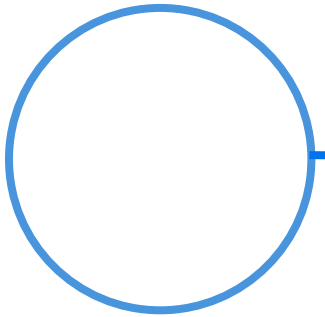
Szoftvertesztelés | 1-2. gyakorlat

Elméleti alapok, unit tesztelés, mockolás

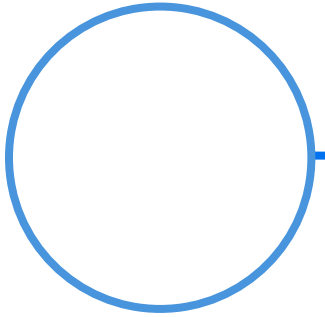
11 September 2025

Bemutatkozás
Előadók



Höltzl Tamás

Tesztmenedzser



Masa Tibor

Teszt architect

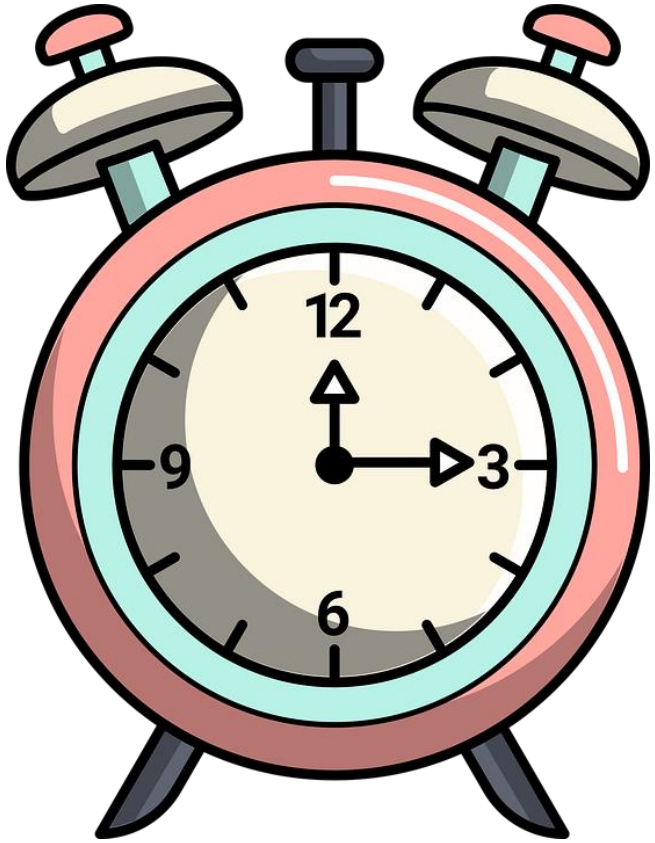
Bemutatkozás

ZDI – Carl Zeiss Digitial Innovation Hungary

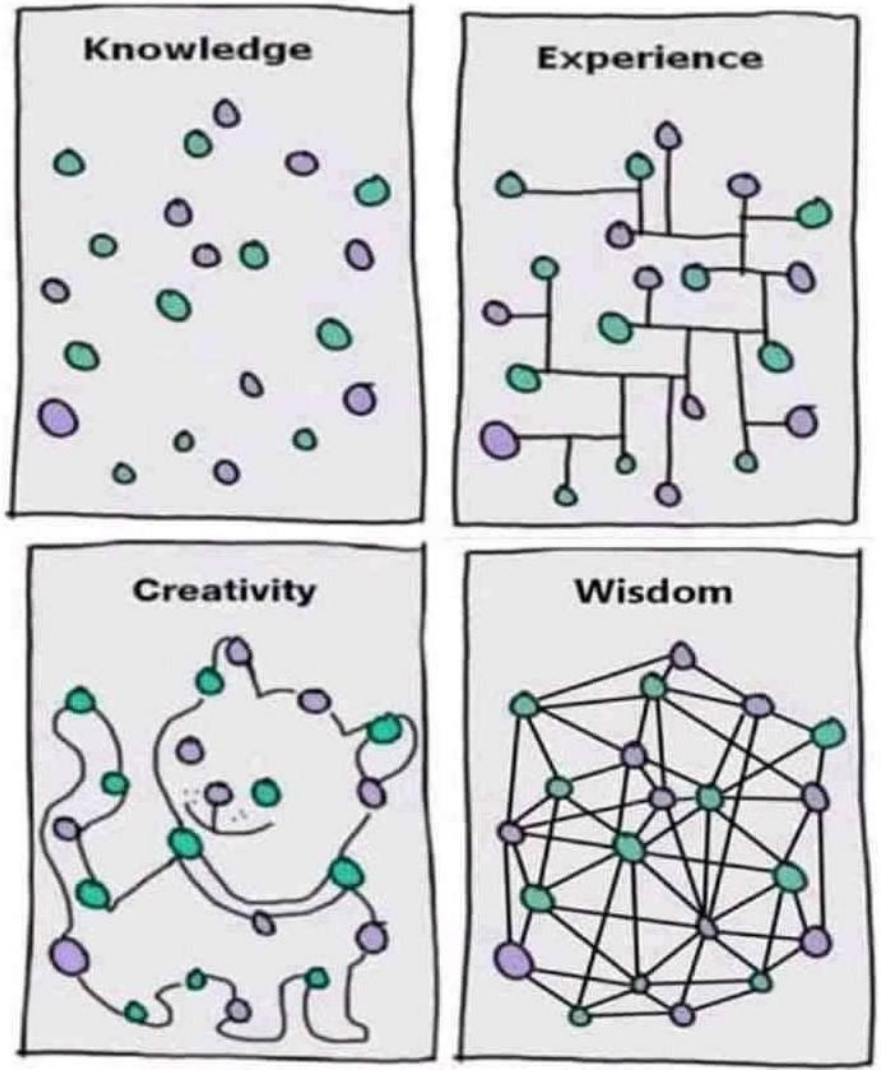
Mielőtt elkezdjük…
Óra menete, kérdések

Menetrend

- 10:00 – 11:30
  - Elméleti alapok
  - Környezet beállítása
- 11:30 – 12:15
  - Ebédszünet
- 12:15 – 13:45
  - Unit tesztelés



Bevezető
Milyen céllal jöttünk ma ide?



Agenda

01 Elméleti alapok

02 Környezet beállítása

03 Unit tesztelés

Definíció

Mi az a teszt automatizálás?

Speciális szoftverek használata

- tesztesetek futtatására
- az elvárt és aktuális eredmények összehasonlítására
- bármilyen egyéb, teszteléssel összefüggő tevékenység automatizálására

Tesztautomatizálás
Kulcsszavak

Szoftver
- Selenium, pytest, JUnit, saját keretrendszer

Teszteset-futtatás
- A manuális munka kiváltására

Eredmények összehasonlítása
- Elvárt eredmény == aktuális eredmény?

Scope
- Funkcionális és nem-funkcionális tesztek

Automatikus végrehajtás
- Elsősorban az ismétlődő, hibára hajlamos tevékenységek kiváltására

