



**T.C.
DÜZCE ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

GOLANG PROGRAMMING LANGUAGE OVERVIEW

KÜBRA KILIÇ

LİSANS BİTİRME TEZİ

**DANIŞMAN
PROF. DR. İBRAHİM YÜCEDAĞ**

DÜZCE, 2020

T.C.
DÜZCE ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GOLANG PROGRAMMING LANGUAGE OVERVIEW

Kübra Kılıç tarafından hazırlanan bitirme tezi çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü'nde **LİSANS BİTİRME TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. İbrahim Yücedağ

Düzce Üniversitesi

Jüri Üyeleri

Prof. Dr. İbrahim Yücedağ

Düzce Üniversitesi

Tez Savunma Tarihi:

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

10 Eylül 2020



Kübra Kılıç

TEŞEKKÜR

Lisans öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Prof. Dr. İbrahim Yücedağ'a en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

10 Eylül 2020

Kübra Kılıç

İÇİNDEKİLER

ŞEKİL LİSTESİ.....	VIII
KISALTMALAR.....	X
SİMGELER	XI
ÖZET	XII
ABSTRACT	XIII
1. GİRİŞ.....	1
2. MATERYAL VE YÖNTEM.....	2
2.1 İLK GO UYGULAMASI	4
2.2 GO PROGRAMLAMA DİLİ ÖZELLİKLERİ.....	5
3. VERİ TİPLERİ	6
3.1 SAYISAL VERİ TİPLERİ(İNT).....	6
3.2 FLOAT VERİ TİPLERİ	7
3.3 SABİT DEĞİŞKENLERİN TANIMLANMASI	7
3.4 STRING VERİ TİPLERİ.....	8
3.5 BOOLEAN VERİ TİPLERİ	8
4. OPERATÖRLER.....	10
4.1 ARİTMETİKSEL OPERATÖRLER	10
4.2 KARŞILAŞTIRMA OPERATÖRLERİ	11
4.3 MANTIKSAL OPERATÖRLER	11
4.4 ATAMA OPERATÖRLERİ	12
5. TÜR DÖNÜŞÜMÜ VE PROGRAM AKIŞ DEĞİMLERİ	13
5.1 TÜR DÖNÜŞÜMÜ	13
5.2 PROGRAM AKIŞ DEĞİMLERİ.....	14
5.2.1 Döngüler	14
5.2.2 İf Karar Değimi.....	15
5.2.3 Swich Karar Değimi	16
5.2.4 Defer	17
5.2.5 Fonksiyonlar(Functions)	18
5.2.5.1 Recursive Fonksiyonlar	19

6. GRUPLAMA	20
6.1 DİZİLER(ARRAYS)	20
6.2 DİLİMLER(SLİCES).....	20
6.2.1 Boş Dilimler(Nil Slices).....	22
6.3 HARİTALAR(MAPS)	22
6.4 PAKETLER	23
6.5 YAPILAR	24
6.6 INTERFACE(ARAYÜZ)	26
7. EŞ ZAMANLILIK VE VERİTABANI İŞLEMLERİ	28
7.1 EŞ ZAMANLILIK(CONCURRENCY)	28
7.1.1 GOROUTİNE.....	28
7.1.2 CHANNELS(KANALLAR).....	29
7.2 VERİTABANI İŞLEMLERİ	29
8. DOSYALAMA İŞLEMLERİ.....	32
8.1 DOSYA OLUŞTURMA	32
8.2 DOSYA BİLGİSİ ALMA	32
8.3 DOSYAYI YENİDEN İSİMLENDİRME VE TAŞIMA	33
8.4 DOSYA AÇMAK VE KAPATMAK.....	34
8.5 DOSYA YAZMA İŞLEMİ.....	35
8.6 DOSYA SİLME İŞLEMİ	35
9. GO'NUN SEKTÖRDEKİ YERİ VE WEB PROGRAMLAMA ...	37
9.1 GOLANG VE SEKTÖR	37
9.2 GOLANG İLE WEB PROJESİ GELİŞTİRME.....	37
9.2.1 Paket inceleme:net/http.....	39
9.3 GOLANG BİLGİ SİTESİ	39
10. GOLANG VS DİĞER DİLLERİ.....	43
10.1 GOLANG VS C++	43
10.2 GOLANG VS PYTHON	43
10.3 GOLANG VS JAVA	44
10.4 GOLANG VS PHP	44
10.5 GOLANG VS SWİFT.....	44
11. SONUÇLAR VE ÖNERİLER	45

KAYNAKLAR.....	46
ÖZGEÇMİŞ.....	47

ŞEKİL LİSTESİ

	<u>Sayfa No</u>
Şekil 1 Eklentiler Sekmesi.....	3
Şekil 2 İnstall İşlemi.....	4
Şekil 3 İlk Uygulama.....	4
Şekil 4 Yorum Satırı.....	5
Şekil 5 Tanımlayıcı Örnek.....	5
Şekil 6 Tanımlama İşlemi.....	7
Şekil 7 Sabit Değişken.....	8
Şekil 8 String Veri Tipi.....	8
Şekil 9 Boolean Veri Tipi.....	9
Şekil 10 Tür Dönüşümü Örnek.....	13
Şekil 11 Strconv Kullanımı.....	14
Şekil 12 For Döngüsü.....	15
Şekil 13 İf-Else.....	16
Şekil 14 Swich-case.....	17
Şekil 15 Defer.....	18
Şekil 16 Fonksiyonlar Örnek-1.....	18
Şekil 17 Recursive Fonksiyonlar.....	19
Şekil 18 Diziler.....	20
Şekil 19 Slices.....	21
Şekil 20 Slices-2.....	21
Şekil 21 Nil Slices.....	22
Şekil 22 Maps.....	22
Şekil 23 Range Kullanımı.....	23
Şekil 24 Paketler.....	24
Şekil 25 Yapılar ve Metotlar.....	26
Şekil 26 Yapıcı Metot.....	26
Şekil 27 Golang'ın desteklemediği kullanım.....	26
Şekil 28 İnterface Örnek.....	27
Şekil 29 Goroutine.....	28
Şekil 30 Channels.....	29
Şekil 31 MySql.....	30
Şekil 32 Veri Ekleme.....	30
Şekil 33 Veri Güncelleme.....	31
Şekil 34 Veri Silme.....	31
Şekil 35 Dosya Oluşturma.....	32
Şekil 36 Dosya Bilgisi Alma.....	33
Şekil 37 Taşımadan Önce.....	33
Şekil 38 Taşımadan Sonra.....	34
Şekil 39 Dosya Açma ve Kapatma.....	34
Şekil 40 Dosya Yazma.....	35
Şekil 41 Dosya Yazma-2.....	35
Şekil 42 Dosya Silme.....	36
Şekil 43 Dosya Silme sonra.....	36
Şekil 44 HTML Örnek.....	38
Şekil 45 HTML Örnek Devam.....	38

Şekil 46 Localhost Çıktısı.....	38
Şekil 47 Çağırdığımız Adress.....	39
Şekil 48 Anasayfamız.	40
Şekil 49 Sayfa Örnek	41
Şekil 50 Sayfa Örnek-2.....	41
Şekil 51 Sayfa Örnek-3.....	42

KISALTMALAR

Go
VsCode
TIOBE

Golang Programlama Dili
Visual Studio Code
Programlama Topluluk Endeksi

SİMGELER

int	Tam Sayı Veri Tipi
string	Metinsel İfadeler için veri tipi
Bool	Boolean Veri Tipi

ÖZET

GOLANG PROGRAMMING LANGUAGE OVERVIEW

Kübra Kılıç
Düzce Üniversitesi
Teknoloji Fakültesi
Bilgisayar Mühendisliği Bölümü
Lisans Bitirme Tezi
Danışman: Prof. Dr. İbrahim Yücedağ
Eylül 2020, 58 sayfa

