Task Manager Desktop App

SİSTEM PERFORMANS MONİTÖRÜ MASAÜSTÜ UYGULAMASI RAPORU

İşletim Sistemleri | 23.12.2024

İÇİNDEKİLER

Proje

- **Proje Tanımı:** Linux/Windows üzerinde çalışan süreçleri ve kaynak kullanımını izleyen bir performans izleyici uygulaması geliştirin. CP U kullanımı, bellek kullanımı, açık dosyalar gibi bilgileri toplayın ve bir kullanıcı arayüzünde gösterin.
- **Amaç:** İşletim sistemi süreçlerinin nasıl yönetildiğini öğrenmek ve kaynak kullanımı izleme araçları geliştirmek.
- **Teknolojiler:** C/C++, Python (psutil kütüphanesi), Java Swing GUI
- **Adımlar:** Süreç tablosu çekme, CPU ve bellek kullanımı ölçümü, basit bir kullanıcı arayüzü oluşturma.

SEÇİMLERİMİZ

- Uygulamamız Windows işletim sistemi üzerinde çalışmaktadır.
- Masaüstü uygulamasının arayüz kısmı Java Swing GUI ile yapılmıştır.
- C++ aracılığıyla CPU, bellek kullanımı ve açık dosyalar donanım üzerinden çekilmiştir.
- C++ kodu Java üzerinde otomatik olarak başlatılıp donanımdan çekilen bilgiler File işlemleri ile .txt dosyasına yazılmış ve Java üzerinden ilgili .txt dosyaları okunaraktan arayüze aktarılmıştır.

SEÇİMLERİMİN NEDENLERİ

Projemin gereksinimleri CPU, bellek kullanımının ve açık olan dosyaların basit bir kullanıcı arayüzünde görünmesidir. Bunun üzerine donanımdan bu bilgilerin çekilmesi ve gerekli hesaplamalar veya işlemler yapıldıktan sonra kullanıcı arayüzüne aktarılması gerekiyor. Bilgilerin donanımdan çekilmesi kısmını gerçekleştirmek için diğer dillere göre daha düşük seviyeli bir dil olan C++ ' ı tercih ettim. Kullanıcı arayüz kısmı için bana verilen seçenekleri değerlendirdim. Araştırmalarım sonucunda C ve C++ da arayüz uygulaması hayata geçirmenin bir hayli zor olduğunu bildiğimden ayrıca Python'un ise Java ya göre eksilerinin olmasından dolayı Java 'da yazmaya karar kıldım. Java ile arayüz uygulaması gerçeklemenin çok farketmese de birkaç farklı yolu vardı. Deneyimlerimden ve Java Swing GUI 'ın daha kolay kullanılabilir olmasından ötürü kararımı bu şekilde verdim.

Verileri donanımdan alma işlemini C++ ile yapmayı ancak kullanıcı arayüzünün Java ile yazmaya karar verdiğim için donanımdan alınan verilerin bir şekilde Java koduna iletilmesi gerekiyordu.

İki Farklı Yazılım Dilinin İletişimi için aklıma iki farklı yol geldi;

1. ***Soket Programlama***:"Temel olarak alıcı ve gönderici arasındaki iletişim yönetmek için kullanılan programlama tekniğidir." C++ ile soket oluşturup (localhost) Java'da ise sokete bağlanmaya çalışılacak ve donanımdan çekilen veriler soket üzerinden Java uygulamasına aktarılabilir.
2. ***File (Dosya) İşlemleri*** : Dosyaya yazma, okuma, silme işlemlerinin yapılmasıyla bir yazılım dili ile bir dosyaya yazılıp diğer yazılım diliyle aynı dosyanın okuması aracılığıyla iletişimin sağlanmasıdır. C++ ile donanımdan okunan veriler oluşturulan .txt(metin belgesi) dosyasına yazılması ve Java uygulamasının bu .txt dosyasını okuması ile veriler arayüze aktarılabilir.