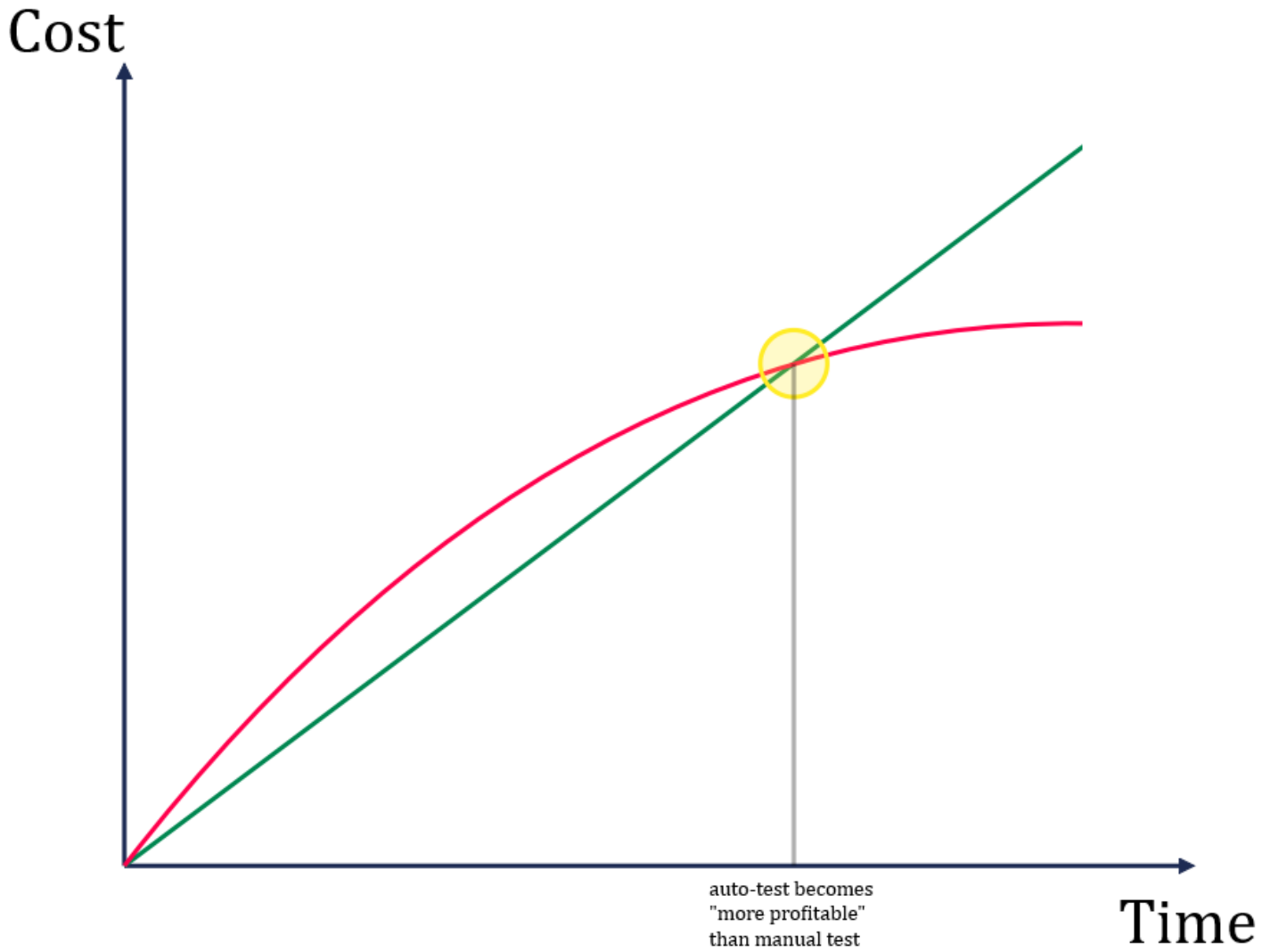
Előnyök
„The good parts”

Hatékonyság-növelés	■ Gyorsabb, mint a manuális tesztelés ■ Ideális ismétlődő feladatokra (pld. regressziós tesztelés
Pontosság	■ Emberi hiba csökkentése ■ Az előre definiált lépéseket követi
Lefedettség	■ Több teszteset és tesztadat adott időn belül ■ Különböző konfigurációk lefedése
Gyorsabb visszajelzés	■ Azonnali visszajelzés a futtatás végén
Újrahasznosíthatóság	■ Teszt scriptek újrafelhasználása (különböző adatok, folyamatok során)
24/7 elérhetőség	■ Éjszakai tesztfuttatás -> reggeli kiértékelés

Hátrányok
„The bad parts”

Magas kezdeti költség	■ Idő, energia, pénz a rendszer beállítása, elkészítése ■ Kollégák betanítása
Karbantartás	■ Szkriptek folyamatosan frissítendőek ■ Törékeny szkriptek
Nem váltja ki a manuális tesztelést	■ Emberi „megérzés” nem írható felül -> felderítő tesztelés ■ Nem tudja tesztelni azt, amit nem írtunk meg neki
Eszközök kiválasztása	■ Külön kihívás, technikai és emberismeret is kell hozzá ■ Tipikus expert & lead feladat
Függőségek	■ Tesztkörnyezet, tesztadatok ■ Mindennek a helyén kell lennie