Günümüzde teknolojinin gelişmesiyle programlama dillerine olan ilgi gün geçtikçe artmaktadır. Bu nedenlerden dolayı programlama dili seçimi de büyük önem kazanmıştır. Doğru programlama dili seçimi hem hızlı hem de yüksek performans göstermesi ile üzerinde durulan projenin daha kısa sürede ve yüksek verim ile tamamlanması sağlar. Günümüzde kullanılan Java, C, Python, C++ gibi dillerin yanı sıra yeni bir dil olan Golang programlama dili zamanla gelişip üst sıralarda yer almaktadır. Hemen hemen her uygulama, hızlı, öğrenilmesi kolay ve geniş bir kütüphaneye sahip bir dil olan Go programlama dili ile yazılabilir. Go programlama dili geleneksel dillerde var olan eksikliklerin birçoğunu ortadan kaldırmıştır. Ülkemizde henüz istenilen başarıya ulaşmasa da son yıllarda kullanım alanlarının genişlemesi ve kullanıcı sayılarının gittikçe artmasıyla yükselişe geçen dillerdir. Go, kullanımı kolay, güvenli ve performanslı bir web sunucusuyla birlikte gelir ve kendi web şablon kitaplığını içerir aslında web uygulama geliştirme isteyenler için en doğru seçimdir. Bu çalışmanın amacı güçlü, hızlı, öğrenilmesi kolay bir programlama dili olan Go programlama dili hakkında bilgi vermek ve temel kavramları hakkında açıklamalarda bulunmak ve son olarak Web projesi ile Golang dilinin avantajlı tarafını anlatmaktır.

Anahtar sözcükler: Golang, Go Programlama, Web Programlama

ABSTRACT

GOLANG PROGRAMMING LANGUAGE OVERVIEW

Kübra Kılıç

Düzce University

Technology Faculty, Department of Computer Engineering

Undergraduate Graduation Thesis

Supervisor: Prof. Dr. İbrahim Yücedağ

September 2020, 58 pages

Nowadays, with the development of technology, interest in programming languages is increasing day by day. For these reasons, the choice of programming language has gained great importance. Choosing the right programming language allows the project to be completed in a shorter time and with high efficiency, with its fast and high performance. In addition to languages such as Java, C, Python, C ++, which are used today, the Golang programming language, which is a new language, has developed over time and ranks at the top. Almost any application can be written in the Go programming language, a language that is fast, easy to learn, and has a large library. The Go programming language eliminates many of the shortcomings of traditional languages. Although they have not yet achieved the desired success in our country, they are on the rise with the widening of usage areas and the increasing number of users in recent years. Go comes with an easy-to-use, secure and powerful web server and includes its own web template library, which is actually the right choice for those who want web application development. The purpose of this study is to give information about the Go programming language, which is a powerful, fast, easy-to-learn programming language, and to explain its basic concepts and finally to explain the advantageous side of the Golang language with the Web project.

Keywords: Golang, Go Programming, Web Programming

1. GİRİŞ

Golang diğ er bir ismiyle Go dili, programlama verimliliğini arttırmak ve geliřtiricilerin yařadığı problemlerden yola çıkılarak ortaya  ıkarılmıştır. Google tarafından geliřtirilen bu dil statik olarak yazılmasının yanında bellek y netimini kendisi saėlayan ve ertelenmiř fonksiyonları destekleyen bir   p toplama sistemine sahiptir. Golang resmi web sitesinde (<https://golang.org/>) bu dil i in “Go basit, g venilir ve verimli yazılım oluřturmayı kolaylařtıran a ık kaynaklı bir programlama dilidir” ifadeleri kullanılmıştır. Golang ilk zamanlar Google’ın arka plandaki sistem performansını arttırmak amacıyla geliřtirilmiř bir sistem programlama dili iken zamanla az kaynak t ketmesi, web uygulamalarının kendi kendini host ediyor olması ve y ksek performans  zellikleri sayesinde Web programlayıcıları tarafından dikkat  ekmeye bařlamıřtır [1]. Golang 2007 yılında Google ekibinden Robert Griesemer, Rob Pike ve Ken Thompson tarafından geliřtirilen programlama diliydi fakat ilk kez Kasım 2009 a ık kaynak olarak piyasaya tanıtıldı. İlk s r m 1.0, Mart 2012’de yayınlandı řuan kullanılan son s r m ise 2020 yılında g ncellenmiř olan 1.15’tir. Neden yeni bir dil  ıktığı kısmına deėinecek olursak bazı kaynaklarda dili geliřtiren tasarımcıların k kl  dillerin(C++,Java, python vb.) b y k projelerde performans, derleme gibi sorunlarından sıkılıp yeni bir geliřtirmek i in motive olup bunu bařardıkları yazmaktadır. Bu nedenle Golang’a eklenen her  zellik yılların verdiėi deneyimlerle eklenmiř olup herkesin anlayabileceėi tarzda geliřtirilmektedir. Golang diline  zel olarak geliřtirilmiř maskot Gopher ayırt edici  zelliklerinden biridir. Go Gopher daha  nceden baėıř toplama etkinliėi i in bir projede tasarımcısı Renee French tarafından yayınlanmıřtır. Go projesine bařlanıldığı zamanda Renee French tasarımcısı olmayı istemiřtir ve Gopher’ı tasarlamıřtır. Sincaba benzeyen bu maskotun adı “Go Gopher” olarak kalmıřtır. Daha sonradan mavi renge boyanmıřtır ve kullanılmaya bařlanmıřtır. Bu y zden Golang yazılımcılarına Gopher denmektedir.

2. MATERYAL VE YÖNTEM

Dille ilgili çalışmalar yapmadan önce dil hakkında bilgiler edinilmelidir. Bu araştırma öncelikle Go dilinin temel özellikleri belirtilmiş, veri yapıları üzerinde durulmuş, uygulamalar geliştirilmiş (Kodlanmış) ve derlenerek nasıl bir sonuç ürettiği ortaya çıkartılmıştır. Son olarak görsel bir web ara yüzü ile dilin anlaşılması kolay gele getirilmiştir.

&Dilin Özellikleri

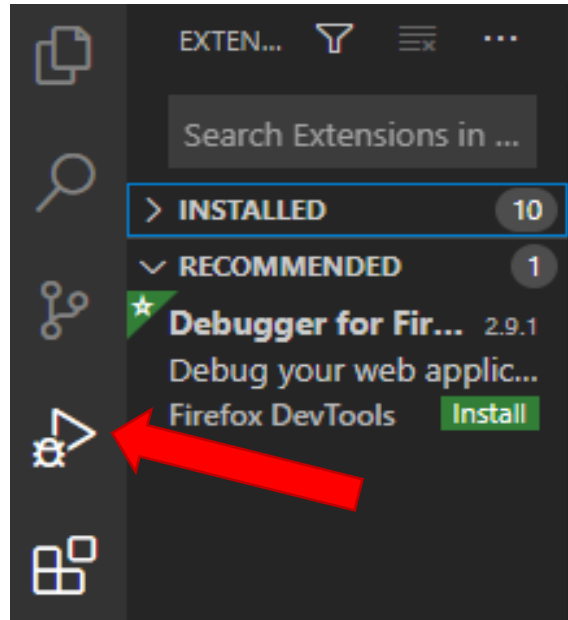
Golang dilinin bazı özellikleri aşağıda sıralanmıştır.

- Açık kaynak kodlu dildir.
- Derleme süresi hızlıdır.
- Modern bir dildir.
- Modern teknolojinin ihtiyaçları göz önünde bulundurularak tasarlandı.
- Yüksek performans ve hızlı gelişebilme özelliği ile Java ile kıyaslanabilecek seviyededir.
- Çoğu iş sadece standart kütüphaneler ile yapılabildiği için dil yapısı basittir.
- Eş zamanlı programlamada Go basit fakat performansı yüksek imkânlar sunmaktadır.
- Eşzamanlılık(Birden fazla işlemin eşzamanlı ve etkili bir şekilde çalışmasını sağlar) özelliği vardır.
- Zengin bir standart kitaplık sağlar.
- Çöp toplama özelliği vardır.
- Arayüzü ve tip yerleştirmeyi doğrular.
- Yazım biçimi C diline benzer.
- Statik tipli bir dildir.
- Uzak paket yöneticisi mevcuttur.
- Ağ ve çoklu işlemleri destekler.
- Değişken tanımında tür belirtimi isteğe bağlıdır.
- Büyük sistemlere ölçeklenebilme özelliği vardır.
- Mevcut toplam 25 anahtar kelimesi vardır. Bunlar: break, default, func, interface, select, case, defer, go, map, struct, chan, else, goto, package, switch, const, fallthrough, if, range, type, continue, for, import, return, var.

