**Vaka-2 Askeri Araç Sistemi Tasarım Raporu**

Bu rapor, **Askeri Araç Sistemi** adıyla geliştirilen ve Ada dilinde yazılmış olan yazılımın tasarım detaylarını içermektedir. Proje, askeri hava, kara ve deniz araçlarını kayıt altına alma, filtreleme, sorgulama ve güncelleme işlevlerini yerine getirmektedir. Sistemin XML dosyası kullanarak veri saklama özelliği vardır. Bu tasarım raporu, gereksinimler, mimari tasarım, algoritma analizleri ve uygulama detayları üzerine odaklanmıştır.

## 1. Genel Amaç

Bu program, çeşitli askeri araçları kaydetmek, listelemek ve filtrelemek üzere tasarlanmıştır. Kullanıcılar:

* Araç bilgilerini girebilir,
* Mevcut araçları listeleyebilir,
* Belirli kriterlere göre araçları filtreleyebilir. Ayrıca, sistem eklenen araç bilgilerini XML formatında bir dosyada saklar,
* Araç adı alınarak mevcut durumunu güncelleyebilir,
* Programdan çıkış yapabilir.

## 2. Kullanılan Veri Yapıları ve Tanımlar

### 2.1. Araç ve Durum Tipleri

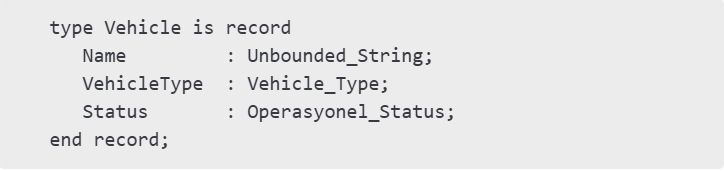
Projede, askeri araçların farklı özelliklerini temsil etmek için iki temel "enum" tipi tanımlanmıştır. "Enum" türleri, sabit ve önceden tanımlanmış değerler kümesi oluşturmak için kullanılır. Bu, kodun okunabilirliğini artırır, hataları azaltır ve tür güvenliği sağlar. Örneğin, araç türlerini veya durumlarını temsil eden bu türler, yanlış veya geçersiz değerlerin kullanılmasını engeller ve programın mantıksal tutarlılığını korur.

* **Vehicle\_Type**: Araç türlerini temsil eder. Değerler: Hava, Kara, Deniz.
* **Operasyonel\_Status**: Araçların operasyonel durumunu belirtir.
* Değerler: Operasyonel, Arizali.

### 2.2. Araç Kaydı

Her araç aşağıdaki özelliklere sahip bir veri kaydı olarak saklanır. Bu veri tiplerinin kullanımı, performans ve kullanım kolaylığı açısından bir dizi fayda sağlar:

1. Unbounded\_String:
   * Esnek uzunluklu metinleri depolamak için idealdir. Sabit uzunluklu dizi kullanmanın getirdiği bellek sınırlamalarını ortadan kaldırır.
   * Büyük metinler için dinamik bellek tahsisi sağlar, bu da hafızanın verimli kullanılmasını mümkün kılar.
2. Vehicle\_Type ve Operasyonel\_Status:
   * Enum türleri, sabit bir değer kümesi sunar. Bu, yanlış girişlerin önüne geçerek tür güvenliği sağlar.
   * Kodun okunabilirliğini artırır ve hata ayıklama süreçlerini kolaylaştırır.
3. Vehicle\_List:
   * Statik bir dizi yapısıdır, sabit boyutlu bir liste sağlayarak bellek yönetimini basitleştirir.
   * Dizi kullanımı, araçlara doğrudan indeksleme ile hızlı erişim sağlar.



* + Name: Araç adını temsil eder ve esnek uzunluklu metin için Unbounded\_String tipi kullanılmıştır.
  + VehicleType: Araç türünü belirtir.
  + Status: Araç durumunu belirtir.

### 2.3. Araç Listesi

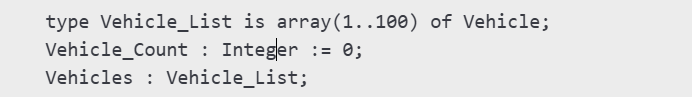
100 adet araç kaydını saklayabilen sabit uzunlukta bir dizi tanımlanmıştır:

type Vehicle\_List is array(1..100) of Vehicle;

Vehicle\_Count : Integer := 0;

Vehicles : Vehicle\_List;

* Vehicle\_List: Araçların toplu olarak saklandığı dizidir.
* Vehicle\_Count: Listeye eklenmiş olan toplam araç sayısını tutar.



* Vehicle\_List: Araçların toplu olarak saklandığı dizidir.
* Vehicle\_Count: Listeye eklenmiş olan toplam araç sayısını tutar.

## 3. Ana Fonksiyonlar ve Mantıklar

### 3.1. Araç Ekleme (Add\_Vehicle)

Bu prosedür, kullanıcı tarafından girilen bilgileri alır ve hem bellek üzerindeki araç listesine hem de XML dosyasına yazar.