Başlangıçta soket programlama ile uğraştım, iletişimi sağlamayı başarmama rağmen belli başlı hatalar almam sebebiyle daha kolay bir yol olan File işlemleri ile iletişimi sağlamayı karar kıldım.

UYGULAMANIN TANITIMI

Uygulamam bir masaüstü uygulamasıdır. Basit bir kullanıcı arayüzüne sahiptir. Windows işletim sisteminde kurulu olarak gelen "Görev Yöneticisi" uygulamasına benzer bir uygulamadır. Uygulamayı çalıştıran **.jar** dosyasının çalışabilmesi için çalıştırılacak cihazda **JDK** yüklü olması gereklidir.

(Güncel JDK linki : https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe)

Uygulamada görünteleyebileceğiniz üç kısım vardır :

1. ***CPU Kullanımı*** : "Merkezi işlem birimi (CPU), sunucudaki temel bilgi işlem birimi olan donanım bileşenidir. Sunucular ve diğer akıllı cihazlar, verileri dijital sinyallere dönüştürür ve bunlar üzerinde matematiksel işlemler gerçekleştirir. CPU, sinyalleri işleyen ve bilgi işlem olanağı sağlayan birincil bileşendir.", CPU kullanımı ise "Bir bilgisayarın merkezi işlemcisinin görevleri yürütmede aktif olarak ne kadar zaman harcadığını ölçen bir ölçümdür. CPU kullanım yüzdesi, işlemcinin boşta kalma süresine kıyasla hesaplamaları aktif olarak gerçekleştirmek için harcadığı zaman oranını gösterir." Uygulamamızda CPU kullanımı hem % olarak gösterilmekte hem de grafik olarak gösterilmektedir.

2. ***Bellek Kullanımı*** : "Toplam kullanılabilir belleğe göre kullanımda olan bellek miktarını temsil eden bir yüzde olarak ifade edilir. Örneğin, bir sunucunun 8 GB RAM'i varsa ve şu anda 6 GB kullanıyorsa, bellek kullanımı %75 olur." Uygulamamızda bellek kullanımı hem % olarak hemde kullanılan **MB** cinsinden gösterilmektedir. Aynı zamanda grafik bölümünde grafik olarak da değişim görüntülenebilmektedir.

3. ***Açık Olan Dosyalar*** : Aktif olarak açık olan dosyalar donanımdan çekilen bilgiler sayesinde uygulamanın kullanıcı arayüzünde görüntülenebilmektedir. Yeni bir dosya açılırsa bu sekmede **dosyanın ismi**, **Process ID** 'si, **Process ismi** gibi bilgiler görüntülenmektedir. Aynı şekilde eğer açık olan bir dosya kapatılırsa bu dosyanın bilgilerinin görüntülenmesi de son bulacaktır.

KAYNAK KODU

Bu kısımda uygulamamı gerçekleyen kodlara yer verdim. Uygulama kaynak kodu açısından iki kısıma ayrılabilir :

1. Donanımdan bilgilerin çekildiği **C++** kısmı
2. Donanımdan çekilen bilgilerin üstünden hesaplmaların yapıldığı, arayüzün oluşturulduğu **Java** kısmı

IDE olarak;

- ***C++ için Visual Studio***
- ***Java için Intellij IDEA***

kullandım.

DONANIMDAN BİLGİ ÇEKİLEN KISIM

Bu kısımda C++ kodları göreceksiniz.

GEREKLİ OLAN KÜTÜPHANELER VE AÇIKLAMALARI

- ***#include <windows.h>*** → Windows işletim sistemi API'lerine erişim sağlayan temel kütüphanedir. Kullandığımız *GetWindowThreadProcessId*, *OpenProcess*, *GetLastError* gibi fonksiyonlar bu kütüphane ile sağlanır.