Bu dilin eksik yönleri ise aşağıda sıralanmıştır.

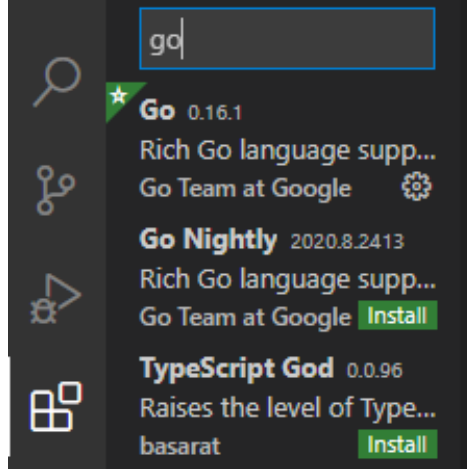
- Go, geleneksel anlamda çok nesne odaklı değildir.
- Metot ve operatör için aşırı yükleme (Overloading) yoktur.
- Jenerik (Generic) programlama desteği yoktur.

Bilgisayar üzerinde Go ile çalışmak istiyorsanız öncelikle bilgisayarınıza Golang programını indirmeniz gerekiyor. Kurulumu oldukça basittir next diyerek kolayca kurulabilir. Kurulum gerçekleştikten sonra kodlarımızı yazacağımız bir IDE(Tümleşik Geliştirme Ortamı) gereklidir. IDE'ler kodlarımızı yazarken kodların doğruluğunu kontrol etmemizde yardımcı bir rol oynarlar. Bu çalışmada Visual Studio Code programı kullanılacaktır siz farklı bir IDE seçebilirsiniz. Visual Studio Code kurulu değilse kurulduktan sonra içerisine Golang eklentisinin kurulması gerekir. Eklenti için öncelikle Visual Studio Code programında pencerenin sol tarafında eklentiler sekmesine basılır.



Şekil 1: Eklentiler Sekmesi

Arama yerine “go” yazıyoruz ve çıkan Şekil 2’deki eklentiye Install butonuna basarak yüklüyoruz. Ben daha önceden yükleme yaptığım için Install butonu gözükmeyecektir. İndirdikten sonra programı kapatıp açtıktan sonra hazır hala gelecektir. Geliştirdğimiz web sitesinde detaylı kurulum aşamaları mevcuttur ordan bu adımlar takip edilerek kurulum gerçekleştirilebilir.



Şekil 2: Install İşlemi

2.1 İlk Go Uygulaması

Programlama dünyasında bir dili öğrenirken gelenekselleşmiş bir durum haline gelen “Hello World” çıktısı yazılır. İlk uygulamamızı yazarken izleyecemiz adımlar öncelikle Visual Studio Code programını açıp dosyasımıza isim verdikten sonra .go uzantılı şekilde kaydetmek olmalıdır. Örneğin “main.go” şeklinde yazılabilir. Daha sonra her kaynak dosyamızın başında bir paket belirtmek zorundayız. Main (package main) özel bir paket ismidir. Diğer programlama dillerinde olduğu gibi import ile kullanılacak kütüphaneler belirtilir. FMT, Go'nun ana kütüphanesinden bir pakettir. Dikkat edilmesi gereken nokta paketlerimizin tanımlama şartının aranmasıdır. Programı çalıştırmak için terminalde GO RUN MAIN.GO yazmamız yeterlidir terminal kısmına ise sol menüden .go uzantılı dosyanıza sağ tıklayarak "Open in Integrated Terminal" kısmına basmanız yeterlidir.

```
main.go X
c: > Users > Kübra > Desktop > golang > İsimlendirme > -go main.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6
7     fmt.Println("Hello World")
8 }
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Kübra\Desktop\golang\İsimlendirme> go run main.go
Hello World
```

Şekil 3: İlk Uygulama

2.2 Go Programlama Dili Kuralları

Her dilde olduğu gibi Golang programlama dilinde de uyulması gereken temel kurallar vardır. Diğer programlama dillerinde olan satır sonunu noktalı virgül(;) ile bitirme olayı Go dilinde yoktur. Satır sonlarında herhangi bir işaret konmaz. Golang dilinde açıklama(yorum) satırları “//” çift taksim veya /* */ taksim yıldız yöntemi ile yapılır.

```
package main

func main() {
    //Tek satır açıklama
    /* birden
    fazla
        açıklama
        gelebilir */
}
```

Şekil 4:Yorum Satırı

Golang dilinde tanımlayıcılar mutlaka bir harf ile başlamalıdır. Noktalama işareti barındırmamalıdır. Golang dili büyük ve küçük harf duyarlıdır. Küçük harf ile tanımladığınız “sayi” kelimesini “Sayi” ile çağıramazsınız. Bir diğer dikkat edilmesi gereken noktada Golang dilinde tanımladığınız bir değişkeni mutlaka kullanmak zorundasınız aksi takdirde hata ile zaten bunu size bildirecektir. Şekil 5’de altını kırmızı çizerek bunu belirtmiştir hâlbuki yanlış olan bir tanımlama yok. Bunun yapılmasının sebebi ise Golang kullanılmayacaksa tanımlanmasının önüne geçmek istemiştir. Ufak projelerde eksiklik gibi görünse de büyük projelerde çok avantajdır.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var sayi int = 10
7     var sayi2 int = 11
8     fmt.Print(sayi)
9 }
10
```

Şekil 5: Tanımlayıcı Örnek

3. VERİ TİPLERİ

Program kodlarken kullandığımız değişken ve fonksiyonların hangi türde olduğu ve bilgisayarımızın belleğinde ne kadar yer kapladığını göstermek için veri tipi kavramını kullanırız. Her programlama dillerinin veri tipleri olduğu gibi Go programlama dilinin de mevcuttur. Var ifadesi ise değişken (variable) tanımlamak için kullanılır. Veri tiplerini sınıflar halinde inceleyelim.

3.1 SAYISAL VERİ TİPLERİ (İNT)

Go programlama dilinde, hem işaretli hem de işaretsiz tam sayılar, dört farklı şekilde gösterilir. İşaretli sayılar int ile temsil edilirken işaretsiz tamsayılar uint ile temsil edilir. Aşağıdaki tabloda açıklanmıştır.

İNT8	8 bitlik işaretli tamsayı
İNT16	16 bitlik işaretli tamsayı
İNT32	32 bitlik işaretli tamsayı
İNT64	64 bitlik işaretli tamsayı
UİNT8	8 bitlik işaretsiz tamsayı
UİNT16	16 bitlik işaretsiz tamsayı
UİNT32	32 bitlik işaretsiz tamsayı
UİNT64	64 bit işaretsiz tamsayı
İNT	Hem int hem de uint için kullanılır, 32 veya 64 bit olarak aynı boyutu içerir.
UİNT	Hem int hem de uint için kullanılır, 32 veya 64 bit olarak aynı boyutu içerir.
RUNE	İnt32 ile aynıdır ve Unicode kod noktalarını temsil eder.
BAYT	İnt8 aynıdır.
UİNTPTR	İşaretsiz bir tamsayı türüdür. Genişliği tanımlanmamıştır.

Değişken tanımlamak için ise şekil 6'daki gibidir.

