LAPORAN TUGAS BESAR CLO 2

CUI AUTO VENDING Using C# with Visual Studio 2022

Dibuat Oleh:

Radinka Putra R 1201220020

SOFTWARE ENGINEERING

TELKOM UNIVERSITY

2025

DAFTAR ISI

DESKRIPSI SINGKAT

Auto Vending adalah aplikasi smart vending machine berbasis self-checkout yang memungkinkan pengguna melakukan pemesanan dan pembayaran produk secara mandiri tanpa bantuan operator. Aplikasi ini dikembangkan menggunakan C# dalam Visual Studio 2022 dan dirancang untuk diintegrasikan langsung dengan perangkat vending fisik. Dengan antarmuka yang intuitif dan pengalaman transaksi yang cepat, Auto Vending bertujuan untuk meningkatkan efisiensi layanan penjualan otomatis, serta memberikan fleksibilitas tinggi dalam pengelolaan produk dan konfigurasi sistem.

Fitur-fitur yang tersedia:

- Perubahan Status
- Pemilihan Produk
- Ganti Bahasa
- Manajemen Produk
- Jam operasional
- Convert Mata Uang
- Checkout
- Transaksi

DAFTAR ANGGOTA KELOMPOK

1. Zidan Irfan Zaky 1201220003
2. Farhan Nugraha Sasongko Putra 1201220449
3. Radinka Putra Rahadian 1201220020
4. Giovan Deo Pratama 1201220450
5. Evi Fitriya 1201222005

Dengan Pembagian Tugas sesuai arahan dimana 1 teknik konstruksi dipegang oleh maximal 2 orang:

Nama Anggota / Nama Teknik konstruksi	Zidan Irfan Zaky	Farhan Nugraha Sasongko	Radnka Putra Rahadian	Giovan Deo Pratama	Evi Fitriya	Total Teknik Dipegang
Automata		✓			✓	2
Table Driven		✓	✓			2
Runtime	✓			✓		2
Code Reuse			✓		✓	2
Generics	✓			✓		2
Total Peserta Memegang	2	2	2	2	2	