- ***#include <tlhelp32.h>*** → Windows sistemindeki işlem (process) ve thread bilgilerine erişim sağlayan kütüphanedir. Açık işlemleri listelemek, işlem bilgilerine (ID, isim vb.) erişmek için kullanılır. *EnumWindows* fonksiyonu, pencereleri ve ilişkili işlemleri taramak için bu kütüphaneden gelen özelliklerle çalışır.

- ***#include <psapi.h>*** → Windows API'si ile ilgili, özellikle işlem (process) ve modül (module) bilgilerine erişmek için kullanılan bir kütüphanedir. Bir işlemin çalıştırdığı modülleri (örneğin, .exe veya .dll dosyalarını) almak için kullanılır.GetModuleBaseNameW fonksiyonu ile bir işlemin adı (processName) alınır.

- ***#include <pdh.h>*** → Windows Performance Data Helper (PDH) API'si, performans sayaçları ve metriklerine erişim sağlayan bir kütüphanedir. CPU ve bellek kullanımı gibi performans verilerini almak için kullanılır. PdhOpenQuery, PdhAddCounter ve PdhCollectQueryData gibi fonksiyonlarla CPU kullanımını sorgulamak için gereklidir.

- ***#include <iostream>*** → C++ dilinde temel giriş/çıkış (I/O) işlemleri için kullanılan kütüphanedir. Konsola veri yazdırmak için cout, hataları yazdırmak için cerr gibi akışları kullanır.

- ***#include <string>*** → C++'ta metin verilerini işlemek için kullanılan temel string sınıfı. Uygulamanın çıktılarında string verilerini (Unicode karakterler) işlemek için kullanılır.

- ***#include <map>*** → C++'ta anahtar-değer (key-value) çiftleri ile veri saklamanızı sağlayan bir konteyner sınıfıdır. Açık uygulamaların ve işlemlerin saklanmasında kullanılır. map<DWORD, wstring> tipi, her bir işlem için işlem ID'si ve uygulama adını saklamak için kullanılır.

- ***#include <locale>*** → C++'ta yerel (locale) ayarlarını yapmayı sağlayan kütüphanedir. Türkçe karakterler gibi özel karakterlerin doğru şekilde işlenmesini sağlamak için kullanılır. locale nesnesi, dosya işlemlerinde doğru karakter setini (UTF-8) kullanmak için ayarlanır.
- ***#include <fstream>*** → Dosya okuma ve yazma işlemleri için kullanılan C++ kütüphanesidir. wofstream sınıfı, geniş karakter (wchar_t) tipinde dosyalarla işlem yapmak için kullanılır. Program, sistem kullanım ve açık uygulama bilgilerini dosyaya yazarken bu kütüphane ile çalışır.

- ***#include <thread>*** → Çoklu iş parçacığı (multithreading) programlaması için kullanılan kütüphanedir. Arka planda veri toplama ve dosyaya yazma işlemleri için ayrı bir thread başlatır. Bu sayede, ana programın çalışmasına engel olmadan dosya işlemleri yapılabilir.

- ***#include <chrono>*** → Zamanla ilgili işlemler için kullanılan C++ kütüphanesidir. high_resolution_clock sınıfı, işlem sürelerini milisaniye cinsinden hesaplamak için kullanılır. Veri güncelleme sürelerini ölçmek için kullanılır.

- ***#include <mutex>*** → C++'ta iş parçacıkları arasında veri erişimini senkronize etmek için kullanılan kütüphanedir. Dosyaya erişim sırasında veri çakışmalarını önlemek için mutex kullanılır. Bu sayede bir thread dosya yazarken, diğer thread dosyaya yazamaz.

- ***#include <codecvt>*** → Karakter dönüşümü yapmak için kullanılan kütüphanedir. codecvt_utf8<wchar_t> sınıfı, geniş karakterleri (Unicode) UTF-8 formatına dönüştürmek için kullanılır. Bu, Türkçe karakterlerin doğru şekilde işlenmesini sağlar.