```
//tanımlama işlemi --var değişken_adı değişken_tipi-- şeklindedir
var sayi int
//veya ilke değer ataması ile tanımlanır.
a := true // bool
b := "Istanbul" // string
c := -13 // int (int8, int16, int32, int64)
d := 2020 // uint (uint8, uint16, uint32, uint64)
e := 3.14 // float32 (float64)
```

Şekil 6:Tanımlama İşlemi

Go dilinde UTF-8 desteği kendiliğinde gelmektedir. String işlemlerde Türkçe karakterler de (ğ, ş, ı, İ, ö, ü) dâhil olmak üzere tüm UTF-8 karakterler sorunsuz şekilde kullanılabilir [2].

3.2 FLOAT VERİ TİPLERİ

Float veri tipleri int veri tiplerinden farklı olarak virgüllü sayıları tutarlar. Şekil 6'daki gibi kullanımında farklılık yoktur sadece küsuratlı bir değer ataması yapacak isek float tipinde tanımlamamız gerekir. Sadece yazılma işleminde dikkat edilmesi gereken nokta float sayılar virgül ile değil de nokta ile tanımlanır. Örneğin virgül (3,14) ile ifade değil nokta (3.14) ile yazılmalıdır.

3.3 SABİT DEĞİŞKENLERİN TANIMLANMASI

Tanımlamasını yaptığının bir değişkenin program boyunca değişmesini istemiyorsanız sabit(const) tipinde tanımlama yapılmalıdır. Const “:=” şeklinde kullanılamaz Pi sayısı gibi ifadeleri kullanmak istiyorsanız const ile tanımlamak işinizi kolaylaştıracaktır. Const ile tanımlanmış veri tipleri hem bellekte daha az yer kaplamakta hem de kodların okunmasını kolaylaştırmaktadır.

```
func main() {
    const pi = 3.14
    var sayı float64 = 11.2
    fmt.Println(sayı + pi) //ÇIKTI: 14.34
    pi := 11
}
```

Şekil 7:Sabit Değişken

Şekil 7’de aslında hatasının sebebi const işlemini tanımladıktan sonra bir daha onu değiştirmek için değer ataması yapamıyor oluşundan kaynaklanır.

3.4 STRING VERİ TİPİ

String veri tipleri içerisinde metinsel yani sözel ifadeler barındırırlar. Şekil 6’da b değişkenine ilk değer ataması yapılmıştır. Bunu belirtmesek de Golang içerisinde String ifade girildiği için onun türünü zaten String olarak tanımlamıştır.

```
var mesaj string
mesaj = "Merhaba! "
fmt.Println(mesaj)

//Başka bir yolu
/*
    var mesaj string = "Selam go! "
    fmt.Println(mesaj)*/

//String tanımlamasak da olur
/*var mesaj = "Merhaba !"
    fmt.Println(mesaj)
*/
}
```

Şekil 8: String Veri Tipi

3.5 BOOLEAN VERİ TİPİ

Boolean veri tipleri bir durumun var olması durumunda “true” olmaması halinde ise “false” değerini döndüreceklerdir. Şekil 6’da a değişkenine Boolean türünde atama yapılmıştır. İlerde yapılacak işlemlerde kontroller ile false veya true döndüreceklerdir.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     saat := 10
7     var acikMi = saat < 24
8     fmt.Println("Avm şuan açık mı? :", acikMi)
9 }
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Avm şuan açık mı? : true

Şekil 9: Boolean Veri Tipi

Kısacası Golang dilinde tam sayı değişkenlerini “int”, metinsel değişkenleri “String”, küsuratlı değişkenler de “float”, sabit değişkenlerde “const”, mantıksal değerler için “Boolean” kullanılmaktadır. Golang dilinde türü belirtmek zorunda olmadığınız için ilk değer ataması yaptığınızda aslında fark etmemiş olsanız da Golang kendisi zaten bu işlemi gerçekleştiriyor olacaktır. Ayrıca Go programlama dilinde diğer diller de olan private, public, protected gibi erişim belirleyiciler yoktur. Değişken ismi büyük harf ile başlıyor ise bu değişken tanımlı olduğu paket dışından da erişilebilir, küçük harf ile başlıyorsa sadece tanımlı olduğu paket içerisinde erişilebilir demektir.

4. OPERATÖRLER

Programlama dillerinde işlem yapmamızı sağlayan işaretlere operatör denir. Operatörlerde işlem yaparken işlem önceliği sırasına dikkat edilmesi gerekmektedir [3]. Golang dilinde operatörleri 4 başlık altında toplayabiliriz.

4.1 ARİTMETİKSEL OPERATÖRLER

Aritmetiksel operatörler matematiksel işlemler yapmamızı sağlar. Sözel veya sayısal bir ifadeye değer atamak için “=” (atama) operatörü kullanılır. Atama operatörünün matematiksel operatörler ile birleşmesiyle yeni operatörler oluşturabilir (“+=”, “-=”, “*=” vb).

Operatör Adı	Ne iş yaptığı	Örnek
+	Toplama İşlemi yapar	1+2=3
-	Çıkarma İşlemi yapar	3-2=1
/	Bölme İşlemi yapar	10/2=5
*	Çarpma İşlemi yapar	10*2=20
%	Mod alma işlemi yapar	10%3=1
++	1 arttırır	10++ //11
--	1 azaltır	10-- //9

4.2 KARŞILAŞTIRMA OPERATÖRLERİ

Karşılaştırma operatörleri iki verinin karşılaştırılması işleminde kullanılır. Karşılaştırma sonucunda sonuç doğru ise “true” yanlış ise “false” değerlerini döndürürler.

Operatör Adı	Ne iş yaptığı	Örnek
==	Karşılaştırılan veriler eşit ise true verir	9==4 false değerini döndürür
!=	Karşılaştırılan veriler eşit değil ise true verir	5!=133 true değerini döndürür
>	İlk veri ikinci veriden büyükse true verir	15>13 true değerini döndürür
<	İlk veri ikinci veriden küçükse true verir	1<2 true değerini döndürür
>=	İlk veri ikinci veriden büyük veya eşitse true verir.	7>=1 true değerini döndürür 1>=1 true değerini döndürür
<=	İlk veri ikinci veriden küçük veya eşitse true verir	4<=2 false değerini döndürür 13<=13 true değerini döndürür

4.3 MANTIKSAL OPERATÖRLER

Mantıksal operatörler verileri kontrol eder ve kullanılan operatöre göre mantıksal bir değer döndürürler.

Operatör Adı	Ne iş yaptığı	Örnek
&&	VE operatörüdür.	5==5 && 2==3 (false) 5==5 && 3==3 (true)
	VEYA operatörüdür.	2==2 2==3 (true) 2==5 2==3 (false)
!	DEĞİL operatörüdür. Sonucun tersini alır.	!(1==1 && 3==3) (false) !(5==1 && 8==3) (true)

4.4 ATAMA OPERATÖRLERİ

Atama operatörleri değer atamak için kullanılır. Golang dilinde ilk değer ataması da “:=” ile yapılır.

Operatör Adı	Ne iş yaptığı	Örnek
=	Atama yapar	Sayi=10
+=	Değişkeni kendisiyle toplar	Sayi=10 Sayi+=10 //Sonuç:20
-=	Kendinden çıkarır	Sayi=10 Sayi-=2 // Sonuç: 8
=	Kendiyle çarpar	Sayi=10 Sayi=2 // Sonuç: 20
/=	Kendine böler	Sayi=10 Sayi/=2 // Sonuç: 5
%=	Kendine bölümünden kalanı atar	Sayi=10 Sayi-%=2 // Sonuç: 0

5. TÜR DÖNÜŞÜMÜ VE PROGRAM AKIŞ DEĞİMLERİ

5.1 TÜR DÖNÜŞÜMÜ

Tür dönüşüm işlemleri genel olarak tür(değer) şeklinde yapılır. Golang 'da C'den değişik olarak farklı türdeki öğeler arasındaki atama için açık bir dönüştürme işlemi gerektirir. Şekil 10'da float türündeki a değişkenini int türüne dönüştürme işlemi gösterilmiştir.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     a := 11.55
7     b := uint(a)
8     fmt.Println(b)
9 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

11

Şekil 10:Tür Dönüşümü örnek

Fakat Golang bu kullanımı tüm türler arasında izin vermemektedir. İnt olan bir ifadeyi bu şekilde yaptığınızda sonuç karakter olarak değer alır. Bu tarz dönüşümler için “strconv” kütüphanesinden yararlanılmaktadır. Şekil 11’de string türünde fakat değeri 11 olan bir atama yapılmıştır. Bu ifadeyi int yapmak için strconv.Atoi kullanılırken int ifadeyi string yapmak için ise strconv.Itoa kullanılmaktadır.

