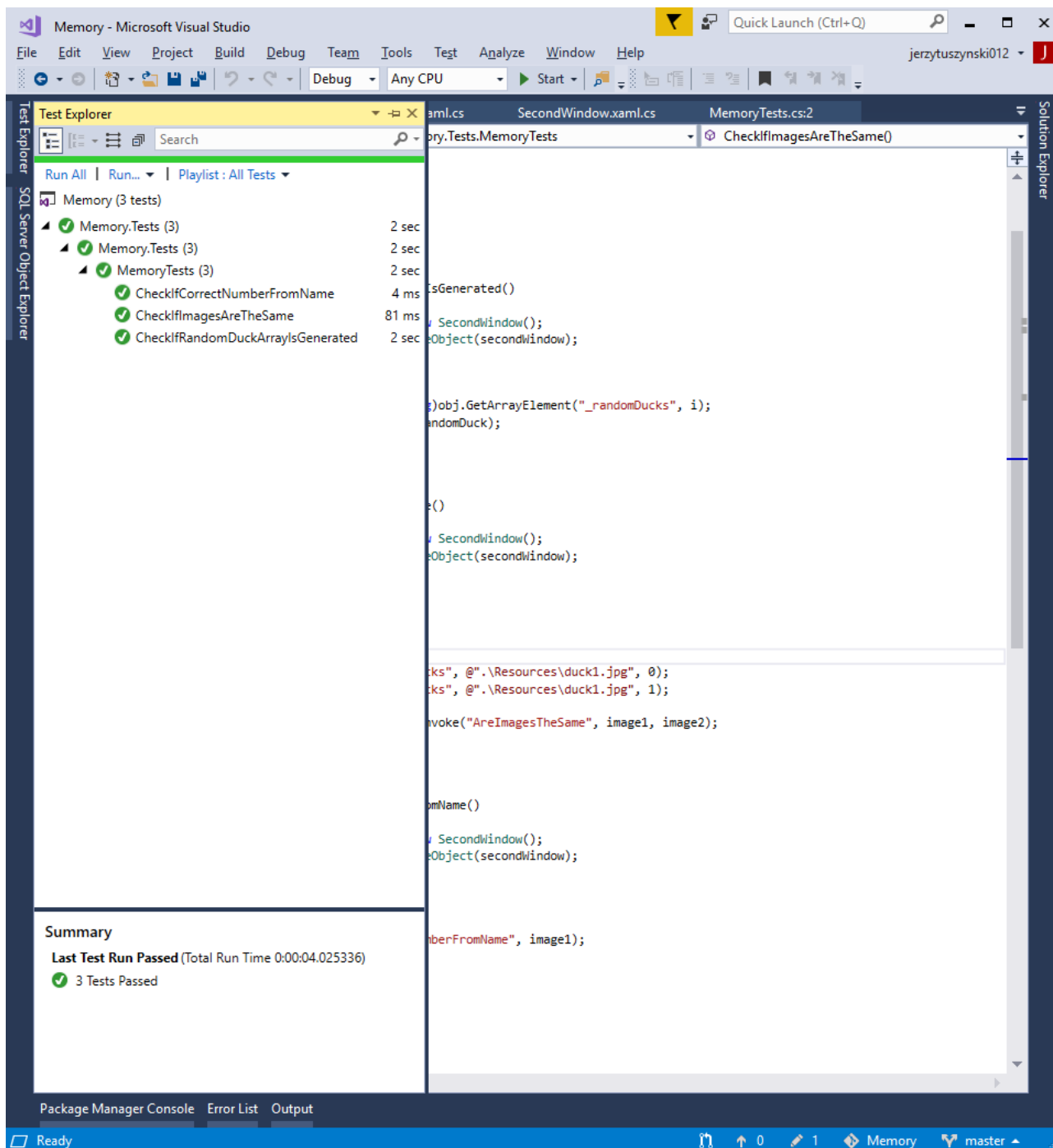
Testy są jednostkowe są nam potrzebne przy rozwijaniu aplikacji. Gdy mamy napisany test i zmienimy daną funkcjonalność, dzięki testowi możemy sprawdzić, czy zmiana została napisana poprawnie.