- ***#pragma comment(lib, "pdh.lib")*** → Bu komut, **PDH kütüphanesini** (pdh.lib) proje dosyasına otomatik olarak bağlar. Windows'un Performance Data Helper (PDH) API'sini kullanmak için gerekli olan kütüphanedir. **CPU kullanımı** gibi performans sayaçlarıyla ilgili verilere erişmek için kullanılır. Bu API'den gelen fonksiyonlar PdhOpenQuery, PdhAddCounter, PdhCollectQueryData bu kütüphaneye ihtiyaç duyar.

- ***#pragma comment(lib, "psapi.lib")*** → psapi.lib kütüphanesini projeye bağlar. Windows API'nin **Process Status API** (PSAPI) kısmını içeren bir kütüphanedir. PSAPI, işlemler (processes) ve modüller (modules) hakkında bilgi toplamak için kullanılır. bir işlem adı almak için kullanılan GetModuleBaseNameW gibi fonksiyonlar PSAPI kütüphanesine bağımlıdır. Bu fonksiyonlarla işlem ve uygulama bilgileri alınır.

- ***using namespace std;*** → Bu ifade, `std` namespace'ini (ad alanını) kullandığınızı belirtir. C++ Standard Library'nin (standart kütüphanenin) içindeki sınıf, fonksiyon, ve nesneler varsayılan olarak std namespace'inde bulunur. Örneğin, std::cout, std::map, std::wstring, std::endl, vb.std namespace'ini belirtmeden standart kütüphanedeki elemanları kullanabilmek için yazılır. Örneğin, std::cout yerine doğrudan cout yazabilirsini

METODLARIN AÇIKLAMALARI

- ***enumWindowsCallback :*** Sistem üzerindeki tüm açık pencereleri tarar. Her pencere için işlem kimliği (Process ID) ve pencere başlığını alır.
- ***getProcessName :*** Bir işlem kimliği (Process ID) alarak, o işleme ait çalıştırılabilir dosyanın (process) adını döndürür.

- ***listOpenApplications :*** Açık uygulamaların listesini çıkarır. Her uygulamanın adı, işlem adı ve kimliği dosyaya yazılır."open_applications.txt" dosyasına açık uygulama bilgilerini kaydeder.

- ***getCpuUsage :*** PDH API kullanarak sistemin toplam CPU kullanımını yüzdesel olarak hesaplar. "system_usage.txt" dosyasına CPU kullanım oranını kaydeder.

- ***getMemoryUsage :*** Sistemin bellek kullanımını yüzdesel (%), kullanılmakta olan bellek miktarını ise megabayt (MB) olarak hesaplar. "system_usage.txt" dosyasına bellek kullanım oranını ve miktarını kaydeder.

- ***writeDataToFile :*** "`open_applications.txt`" dosyasına açık uygulama bilgilerini yazar."system_usage.txt" dosyasına CPU ve bellek kullanım oranlarını yazar. işlem sırasında dosya erişimini güvenli hale getirmek için **mutex** kullanır. Bilgileri dosyalara kaydeder ve yazma işlemlerini optimize eder.

- ***main :*** Sürekli olarak her 600 milisaniyede bir (`writeDataToFile`) metodu çağırarak bilgileri dosyalara günceller. Türkçe karakter desteği için setlocale kullanır. Programın ana döngüsü ve çalışma mantığıdır.

ARAYÜZ KISMI (JAVA SWING GUI)

Asıl uygulamanın çalıştığı kısım bu kısımdır. Uygulama .jar dosyasının açılmasıyla çalışmaya başlanır. Ve bu java kısmında arayüzle ilgili kodlamalar ve c++ kodunu çalıştıran kısım thread ile .txt dosyasına yazma ve okuma sırasını paylaşmayı thread ile belirli milisaniye sonrasında veriyi güncellemeyi sağlar. Java kod kısmında grafik ayarlarıyla ilgili kısımda vardır. Java kısmını yazmak için "IntelliJ IDEA" ide 'sini kullandım. Aşağıda java kısmında yazılmış olan metodlar ve fonksiyonların açıklamasına yer verilmiştir.