```
8 func main() {
9
10     //Tür Dönüşümü
11     var yazi = "11"
12     var x = 5
13     var f float64 = 2.0
14
15     fmt.Println(yazi, x, f)
16     //String olan yaziyi int yapalım
17     sayi, _ := strconv.Atoi(yazi)
18     fmt.Println("Sonuç(int): ", sayi)
19
20     //İnt string ile yapma strconv.Itoa(sayi)
21
22     fmt.Println("Sonuç(string): ", strconv.Itoa(sayi))
23 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
11 5 2
Sonuç(int): 11
Sonuç(string): 11
```

Şekil 11: Strconv Kullanımı

5.2 PROGRAM AKIŞ DEĞİMLERİ

5.2.1 Döngüler

Genel olarak programlama dillerinde 3 tane döngü çeşidi vardır. Bunlar while, do-while ve for döngüsüdür. Döngülerin amacı, aynı kodu defalarca yazmaktansa birkaç kez tekrar etmektir. Golang dilinde diğer dillerin aksine sadece for döngüsü vardır. Bu durum bize while ve do while ile yapılanlarını kullanamayacağımız anlamına gelmiyor. Çünkü Golang for ile hepsini yapmamıza izin veriyor. For Döngüsü, komut ya da komutların istenilen sayıda tekrarlanmasını sağlayan ve sayaç mantığıyla çalışan döngüdür. Diğer kullanımların aksine parantez kullanılmadan yazılır. Kullanımı "for döngü değişkeni=başlangıç değeri; şart; değişim miktarı" dır. Şekil 12'de 1 ile 10 arasındaki sayıları yazdırma işlemi yapılmıştır.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     //For Döngüsü
7     var i int
8     for i = 1; i < 10; i++ {
9         fmt.Println("i değeri ", i)
10    }
11 }
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Kübra\Desktop\golang\for while> go run main.go
i değeri 1
i değeri 2
i değeri 3
i değeri 4
i değeri 5
i değeri 6
i değeri 7
i değeri 8
i değeri 9
```

Şekil 12: For Döngüsü

5.2.2 İf Karar Değimi

İf: Eğer, Else: Yoksa anlamına gelir. If-Else program akışını koşullandırmalar için kullanılır. Go dilindeki if bloğunun diğer dillerdeki if bloklarından pek farkı yoktur. Tek farkı gereksiz parantez kullanımı Go da yoktur. Şekil 13’de klavyeden iki sayı aldık ve if ile kontrolünü gerçekleştirdik dikkat edilmesi gereken tek nokta düzenidir. Yani yazım düzeni yanlış olduğu takdirde hata alacaksınız. Yani "} else if" bu şekilde yazılmaz alt alta yazımı doğru değildir. İf içerisinde ilk değer ataması da yapılabilir örneğin "if a:=10;" bu yazım da doğrudur.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     //Klavyeden girilen iki sayıyı kontrol edelim
7     var a int
8     var b int
9     fmt.Println("a değerini giriniz")
10    fmt.Scan(&a)
11    fmt.Println("b değerini giriniz")
12    fmt.Scan(&b)
13    if a < b {
14        println("b büyüktür")
15    } else if a > b {
16        println("a büyüktür")
17    } else {
18        println("İki sayı eşittir")
19    }
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Kübra\Desktop\golang\Döngüler> go run main.go
a değerini giriniz
15
b değerini giriniz
7
a büyüktür
```

Şekil 13:İf-Else

5.2.3 Swich Karar Değimi

Switch Türkçe de anahtar olarak geçer aslında if-else den farkı yoktur. Case deyimi durumu ifade eder. Koşul sağlandığı zaman işleme devam edilmez. Switch'te koşulların gerçekleşmediği zaman işlem uygulamak istiyorsak bunu default terimi ile yaparız. Şekil 14'deki örneğe klavyeden 8 gibi bir rakam girdiğiniz size "Yanlış değer girdiniz.." kısmını döndürecektir.

```
5 func main() {
6     //Swich Case Haftanın günleri
7     var gün int
8     fmt.Println("1-7 arası bir sayı giriniz: ")
9     fmt.Scan(&gün)
10    switch gün {
11    case 1:
12        println("Pazartesi")
13    case 2:
14        println("Salı")
15    case 3:
16        println("Çarşamba")
17    case 4:
18        println("Perşembe")
19    case 5:
20        println("Cuma")
21    case 6:
22        println("Cumartesi")
23    case 7:
24        println("Pazar")
25    default:
26        println("Yanlış değer girdiniz..")
27    }
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1-7 arası bir sayı giriniz:
2
Salı
```

Şekil 14:Swich-Case

5.2.4 Defer

Defer ertelemek anlamında kullanılır. "Defer" kelimesini işlemin başına eklersek o işlemi içerisinde bulunduğu fonksiyonun içindeki işlemlerden sonra çalıştıracağı anlamına gelir. Şekil 15’de normalde ilk derlenmesi gereken 7.satırdır fakat defer ile kullanıldığı için bu satır okunmadan geçilir. 8. Satır ekrana yazılır ve 9.satıra inilir fakat burada da defer olduğu için okunmaz. En son 10.satır yazıldıktan sonra defer’ler ekrana sondan başlanarak yazılır.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     //DEFER
7     defer fmt.Println("Normal şartlarda ilk çıktı")
8     fmt.Println("Fakat ilk derlenen benim")
9     defer fmt.Println("Merhaba")
10    fmt.Println("Normal şartlarda en son derlenen")
11 }
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Fakat ilk derlenen benim
Normal şartlarda en son derlenen
Merhaba
Normal şartlarda ilk çıktı

Şekil 15:Defer

5.2.5 Fonksiyonlar (Functions)

Fonksiyonlar ana programdaki ilerleyişi farklı bir alt programa yönlendiren ve bu sayede birden fazla yerde ihtiyaç olacağını düşündüğünüz işlemleri tekrar tekrar çağırmanızı sağlayan yapılardır. Fonksiyonların artlarından biri düzen içerisinde olduklarından hata bulma/düzeltilme, geliştirme, projeyi genişletme ve sadeleştirme gibi işlemler daha çabuk gerçekleşecektir. Go programlama dilinde fonksiyonlar “func” öneki ile tanımlanır. Diğer bir farklılık ise normal de diğer dillerde önce değişken türünden sonra değişken adı gelirken Go programlama önce değişken adı daha sonra değişken türü gelmektedir. Şekil 16’da parametresiz bir fonksiyon tanımlanmıştır. Daha sonra main bloğunda fonksiyon çağırılmıştır.

```
-go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func deneme() {
6     fmt.Print("Merhaba! ")
7 }
8 func main() {
9     deneme()
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Merhaba!

Şekil 16: Fonksiyon Örnek1

5.2.5.1 Recursive Fonksiyonlar

Recursive fonksiyonlar diğer bir ismiyle özyineli fonksiyonlar, fonksiyonun içinde yine kendisinden bir parça bulunuyorsa yani kendi kendini çağıran fonksiyonlara verilen isimdir. Fonksiyonun her seferinde çağrılması sırasında fonksiyonla birlikte tanımlanmış olan parametrelerin ve diğer yerel (local) değişkenlerin, yığın (stack) üzerinde yeniden oluşturulması ve çağırılan fonksiyonun tekrar çalışmasına neden olur [4].

```
1  package main
2
3  import "fmt"
4
5  func fact(n int) int {
6      if n == 0 {
7          return 1
8      }
9      return n * fact(n-1)
10 }
11 func main() {
12     fmt.Println(fact(5))
13 }
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Kübra\Desktop\golang\özyinemeli Fonks> go run main.go
120

Şekil 17: Recursive Fonksiyonlar

6. GRUPLAMA

6.1 DİZİLER (ARRAYS)

Diziler işlerinde bir veya birden fazla değer tutabilen yapılardır ve dizideki her değer numaralandırılır. Numaralandırma işlemi sıfırdan başlar. Dizi içerisindeki tüm elemanlara aynı isimle ulaşılır ve elemanların isimleri ortaktır. Elemanlar arasındaki fark ise bellekteki yeridir [5].