Manuális Vs automatizált tesztelés



Test Automation Engineer szerepkör

- Tesztszkriptek írása (QA)
- Eredmények kiértékelése (QA)
- CI/CD integráció és tesztkörnyezet beállítása (DevOps)
- Keretrendszer-fejlesztés és karbantartás (DEV, Architect)
- Együttműködés a csapaton belül
-
- → Hard és soft skillek megfelelő kombinációja szükséges
-

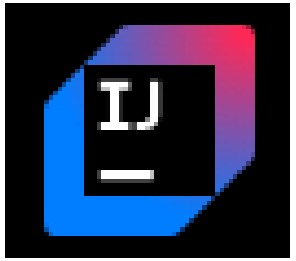
Agenda

01 Elméleti alapok

02 Környezet beállítása

03 Unit tesztelés

Előkészületek 1.
Mire lesz szükségünk?



Előkészületek 2.
Környezet ellenőrzése

1. Windows command line: mvn -version

```
C:\Users>mvn -version
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: C:\Program Files\Apache\apache-maven-3.8.4
Java version: 17.0.11, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: hu_HU, platform encoding: Cp1250
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users>
```

2. Windows command line: git -v

```
C:\Users>git -v
git version 2.45.1.windows.1

C:\Users>
```

Előkészületek 3.
Saját repository használata

Projekt felmásolása saját repository-ba

- git init
- git add --all
- git commit -m „Init project"
- git branch -M main
- git remote add origin https://github.com/username/repository.git
- git push -u origin main