GraphPanel Class'ı :

Bu sınıf, ProjeGUI içinde kullanılan ve CPU ile bellek kullanımını görselleştiren grafik panelini tanımlar.

Alanlar (fields) :

cpuData ve memoryData: CPU ve bellek kullanımı yüzdelerini içeren listeler.

Metodlar-Fonksiyonlar(method-functions) :

- ***GraphPanel(List<Double> cpuData, List<Double> memoryData) :*** Grafik panelini başlatır ve CPU ile bellek verilerini alır.
- ***paintComponent(Graphics g) :*** Paneli yeniden çizmek için çağrılır. CPU ve bellek kullanımı grafiğini ve dolu alanları çizer.
- ***drawTitle(Graphics g, String title, int yOffset, Color color) :*** CPU veya bellek grafiği için başlık çizer.
- ***drawGraph(Graphics g, List<Double> data, int yOffset, Color color) :*** CPU veya bellek verilerini çizgi grafiği olarak çizer.
- ***fillGraphArea(Graphics g, List<Double> data, int yOffset, Color color) :*** Çizilen grafiğin alt kısmını opak bir renk ile doldurur.
- ***drawGrid(Graphics g) :*** Grafik paneli arka planına kareli bir grid ekler.
- ***updateData(List<Double> newCpuData, List<Double> newMemoryData)*** CPU ve bellek verilerini günceller ve paneli yeniden çizer.

TakeTheHardware Class'ı :

Bu sınıf, donanım verilerini almak için kullanılan bir yardımcı sınıftır.

Alanlar (fields) :

- cppProcess : C++ programını çalıştırmak için kullanılan Process nesnesi.
- applicationsFilePath ve systemUsageFilePath: Donanım bilgilerini içeren dosyaların yollarıdır.

Metodlar-Fonksiyonlar(method-functions) :

- ***startCppProgram()*** : C++ programını çalıştırır. Program donanım bilgilerini open_applications.txt ve system_usage.txt dosyalarına yazar.
- ***addShutdownHook()*** : Java uygulaması kapatıldığında C++ işlemini sonlandırır ve geçici dosyaları siler.
- ***deleteFile(String filePath)*** : Parametre olarak gönderilen dosya yolunda bulunan dosyayı siler.(Uygulama kapatılırken bu fonksiyon da çağrılır.)
- ***readFile(String filePath)*** : Belirtilen dosyayı okur ve içeriğini bir String olarak döner(return eder).
- ***extractCpuUsage(String line)*** : CPU kullanım yüzdesini bir satırdan ayrıştırır. Eğer veri mevcut değilse -1 döner(return eder).
- ***extractMemoryUsage(String line)*** : Bellek kullanım yüzdesini ve miktarını bir satırdan ayrıştırır. Eğer veri mevcut değilse -1 döner(return).

ProjeGUI Class'ı :

Bu sınıf, Java Swing kullanarak bir grafiksel kullanıcı arayüzü (GUI) oluşturur. Kullanıcıya, sistem donanım bilgileri ve açık uygulamalarla ilgili verileri gösterir.

Alanlar (fields) :