```
go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func main() {
6     myArray := [4]string{"Ali", "Ayşe", "Murat", "Fatih"}
7     fmt.Println(myArray)
8     fmt.Println(myArray[1])
9 }
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Kübra\Desktop\golang\Diziler> go run main.go
[Ali Ayşe Murat Fatih]
Ayşe
```

Şekil 18: Diziler

6.2 DİLİMLER (SLICES)

Dilimler, dizilerin alt parçacıklarıdır. Kesme işlemi yapıyorlar diyebiliriz. Belirlenen kısımlara erişmek için kullanılan yöntemlerden biridir. Bir dilim iki kısımdan oluşur, düşük ve yüksek sınır olmak üzere iki indeks belirtilerek oluşturulur. Kullanımı: a [düşük: yüksek] şeklindedir [6]. Slices'lar veriyi kopyalamazlar, verinin adreslerini alırlar. Slices'lar herhangi bir veri depolaması da yapmazlar, sadece temeldeki bir dizinin bir bölümünü tanımlarlar.

```
-go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func main() {
6     myArray := [4]string{"Ali", "Ayşe", "Murat", "Fatih"}
7     Slice1 := myArray[1:4]
8     fmt.Println(myArray)
9     fmt.Println(Slice1)
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[Ali Ayşe Murat Fatih]
[Ayşe Murat Fatih]

Şekil 19: Slices

Dilimlerin uzunluk ve kapasiteni hesaplayabiliriz. Dilimin uzunluğunu `len()` fonksiyonu, kapasitesini ise `cap()` fonksiyonu ile hesaplanır. Dilimin uzunluğu, içerdiği elemanların sayısı iken dilimin kapasitesi, dilimdeki ilk öğeden başlayarak dizideki öğelerin sayısıdır.

```
-go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func main() {
6     myArray := [6]string{"Ali", "Ayşe", "Murat", "Fatih", "Emre", "Buse"}
7     Slice1 := myArray[1:5]
8     fmt.Println(myArray)
9     fmt.Println(Slice1)
10    fmt.Println(len(Slice1))
11    fmt.Println(cap(Slice1))
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[Ali Ayşe Murat Fatih Emre Buse]
[Ayşe Murat Fatih Emre]
4
5

Şekil 20: Slices-2

6.2.1 Boş Dilimler (Nil Slices)

Bir dilimin uzunluğu ve kapasitesi 0 ise ve temel dizisi yoksa Boş dilimler denir.

```
go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func main() {
6     var deneme []string
7     if deneme == nil {
8         fmt.Println("Boş Dilim!")
9     }
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Kübra\Desktop\golang\Diziler> go run main.go
Boş Dilim!

Şekil 21: Nil Slices

6.3 HARİTALAR (MAPS)

Haritalar, türünü key-value çiftleri şeklinde elemanlar barındıran bir veri yapısı denebilir. Diğer dillerdeki Hash tablolarına veya sözlüklere (Dictionary) Go'daki built-in karşılığıdır. Map'ler bir anahtar (Key) ve bir değer (Value) değerleri alırlar. Şekil 23'de ise Range ile sıralanmıştır.

```
5 func main() {
6     myMap := make(map[int]string)
7     myMap[06] = "Ankara"
8     myMap[34] = "İstanbul"
9     myMap[81] = "Düzce"
10    myMap[41] = "Kocaeli"
11    myMap[16] = "Bursa"
12    myMap[01] = "Adana"
13    fmt.Println(myMap[01])
14    fmt.Println(myMap[34])
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Kübra\Desktop\golang\Diziler> go run main.go
Adana
İstanbul

Şekil 22: Maps

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     myMap := make(map[int]string)
7     myMap[06] = "Ankara"
8     myMap[34] = "İstanbul"
9     myMap[81] = "Düzce"
10    myMap[41] = "Kocaeli"
11    myMap[16] = "Bursa"
12    myMap[01] = "Adana"
13    fmt.Println(myMap[01])
14    fmt.Println(myMap[34])
15    delete(myMap, 06)
16    fmt.Println("Ankara Silindi")
17    fmt.Println(myMap)
18    for k, v := range myMap {
19        fmt.Println(k, v)
20    }
21 }
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Adana
İstanbul
Ankara Silindi
map[1:Adana 16:Bursa 34:İstanbul 41:Kocaeli 81:Düzce]
81 Düzce
41 Kocaeli
16 Bursa
1 Adana
34 İstanbul
```

Şekil 23: Range kullanımı

6.4 PAKETLER

Her Go programı paketlerden oluşur. Bu aslında C++’da görülen hazır fonksiyonlara benzetilebilir. Programlar main paketinde çalışmaya başlar. Paketler kod organizasyonu sağlarlar, derleyiciyi hızlandırırlar. Örneğim fmt her çalışmada yeniden derlenmez. Çünkü 1 kere derlenmiştir. Şekil 24’de örnekler verilmiştir daha geniş paketlere Golang <https://golang.org/pkg/> adresinde ulaşılabilir.

```
1 package main
2
3 import (
4     "fmt"
5     "math/rand"
6     "strings"
7 )
8
9 func main() {
10
11     //fmt paketi
12     fmt.Println("Hello")
13     //rand paketi
14     fmt.Println("Favori sayım", rand.Intn(10))
15     //string arama varsa true döner
16     fmt.Println(strings.Contains("Kübra", "br"))
17     fmt.Println(strings.Count("Kübraaa", "a"))           //Kübra da kaç tane k var
18     fmt.Println(strings.HasPrefix("Kübra_Kılıç", "Kübra")) //Ön ek
19     fmt.Println(strings.HasSuffix("Kübra_kılıç", "kılıç")) //son ek arama
20     fmt.Println(strings.Index("Kübra", "ü"))             // ü kaçınıcı harf
21
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Hello
Favori sayım 1
true
3
true
true
1
```

Şekil 24: Paketler

6.5 YAPILAR

Veri yapısı içinde birden fazla değişken saklayan bilgilerdir. Aynı nesne için kullanılan verileri sembolize ederken tek bir nesne adı altında toplanırlar. Örneğin öğrenci veritabanı üzerinde çalışan program yazmak istiyorsunuz. Öğrenci için insan (isim, soyisim, yaş vb.) ve ders (kredisi, dersin adı, bölümü vb) bilgilerini içeren tanımlama yapmanız gerekecektir. Bu durumda öğrenci ders ve insan yapılarından oluşacaktır [7]. Golang da sınıf(class) kavramı yoktur yapılar vardır. Ayrıca Golang metot Overloading olayını desteklememektedir.

```
main.go > ...
1  package main
2
3  import (
4      "fmt"
5  )
6
7  type deneme struct {
8      isim    string
9      soyisim string
10     yaş     int
11 }
12
13 func main() {
14
15     a := deneme{isim: "Kübra"}
16     fmt.Println(a.isim)
17     //2.yöntem
18     b := new(deneme)
19     b.isim = "Ayşe"
20     fmt.Println(b.isim)
21     //veriyi okuyalım
22
23 }
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Kübra
Ayşe

Şekil 25: Yapılar ve metotlar

Şekil 25 ‘te gösterildiği örnekte structların genel kullanımı vardır. Başka bir dil ile ilgilenen birisinin aklına dire “class deneme{ }” gelecektir. Aslında type de aynı işlemi görüyor diyebiliriz. Golang dilini farklı kılan özelliklerden biride “int yaş” değil de önce isim daha sonra tür belirtilir yani “yaş int” şeklindedir. İşaretçiler veya pointer’lar var olan değerin bellekteki adresini tutarlar. & işareti değişkenin bellekteki adresini tutarken * (yıldız) işareti tutulan adresteki değeri görüntüler. Şekil 27’de ise yapıcı metot’un isminin değiştirilmesi gerekir sebebi de Golang’ın metot Overloading olayını desteklememesidir.