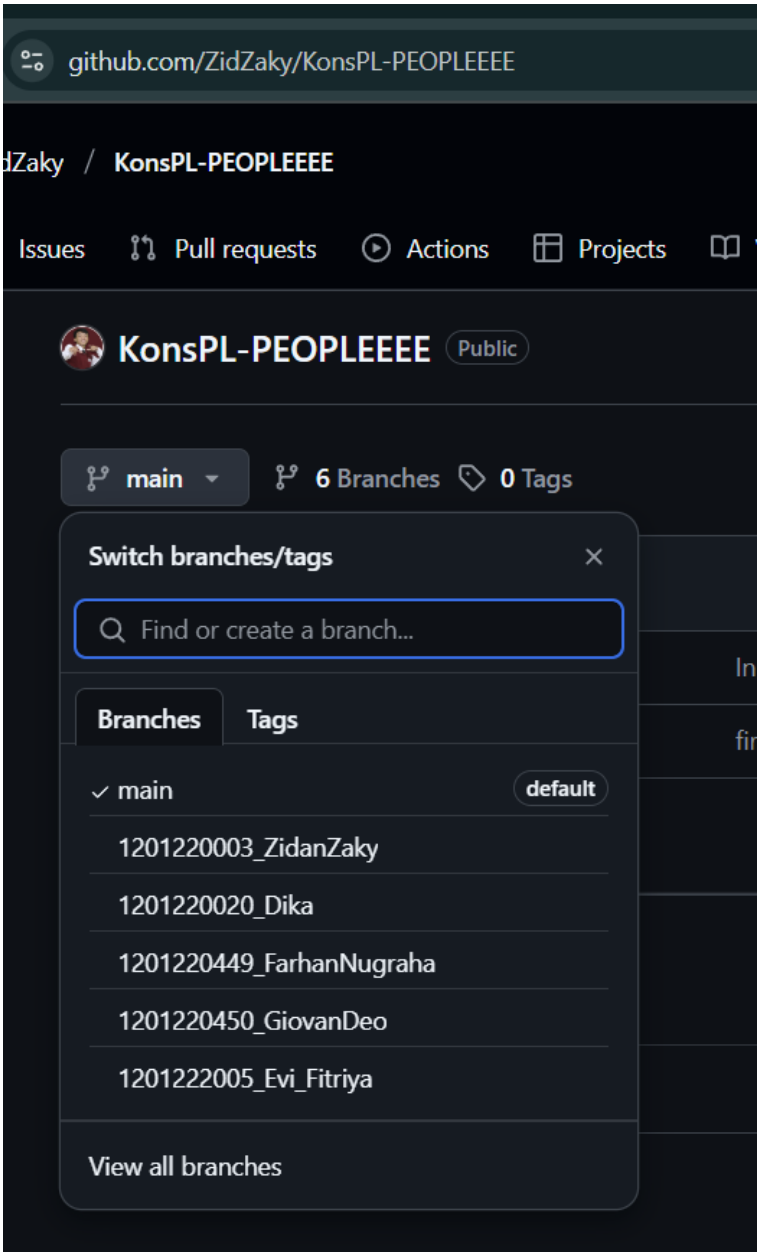
GITHUB

Repositori ini menggunakan sistem version control GitHub untuk mendokumentasikan proses konstruksi proyek secara kolaboratif. Terdapat *branch* pribadi yang digunakan untuk pengembangan fitur atau modul secara terpisah, sebelum digabungkan ke *branch* utama (*main*) melalui proses *commit* dan *merge*. Pendekatan ini memastikan riwayat perubahan tercatat dengan jelas serta mempermudah kolaborasi dan pengelolaan versi.

Berikut Link Github Kami:

https://github.com/ZidZaky/KonsPL-PEOPLEEEE.git

Dengan bukti:



IMLEMENTASI DESIGN BY KONTRAK

Design by Contract (DbC) adalah pendekatan pemrograman yang menetapkan *kontrak* eksplisit antara sebuah metode dan pengguna metode tersebut. Kontrak ini berupa *precondition* (syarat sebelum dieksekusi), *postcondition* (hasil setelah eksekusi), dan *invariant* (aturan yang selalu benar dalam objek)

Dalam kode VendingMachine, DbC diterapkan pada kelas Product:

```
namespace VendLib
{
    27 references
    public class Product
    {
        13 references | 9/9 passing
        public string Name { get; }
        13 references | 9/9 passing
        public decimal Price { get; }

        18 references | 1/1 passing
        public Product(string name, decimal price)
        {
            // Preconditions
            if (string.IsNullOrWhiteSpace(name))
                throw new ArgumentException("Name tidak boleh kosong.");
            if (price < 0)
                throw new ArgumentOutOfRangeException("Price harus >= 0.");

            Name = name;
            Price = price;
        }
    }
}
```

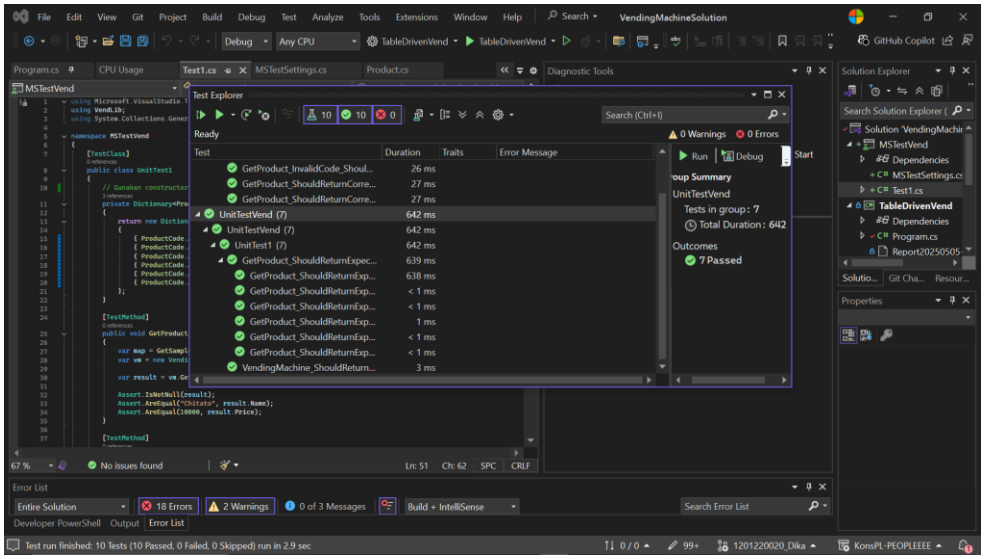
```
4 references | 4/4 passing
public VendingMachine(Dictionary<ProductCode, Product> customMap)
{
    if (customMap == null || customMap.Count == 0)
        throw new ArgumentException("Product map tidak boleh null atau kosong.");

    _products = customMap;
}
```