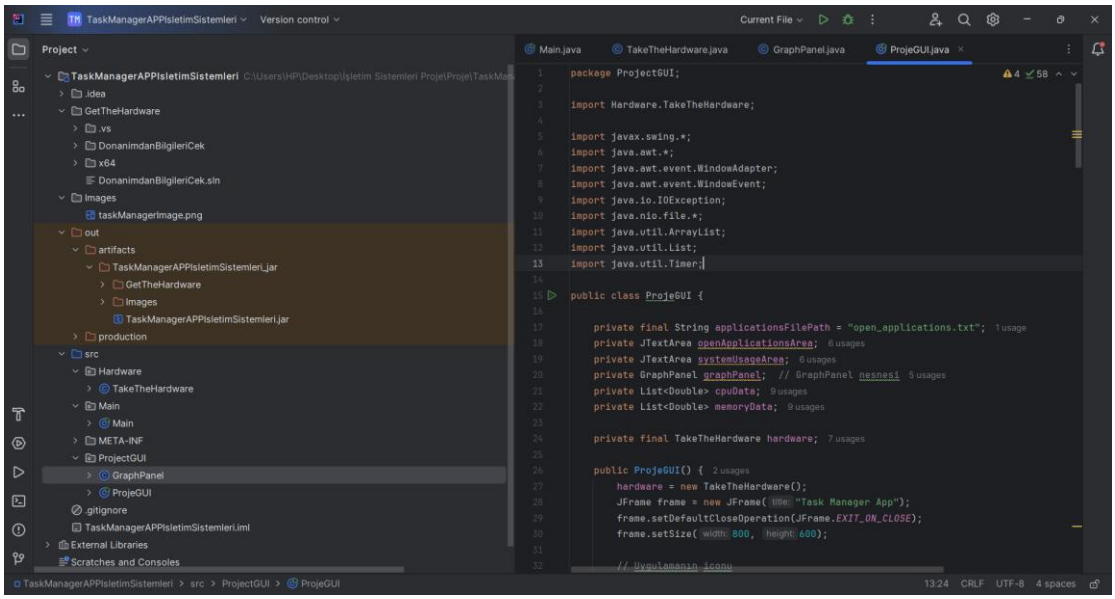
- ***applicationsFilePath*** : Açık uygulamaların saklandığı dosyanın yolu.
- ***openApplicationsArea ve systemUsageArea*** : GUI'de açık uygulamalar ve sistem kullanımı bilgilerini göstermek için kullanılan metin alanları.
- ***graphPanel*** : CPU ve bellek kullanımı grafiğini çizen özel bir JPanel.
- ***cpuData ve memoryData*** : CPU ve bellek kullanımı yüzdelerini saklayan listeler.
- ***hardware***: TakeTheHardware sınıfı üzerinden donanım verilerini okumak için kullanılan nesnedir.

Metodlar-Fonksiyonlar(method-functions) :

- ***ProjeGUI():*** (Constructor)Yapıcı metoddur. GUI bileşenlerini oluşturur ve düzenler, C++ programını başlatır (hardware.startCppProgram), Verileri düzenli olarak güncellemek için bir Timer ayarlayıp çalıştırır, Dosya değişikliklerini takip etmek için bir "File Watcher" başlatır (startFileWatcher).
- ***updateUI() :*** GUI'yi güncellemek için açık uygulamaları ve sistem kullanımını okur ve grafik verilerini yeniler.
- ***updateOpenApplications() :*** `open_applications.txt` dosyasını okur ve metin alanını günceller
- ***updateSystemUsage():*** system_usage.txt dosyasını okur, CPU ve bellek kullanımını ayrıştırır ve listeye ekler. Grafik verilerini sınırlandırır (son 100 veri).
- ***startFileWatcher()*** : Dosyaları gerçek zamanlı olarak izlemek için bir WatchService başlatır. Dosyada bir değişiklik olduğunda UI güncellenir