```

14 //varsayılan ve boş yapıcı metot
15 func yenideneme() *deneme {
16     c := new(deneme)
17     return c
18 }
19
20 func main() {
21
22     a := deneme{isim: "Kübra"}
23     fmt.Println(a.isim)
24     //2.yöntem
25     b := new(deneme)
26     b.isim = "Ayşe"
27     fmt.Println(b.isim)
28     //yapıcı metot kullanımı
29     e := yenideneme()
30     e.yaş = 22
31     fmt.Println(e.yaş)

```

Şekil 26: Yapıcı metot

```

func yenideneme() *deneme {
    c := new(deneme)
    return c
}

// Golang'ın desteklemediği kullanım
func yenideneme(isim, soyisim string, yaş int) *deneme {
}

```

Şekil 27: Golang'ın desteklemediği kullanım

6.6 INTERFACE (ARAYÜZ)

Kalıtım geliştirilen sınıfların genel kullanılabilir olabilmesini, kendi yeteneklerini bir başka sınıfa aktarması işlemidir. Golang direk olarak kalıtımı desteklemez. Arayüz (interface) sınıfta olması gereken metod ve özellikleri tanımlayan yapıdır denebilir [8]. Interface'ler bir uygulama geliştirirken interface'den kalıtım alırsanız interface içerisinde ki metotları, nesneleri implement ettiğiniz struct içerisinde erişilebilir hale getirir. Bu da soyutlama gibi ek faydalar sağlar.

```
interface > -go main.go > ...
1  package main
2
3  import "fmt"
4
5  type inter interface {
6      newinter()
7  }
8
9  type yapı struct {
10     newyapı string
11 }
12
13 func (deneme yapı) newinter() {
14     fmt.Println(deneme.newyapı)
15 }
16
17 func main() {
18     var a inter = yapı{"Kübra"}
19     a.newinter()
20 }
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Kübra

Şekil 28: Interface örnek

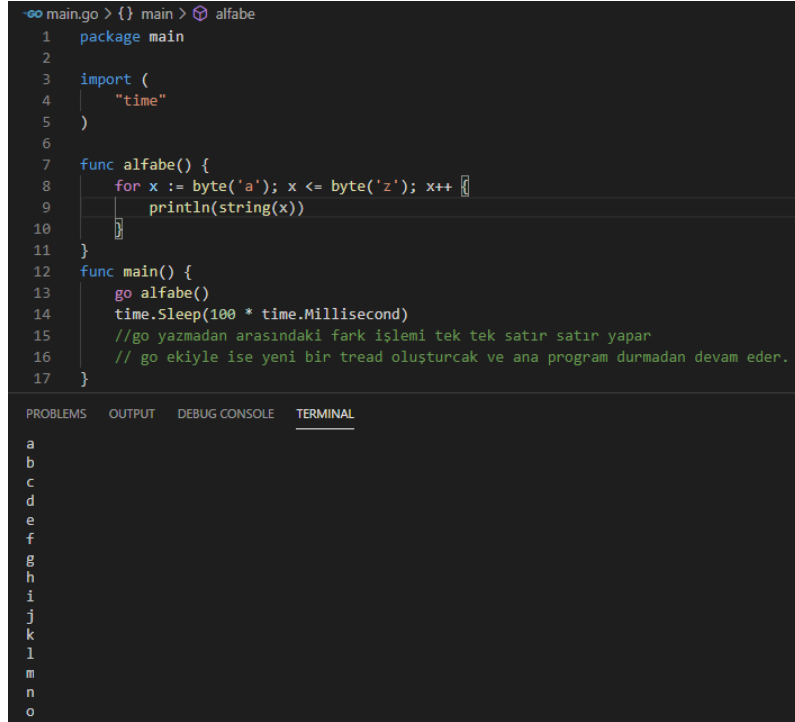
7. EŞ ZAMANLILIK VE VERİTABANI İŞLEMLERİ

7.1 EŞ ZAMANLILIK (CONCURRENCY)

Eş zamanlılık yazdığınız programın parçalarının aynı anda eş zamanlı olarak çalıştırması denebilir [9]. Bir web sitesinde kullanıcıların birbirini beklemeden siteye erişim sağlamaları örnek olarak verilebilir [10]. Eş zamanlılık ve paralellik kavramları birbirinden farklı kavramlardır. Örneğin sadece bir işlemciniz varsa uygulamanız eş zamanlı olarak çalışabilir fakat paralel çalışmaz. Çünkü paralel programlama işletim sistemindeki işlemci seviyesinde gerçekleşir yani donanımımızla alakalıdır. Donanımımızdaki işlemci sayısı ile ilişkilidir. Paralel programlama yapabilmemiz için birden fazla çekirdekli bir donanımımızın olması gerekiyor. Eş zamanlı çalışmak içinse tek bir çekirdek yeterlidir. Go dilinde eş zamanlı çalışabilmek için Goroutine ve Channels (Kanallar) kullanılmaktadır.

7.1.1 Goroutine

Goroutine birbirinden “bağımsız” ve “eş zamanlı” olarak çalışan ufak iş parçacıkları denir. Bunu yapmak içinse bağımsız çalışmasını istediğiniz fonksiyonların başına “go” kelimesini eklememiz gerekir.



```
main.go > {} main > alfabe
1 package main
2
3 import (
4     "time"
5 )
6
7 func alfabe() {
8     for x := byte('a'); x <= byte('z'); x++ {
9         println(string(x))
10    }
11 }
12 func main() {
13     go alfabe()
14     time.Sleep(100 * time.Millisecond)
15     //go yazmadan arasındaki fark işlemi tek tek satır satır yapar
16     // go ekiyle ise yeni bir tread oluşturcak ve ana program durmadan devam eder.
17 }
```

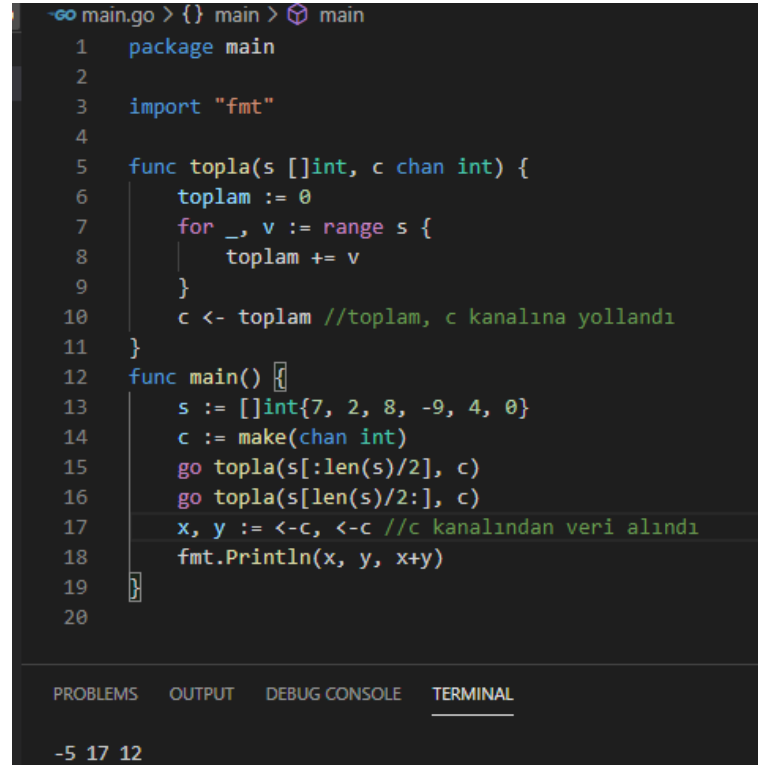
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o