Lásd még: Adding locally hosted code to GitHub - GitHub Docs

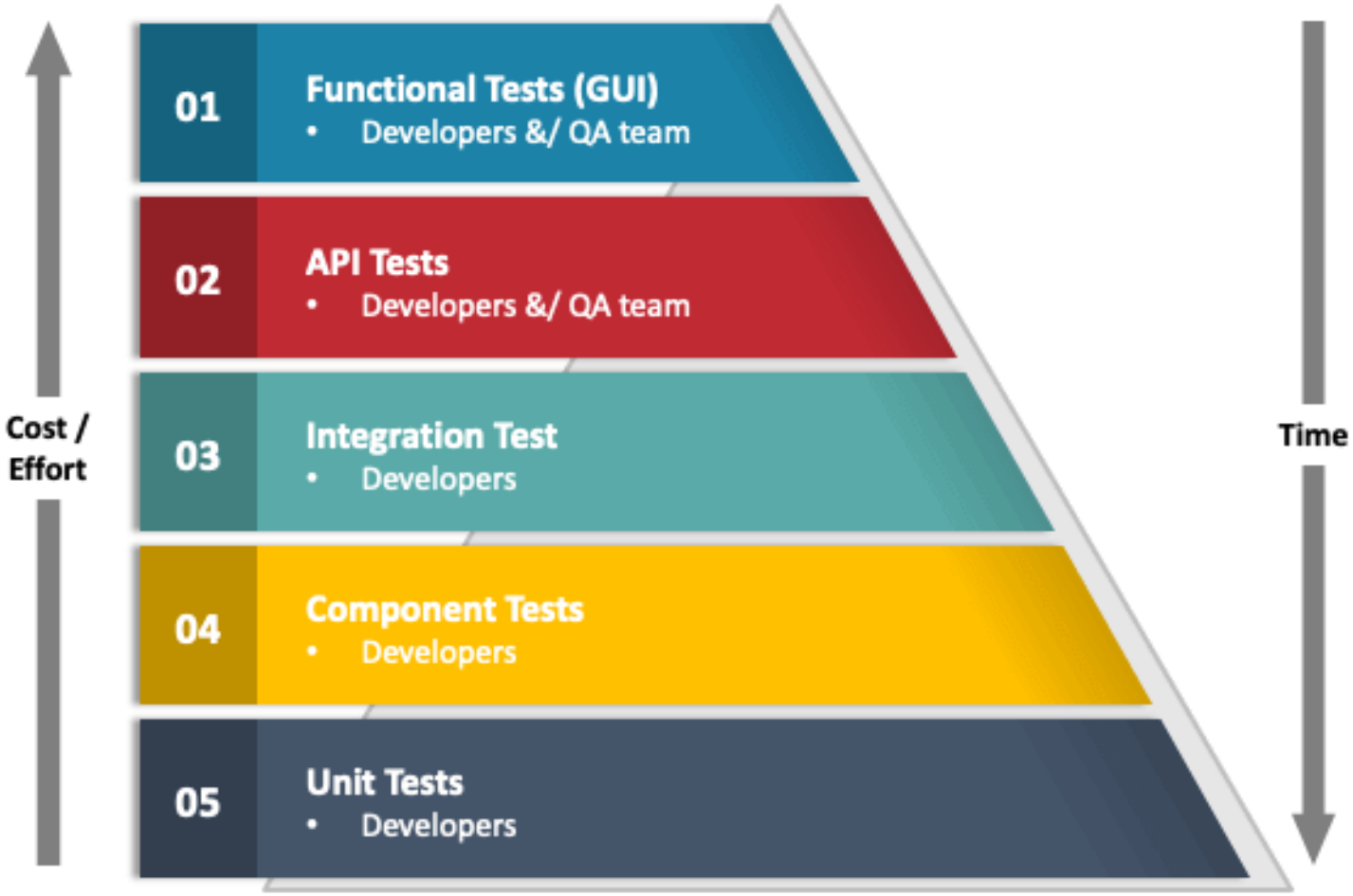
Agenda

01 Elméleti alapok

02 Környezet beállítása

03 Unit tesztelés

Általános bevezető
Teszt piramis



Forrás:: https://www.sketchbubble.com/en/presentation-testing-pyramid.html

Unit tesztelés

Definíció	■ A kód legkisebb tesztelhető egységének ellenőrzése, izoláltan ■ Osztályok, metódusok
Cél	■ Hibák korai felismerése ■ Stabil kódbázis kialakítása
Előnyök	■ Karbantarthatóság ■ Gyors visszajelzés ■ Kód módosítása biztonságosan
Eszköz	■ Unit tesztelési keretrendszerek ■ pld. JUnit 5 (Jupiter - jelentős API-változások JUnit4-hez képest!)
Módszertanok	■ AAA – Arrange / Act / Assert ■ TDD – Test-driven development

JUnit Assertions

Cél	■ Teszteredmény ellenőrzése ■ Helyes viselkedés validálása
Alapvető assert-ek	■ assertEquals(expected, actual); ■ assertNull(actual), assertNotNull(actual);
Boolean ellenőrzése	■ Javasolt: assertTrue(triangle.isIsosceles()); ■ Nem javasolt: assertEquals(true, triangle.isIsosceles());
Kivétel ellenőrzése	■ assertThrows(Exception.class,()-> triangle.isIsosceles());

Komplex elemek összehasonlítása	■ assertSame(expected, actual); //Referenciát hasonlít!
	■ assertArrayEquals(expected, actual); //sorrend

Üzenet hozzáadása	■ assertEquals(expected, actual, „Üzenet”);

Adatvezérelt tesztelés

Cél	■ Egy teszt több különböző adattal történő futtatása ■ Ideális alacsony szintű tesztesetekre
Használata	■ @ParameterizedTest annotációval ■ Forrás megadásával
Források a kódban	■ Egyszerű típusok (@ValueSource) ■ Enumok (@EnumSource) ■ Saját metódus (@MethodSource)
Források külső fájlból	■ @CsvSource(resources=„/path/to.csv”) ■ @CsvSource({ „1,2,3”, „4,5,6” })
Komplex források	■ @MethodSource -> Stream-ként adja vissza az adatot ■ @ArgumentsSource -> saját provider
Paraméterezés	■ Egy elem az inputban -> a metódus egy argumentuma

1. feladat – Triangle osztály tesztelése

Piros szín: nem került implementálásra, UnsupportedOperationException

Adott egy Triangle osztály az alábbi módon:

- Triangle(a,b,c) – inicializálja a háromszöget a megadott 3 oldallal
- isIsosceles() – egyenlő szárú-e a háromszög?
- isEquilateral() – egyenlő oldalú-e a háromszög?
- isRightAngled() – derékszögű-e a háromszög?
- getPerimeter() – visszaadja a kerületet
- getArea() – visszaadja a területet