Dbc validasi di constructor

- **Precondition**: Nama produk tidak boleh kosong, dan harga tidak boleh negatif.
- Jika kondisi ini dilanggar, maka *kontrak gagal* dan program akan melempar exception.
- Tujuannya adalah menjaga agar semua objek Product **selalu dalam keadaan valid (invariant)**.

HASIL UNIT TESTING

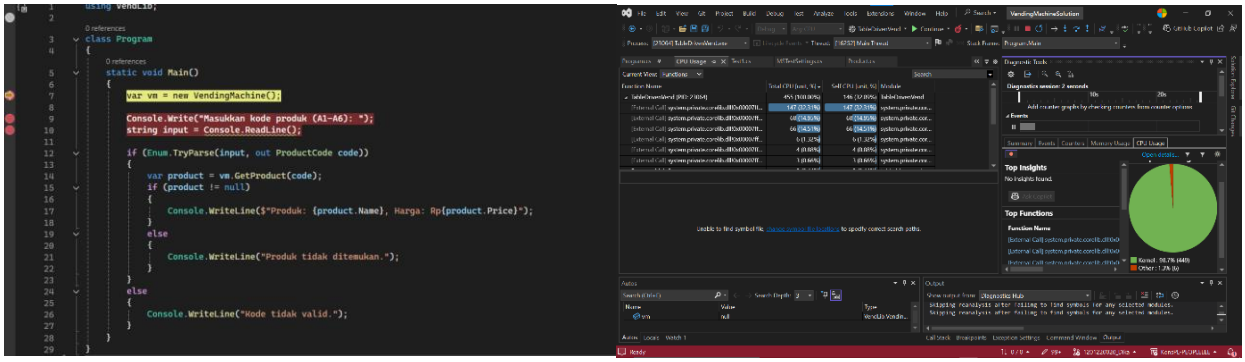


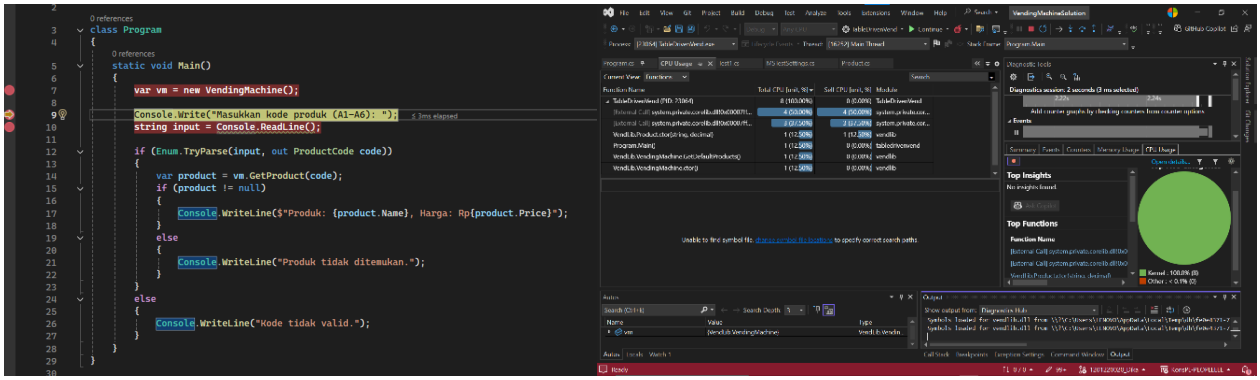
Dalam proyek MSTestVend, pengujian dilakukan menggunakan **kerangka kerja MSTest** untuk memastikan bahwa fungsionalitas dari kelas VendingMachine berjalan sesuai harapan. Beberapa jenis pengujian yang dilakukan antara lain:

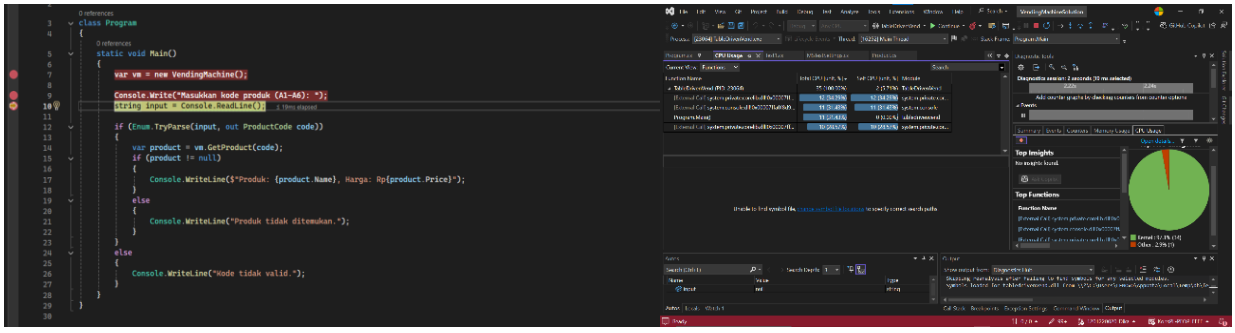
1. **Positive Test Case – Valid Product Code**
   Pengujian dilakukan untuk memverifikasi bahwa metode GetProduct akan mengembalikan produk yang benar saat diberikan kode produk yang valid, seperti A1 (Chitato) dan A6 (Tic Tac).
   Tujuan: memastikan sistem berfungsi sesuai ekspektasi ketika input benar.
2. **Negative Test Case – Invalid Product Code**
   Pengujian dilakukan dengan memberikan kode produk yang tidak tersedia, seperti (ProductCode)999, dan memverifikasi bahwa hasilnya adalah null.
   Tujuan: memastikan sistem mampu menangani input yang tidak valid tanpa crash.
3. **Boundary Testing (implisit)**
   Karena produk didefinisikan hanya sampai A6, maka pengujian terhadap kode di luar batas (999) sekaligus berfungsi sebagai *boundary test* untuk memastikan tidak ada akses data di luar range.

Secara keseluruhan, pengujian ini termasuk dalam kategori **unit testing**, karena fokus pada satu unit kecil kode (VendingMachine) dan menggunakan data simulasi (customMap) tanpa bergantung pada input eksternal.

HASIL PERFOMANCE TESTING







Hasil Pengujian Performa Aplikasi TableDrivenVend

Pengujian performa dilakukan menggunakan fitur **Diagnostic Tools** pada Visual Studio dengan memfokuskan pada **CPU Usage** saat aplikasi dijalankan. Dari hasil pengujian diperoleh bahwa fungsi Program.Main() memanggil VendingMachine.GetProduct() dan VendingMachine.GetDefaultProducts() dengan konsumsi CPU yang relatif ringan.

Beberapa temuan penting:

- **Total penggunaan CPU** aplikasi sebesar 100%, namun sebagian besar disumbangkan oleh **external call dari corelib** milik .NET, yang artinya beban utama berada pada sistem runtime.
- Fungsi internal seperti GetProduct() dan konstruktor VendingMachine() hanya menyumbang sekitar **12.5% CPU masing-masing**, menandakan efisiensi dalam eksekusi metode tersebut.
- Fungsi Product.ctor(string, decimal) sebagai bagian dari inisialisasi data produk juga memakan sedikit resource, dengan total hanya 12.5%.
- Tidak ditemukan adanya fungsi dengan konsumsi CPU tinggi atau indikasi bottleneck selama eksekusi.
- Hasil ini menunjukkan bahwa aplikasi *VendingMachine* sudah cukup optimal dan tidak menunjukkan adanya inefisiensi signifikan dalam pemrosesan logika produk berdasarkan kode input pengguna.

Dengan demikian, performa aplikasi terukur stabil dan efisien untuk kebutuhan simulasi vending machine berbasis konsol.