### 3. TDD – czyli Test Driven Development

Test-driven development (TDD) – technika tworzenia oprogramowania, zaliczana do metodyk zwinnych.

Polega na wielokrotnym powtarzaniu kilku kroków:

1. Stworzenie testu jednostkowego, który powinien być możliwie najprostszy, aby uniknąć możliwości popełnienia błędu w samym teście. Test ma sprawdzać funkcjonalność, która będzie implementowana w kroku 2.

2. Implementacja funkcjonalności – tworzymy funkcjonalność, którą chcemy zaimplementować. Funkcjonalność ta powinna spełniać założenia testu jednostkowego, a wykonanie testu jednostkowego powinno kończyć się sukcesem.

3. Refaktoryzacja, czyli porządki w stworzonej funkcjonalności. Ma to na celu uporządkowanie kodu, tak aby spełnione były standardy. Czynności wykonywane w tym kroku nie mogą zmienić wyniku testów.

### Wady Test Driven Development

Metodologia TDD posiada jednak dwie główne wady – wymaga dodatkowego czasu na stworzenie testów jednostkowych oraz wymaga czasu na utrzymanie testów.

### Zalety Test Driven Development

Podstawową zaletą TDD jest szybkie wychwytywanie błędów. Już w momencie tworzenia oprogramowania wiele błędów można wychwycić za pomocą tej metodologii. Nie jest to bez znaczenia, ponieważ każdy błąd z czasem „nabiera wartości” – im później błąd zostanie wykryty, tym więcej osób angażuje i tym więcej kosztuje zarówno w aspekcie czasu, jak i pieniędzy.

Kolejną zaletą TDD jest bardziej przemyślany kod. Ponieważ w pierwszym kroku tworzony jest kod, który będzie wykorzystywał tworzoną funkcjonalność (test), to zazwyczaj naturalnie ten kod jest bardzo prosty i elegancki.

Największą chyba zaletą tej metodologii pracy jest możliwość przetestowania funkcjonalności bez uruchamiania całego oprogramowania – tworzony kod można uruchamiać częściowo, bez uruchamiania całej aplikacji.

## 4. BDD – czyli Behavior Driven Development

Czym jest BDD? Behavior Driven Development jest procesem wytwarzania oprogramowania w oparciu o konkretną strukturę formułowania wymagań. Cała aplikacja budowana jest z komponentów, małymi historyjkami które opowiadają o tym jak powinien zachować się program gdy zajdą pewne okoliczności – z ang. stories. Każde story budowany jest na schemacie given,when,then :

given – określa warunki początkowe, przedstawia aktora, zapoznaje odbiorcę z kontekstem historyjki,

when – opisuje akcje, występujące zdarzenie,

then – informuje o oczekiwanych rezultatach.

```
Story: Account Holder withdraws cash
-

As an Account Holder
I want to withdraw cash from an ATM
So that I can get money when the bank is closed
-

Scenario 1: Account has sufficient funds
Given the account balance is \$100
  And the card is valid
  And the machine contains enough money
When the Account Holder requests \$20
Then the ATM should dispense \$20
  And the account balance should be \$80
  And the card should be returned
-

Scenario 2: Account has insufficient funds
Given the account balance is \$10
  And the card is valid
  And the machine contains enough money
When the Account Holder requests \$20
Then the ATM should not dispense any money
  And the ATM should say there are insufficient funds
  And the account balance should be \$20
  And the card should be returned
-

Scenario 3: Card has been disabled
Given the card is disabled
When the Account Holder requests \$20
Then the ATM should retain the card
  And the ATM should say the card has been retained
```