1. Feladat

- **1) Teszteljük az összes implementált függvényt unit tesztekkel!**
  - Használjuk az alábbi annotációkat: @Test, @DisplayName

```
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
```

- **2) Teszteljük, hogy a nem implementált függvények kivételt dobnak!**

- **3) Írjunk adatvezérelt tesztet, amely különböző adatokkal teszteli az osztályt!**
  - Annotációk: @ParameterizedTest, @MethodSource
  - Az input adatokat egy Stream-ből olvassuk be

```
private static Stream<Arguments> testData() {

    return Stream.of(
      Arguments.of( ...arguments: "generic", 3,4,6, false, false, false),
      Arguments.of( ...arguments: "isosceles", 3,6,6, true, false, false),
      Arguments.of( ...arguments: "equilateral", 10,8,8, true, false, false)
      );
}
```

JUnit - Függőségek

```
import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.junit.jupiter.api.Assertions.assertNull;
import static org.junit.jupiter.api.Assertions.assertThrows;

import org.junit.jupiter.api.*;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
```

1. Feladat - gyakorlás

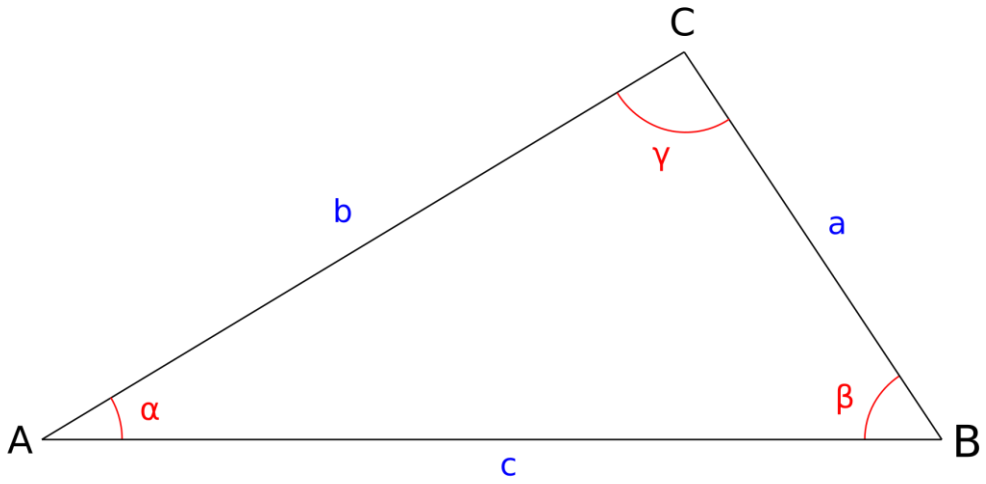
4) Implementáljuk és teszteljük az isRightAngled() és getArea() függvényeket

Segítség: *Hérón-képlet*

- *a*, *b* és *c* a háromszög oldalai
- *s* a háromszög kerületének a fele
- *T* a háromszög területe.

$$T = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a+b+c}{2}$$



2. feladat – TDD-alapú unit tesztelés

Adott egy Table osztály az alábbi módon:

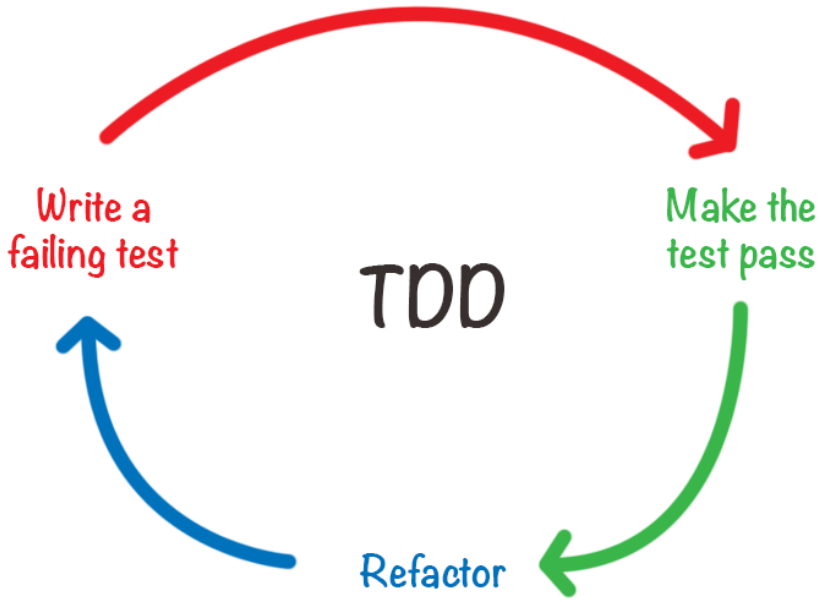
- Konstruktor: Két konstruktor, az isAdjustable és a currentHeight opcionális
- Attribútumok: width, length, height, isAdjustable, currentHeight
- Metódusok:
  - getterek minden attribútumhoz
  - setHeight()
  - area() //szélesség * hossz
  - getCapacity() //60 cm-enként egy személy
- Validációk:
  - Magasság csak akkor állítható, ha isAdjustable == true, és az új magasság a [0,200] intervallumban található

2. Feladat - Table osztály alapja

```
public class Table {
    private int width, length, height, currentHeight;
    private boolean isAdjustable;
    public Table(int width, int length, int height) {…}
    public Table(int width, int length, int height, int currentHeight) {…}
    public void setHeight() {…}
    public int area() {…}
    public int getCapacity () {…}
    public int getWidth() {…}
    public int getHeight() {…}
    public int getLength() {…}
    public int isAdjustable() {…}
    public boolean getCurrentHeight() {…}
}
```

2. Feladat

- **1) Írjunk TDD alapon unit teszteket a setHeight(), area(), getCapacity() függvényekhez**
  - Írjuk meg a tesztesetet
  - Futtassuk, ellenőrizzük, hogy failed (RED)
  - Implementáljuk a hozzá szükséges kódot
  - Futtassuk, ellenőrizzük, hogy passed (GREEN)
  - Alakítsuk át a kódot, ha szükséges (REFACTOR)



2. Feladat - gyakorlás

- **2) Adjunk hozzá még két attribútumot az osztályhoz**
  - color (string)
  - numberOfLegs (int)

- **3) Írjunk metódusokat, és teszteljük le őket:**
  - repaint – módosítja a színt
  - isStable – igaz, ha az asztalnak legalább 3 lába van
  - isFoldable – igaz, ha isAdjustable == true és numberOfLegs >= 4
  - getPerimeter – visszaadja az asztal kerületét

Gyakorló feladat – User osztály tesztelése

Piros szín: nem került implementálásra, UnsupportedOperationException

Adott egy User osztály az alábbi módon:

- Konstruktor: default
- Tulajdonságok: userName, password, id, loginCount
- Metódusok:
  - string getUserName()
  - string getPassword()
  - int getId()
  - boolean isLoggedIn()
  - void login()
  - void logout()
  - int getLoginCount()
  - bool updatePwd(newPwd, isLoggedIn)

Gyakorló feladat

- **1) Hozzunk létre egy felhasználót, majd írjunk unit teszteket az alábbiak ellenőrzése**
  - A user objektum nem null
  - A username és password mezők értéke null
  - Az id és a loginCount mezők értéke 0
  - Az updatePwd metódus hívása hibát dobott

- **2) Bővítsük az osztályunkat TDD használatával**
  - Írjunk egy isLoggedIn unit tesztet úgy, hogy false értéket várjunk el
  - Implementáljuk az isLoggedIn metódust – futtassuk újra az előző tesztünket
  - Hozzunk létre egy üres login metódust, majd írjunk egy unit tesztet az isLoggedIn metódusra úgy, hogy true értéket várjunk a login() meghívása után, és a loginCount legyen 1.
  - Implementáljuk a login metódust, amely az isLoggedIn állapotot true-ra állítja, és megnöveli a loginCount mező értékét eggyel – futtassuk újra az előző tesztünket
  - Hasonló módon írjunk unit tesztet a login() és a getLoginCount() metódusra is
  - Hasonló módon implementáljuk a logout metódust is