KOD DÜZENİ

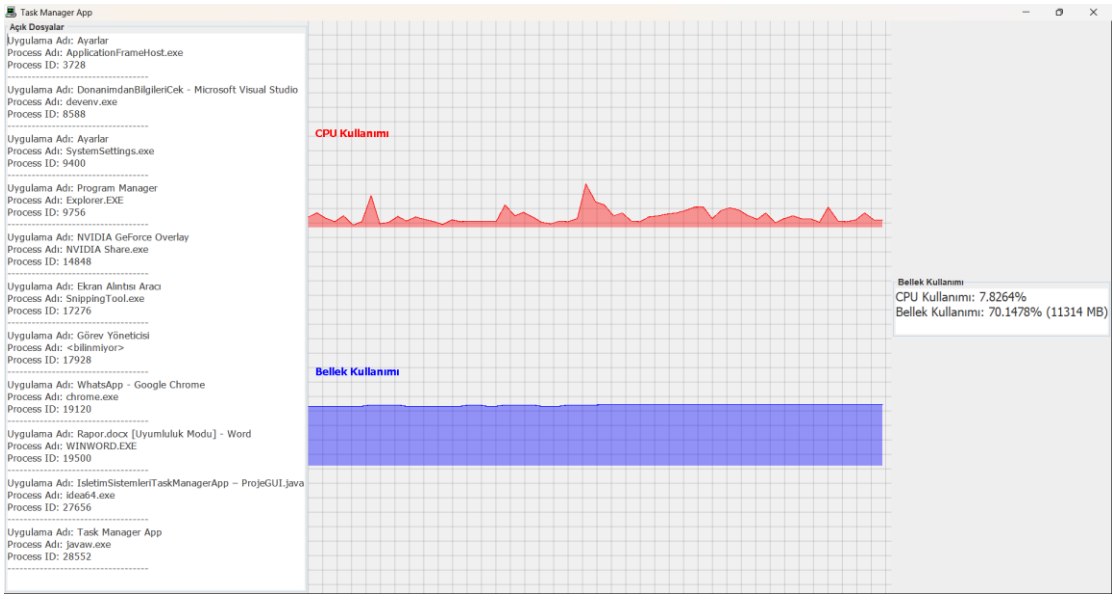
Javadaki kod düzeni aşağıda bulunan görseldeki gibidir. Metot,Fonksiyonlar gerekli konu başlıklarına göre Class'larda toplanmış Class'lar ise ilişkili oldukları alanlara göre Package'lara yerleştirilmiştir.



1. ***GetTheHardware Package :*** Bu paketin içinde C++ kodunun da olduğu dosya vardır. Hem c++ kodunun .exe uzantılı hali hem de .sln kaynak kodu vardır.
2. ***Images Package :*** Masaüstü uygulamasının çalışırken gözüken iconu bu pakette bulunmaktadır.
3. ***Hardware Package :*** Donanımla ilgili işlemlerin yapıldığı pakettir.
4. ***Main :*** Ana programın çalıştığı pakettir. İçinde main fonksiyonun bulunduğu Main Classı vardır.
5. ***ProjectGUI Package :*** Bu paketin içinde Arayüzle ilgili Class'lar Metod'lar vardır. GraphPanel Class'ı CPU/Bellek kullanımı verilerindeki değişime bağlı olarak arayüze grafik oluşturmayı sağlayan metodları barındırır. ProjeGUI Class'ı arayüzle ilgili işlemleri ve genel uygulamanın işleyişiyle ilgili metodları barındırır.

MASAÜSTÜ UYGULAMAMIZIN EKRAN GÖRÜNTÜSÜ

Aşağıdaki görselde masaüstü uygulamamın çalışırkenki ekran görüntüsü gözükmektedir.(Windows işletim sistemine sahip bir bilgisayar)



- Uygulamanın sol kısmında yer alan kısım Açık Dosyaların görüntülenebilceği kısımdır. Herhangi bir yeni dosya, uygulama, süreç açıldığı takdirde sol kısımdaki veriler güncellenir ve o kısımda görüntülenebilir, aynı şekilde orda gözüken bir dosya nezaman kapatılırsa oradan da silinir.
- Uygulamanın orta kısmında CPU/Bellek kullanımı verilerindeki değişikliliğe bağlı olarak grafik görüntülenebilir. Üst kısımda CPU kullanımının grafiği görüntülenirken alt kısımda Bellek kullanımının grafiği görüntülenmektedir.(CPU grafiği aktif olarak daha belirgin değişse de Bellek kullanım grafiğindeki fark okadar azdırki farkedilmesi insanı zorlamaktadır.)
- Uygulamanın sağ kısmında CPU/Bellek kullanımının sayısal değeri % olarak ve kullanılan belleğin MB cinsinden değeri görüntülenebilmektedir. Bu veriler her saniye güncellenmektedir.