#### 3.1.1. Adımlar

1. Boş Yer Kontrolü: Araç listesinin dolup dolmadığı kontrol edilir. Maksimum 100 araç kaydedilebilir.
2. Listeye Ekleme: Yeni araç bilgileri diziye eklenir.
3. XML Dosyasına Yazma:
   * Dosya mevcut değilse, yeni bir dosya oluşturulur.
   * Mevcut dosyaya ekleme yapılır.
4. Veri Formatlama: XML formatında araç bilgileri eklenir.

#### 3.1.2. Kullanılan Yapılar

* Dosya İşlemleri: Ada.Text\_IO kütüphanesi dosya oluşturma ve yazma işlemleri için kullanılmıştır.
* Tip Özelleştirme: To\_String fonksiyonu, esnek uzunluklu metni (Unbounded\_String) XML dosyasına yazılabilir bir dizeye dönüştürmek için kullanılmıştır.

#### 3.2. Araç Listeleme (Print\_Vehicles)

Bu prosedür, mevcut araçları ekranda görüntülemek için tasarlanmıştır.

#### 3.2.1. Adımlar

1. Liste Boşluğu Kontrolü: Araç listesinin boş olup olmadığı kontrol edilir. Liste boşsa kullanıcıya bir mesaj gösterilir.
2. Filtreleme: Kullanıcının belirttiği tür ve durum kriterlerine uygun olan araçlar listeye dahil edilir.
3. Bilgi Yazımı: Araç bilgileri uygun formatta ekrana yazılır.

#### 3.2. Araç Güncelleme (Update\_Vehicle\_Status)

Bu prosedür, daha önce eklenmiş aracın adı alınarak xml dosyasına yer alan durumu ile kullanıcıdan alınan yeni durum bilgisiyle güncelleme işlemi yapılmış olunur.

#### 3.2.1 Adımlar

1. Giriş Kontrolü: Kullanıcıdan alınan araç ismini xml dosyasında kontrol etme ve bu sırada araç adının uzunluk kontrolü yapılır.
2. Araç Güncelleme: Giriş kontrolünün başarılı geçmesinin ardından kullanıcıya yeni durum bilgisi sorulur, alınan yeni durum ile birlikte araç adının yeni durumu xml dosyada ilgili yere yazılır.

#### 3.4. Kullanıcı Arayüzü (UserInformations)

Bu prosedür, kullanıcının seçimler yapabileceği bir menü sistemi sağlar. Öne çıkan özellikler:

#### 3.4.1. Seçenekler

* 1. Araç Ekle: Kullanıcıdan araç adı, türü ve durumu alınır.
* 2. Araç Listele: Tüm mevcut araçları ekranda görüntüler.
* 3. Araç Filtrele: Belirli kriterlere uygun araçları listelemek için kullanılır.
* 4.Araç Güncelle: Araç adı ile xml dosyasında araç bulunur ve kullanıcıdan yeni durum ile güncelleme işlemi yapılır.
* 5. Çıkış: Programdan çıkılır.

#### 3.4.2. Kullanılan Teknikler

* Girdi Doğrulama: Kullanıcıdan alınan veriler kapsam dışındaysa hata mesajları ile bilgilendirilir.
* Tip Dönüşümü: Kullanıcı girdiğini uygun tiplere (enum) dönüştürmek için 'Val fonksiyonu kullanılmıştır.

## 4. XML Dosyası Formatı

Kayıt edilen araç bilgileri XML formatında saklanır. Dosya yapısı aşağıdaki gibidir:

### 

### XML Formatının Kullanım Nedeni

* Veri Organizasyonu: XML, veri yapısını hiyerarşik bir şekilde organize etmeye imkan tanır.
* Platform Bağımsızlığı: Farklı sistemlerde okunabilir ve kolayca entegre edilebilir.
* Ada.XML kütüphanesinin gereksiz karmaşıklığı: Eğer proje küçük çaplı bir sistem ise, Ada.XML gibi bir kütüphanenin kullanılması aşırı yük ve gereksiz karmaşıklık getirebilir. Basit bir XML işleme mantığı, bu tür bir uygulama için yeterli olabilir.
* Kütüphane Bağımlılığından Kaçınma: Proje, minimum harici bağımlılık gereksinimiyle çalıştırılmak istenmiş olabilir. Bu da sistemin taşınabilirliğini artırır ve farklı sistemlere uyum sağlar.
* Öğrenme Amaçlı Yaklaşım: Veri yapılarını ve dosya yönetimini daha iyi anlamak için yararlı bir yöntemdir. Kütüphane kullanmadan bu işlemleri gerçekleştirmek, programcıya XML formatını, dosya okuma/yazma işlemlerini ve hata yönetimini anlamada derinlik kazandırır.

## 5. Sonuç

Askeri Araç Sistemi, basit ve etkili bir yapı kullanarak askeri araçların kaydını, güncellenmesini ve filtrelemesini sağlamaktadır. Projede Ada dilinin özelliklerinden faydalanılarak tip güvenliği, hata kontrolü ve esneklik ön planda tutulmuştur. Verilerin saklanması için xml dosyasını kendi etiketleri ile oluşturulup her araç ekleme işleminde dosya açma/kapama işlemiyle içine veri saklama olanağı sunulmuştur.