Şekil 29:Goroutine

7.1.2 Channels (Kanallar)

Kanallar nesneler arasında veri alış-verişi yapabilmemizi sağlarlar. Kanalları make() ile oluşturulur. Ok ne tarafa yönlendirilirse veri o tarafa gönderilir. Maps ve Slices ise kanallar kullanılmadan önce oluşturulmalıdır.



```
-go main.go > {} main > main
1 package main
2
3 import "fmt"
4
5 func topla(s []int, c chan int) {
6     toplam := 0
7     for _, v := range s {
8         toplam += v
9     }
10    c <- toplam //toplam, c kanalına yollandı
11 }
12 func main() {
13     s := []int{7, 2, 8, -9, 4, 0}
14     c := make(chan int)
15     go topla(s[:len(s)/2], c)
16     go topla(s[len(s)/2:], c)
17     x, y := <-c, <-c //c kanalından veri alındı
18     fmt.Println(x, y, x+y)
19 }
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

-5 17 12

Şekil 30:Channels

Goroutine'ler çok kullanışlıdır fakat tek başlarına birbirleriyle iletişim kurmadan birbirlerine sinyal göndermeden çalışırlar. Kanalların yardımıyla Goroutine'ler birbirleri ile iletişim kurabilir duruma gelirler ve sinyal göndererek senkronize şekilde çalışabilirler. Goroutine herhangi bir kanala veri gönderebilir alabilir [10].

7.2 VERİTABANI İŞLEMLERİ

Golang ile veritabanlı projeler yapmak istediğinizde kodunuza eklemeniz gereken ufak değişiklikler vardır. Go Programlama Dili'nde kullanacağınız veritabanına göre ilgili driver'ı kurmalıyız. Ben sqllite3 ile çalışmaya devam edeceğim fakat diğer kurulumlar için de pek fark yoktur. Öncelikle MySql kullananlar için gerekli MySQL driver indirmeniz gerekir. Bunun için terminalde veya komut istemcide "go get -u github.com/go-sql-driver/mysql" ile indirme işlemini yapmalısınız. Yükleme işlemi bittikten sonra projemizde veritabanı kullanımı için kütüphaneyi dâhil etmelisiniz.

```

~go main.go > {} main
1  package main
2  import (
3      "database/sql"
4      _ "github.com/go-sql-driver/mysql"
5  )
6
7  func main() {
8
9  }
10

```

Şekil 31: MySql

Sqlite3 ile çalışmak için ise “go get github.com/mattn/go-sqlite3” ile indirme işlemini yapmalısınız. Daha sonra bir veritabanı oluşturup örneğin “veritabanı.db” adında içerisine id ve ad diye iki alan açalım. Daha sonra Şekil 32’deki örnek kodlar için veri eklenebilir [11].

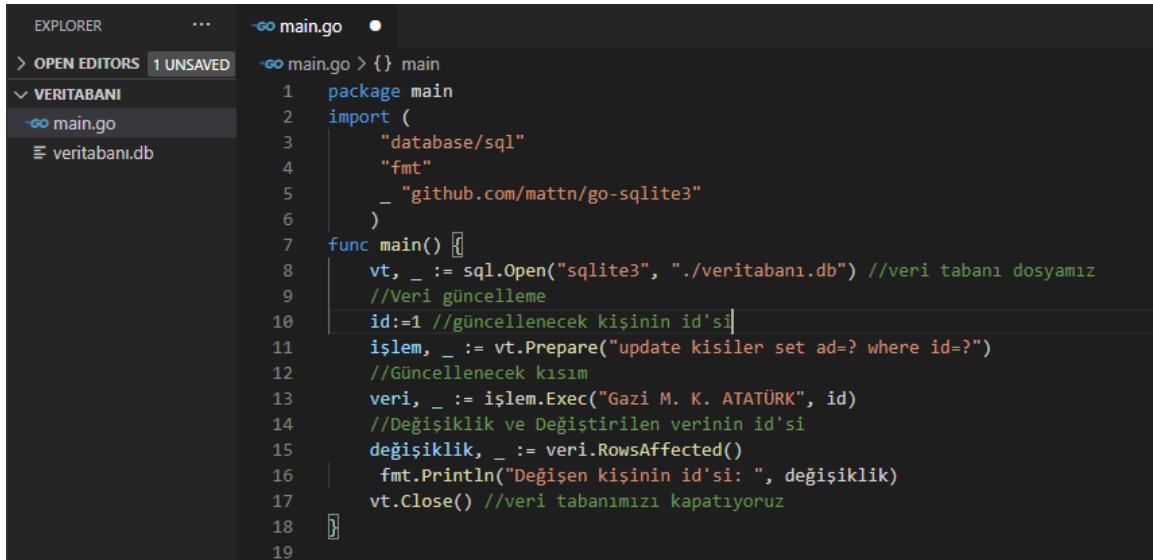
```

EXPLORER
> OPEN EDITORS 1 UNSAVED
VERI...
~go main.go
veritabanı.db

~go main.go > {} main
1  package main
2  import (
3      "database/sql"
4      "fmt"
5      _ "github.com/mattn/go-sqlite3"
6  )
7  func main() {
8      vt, _ := sql.Open("sqlite3", "./veritabanı.db") //veri tabanı dosyamız
9      //Veri ekleme
10     işlem, _ := vt.Prepare("INSERT INTO kisiler(ad) values(?)")
11     veri, _ := işlem.Exec("Kübra Kılıç") //Eklenecek değer
12     id, _ := veri.LastInsertId() //Son id numarası
13     fmt.Println("Son kişinin id'si", id)
14     vt.Close() //veri tabanımızı kapatıyoruz
15 }
16

```

Şekil 32:Veri Ekleme



```
1 package main
2 import (
3     "database/sql"
4     "fmt"
5     _ "github.com/mattn/go-sqlite3"
6 )
7 func main() {
8     vt, _ := sql.Open("sqlite3", "./veritabanı.db") //veri tabanı dosyamız
9     //Veri güncelleme
10    id:=1 //güncellenecek kişinin id'si
11    işlem, _ := vt.Prepare("update kisiler set ad=? where id=?")
12    //Güncellenecek kısım
13    veri, _ := işlem.Exec("Gazi M. K. ATATÜRK", id)
14    //Değişiklik ve Değiştirilen verinin id'si
15    değişiklik, _ := veri.RowsAffected()
16    fmt.Println("Değişen kişinin id'si: ", değişiklik)
17    vt.Close() //veri tabanımızı kapatıyoruz
18 }
19
```

Şekil 33:Veri Güncelleme



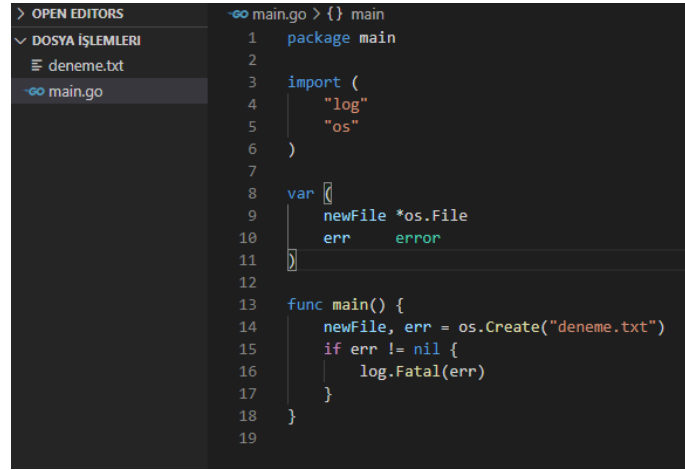
```
1 package main
2 import (
3     "database/sql"
4     "fmt"
5     _ "github.com/mattn/go-sqlite3"
6 )
7 func main() {
8     vt, _ := sql.Open("sqlite3", "./veritabanı.db") //veri tabanı dosyamız
9     //Veri Silme
10    işlem, _ := vt.Prepare("delete from kisiler where id=?")
11    veri, _ := işlem.Exec(id) //Silinecek id
12    değişiklik, _ := veri.RowsAffected() //Silinen kişinin id'sini aldık
13    fmt.Println("Silinen kişinin id'si: ", değişiklik)
14    vt.Close() //veri tabanımızı kapatıyoruz
15 }
16
```

Şekil 34: Veri Silme

8. DOSYALAMA İŞLEMLERİ

8.1 DOSYA OLUŞTURMA

Dosya okuma yazma işlemleri temel görevlerdendir. Öncelikle dosya oluşturmadan başlayalım. Şekil 35’de “deneme.txt” adında bir txt dosyası oluşturduk. İki paket eklendi biri os paketi diğeri de log paketidir. “os” paketi işletim sistemleri ile işlem yapabilmemizi sağlayan pakettir. “log” paketi ise kayıt işlemleri için kullandığımız pakettir. Projemizi kaydettiğimizde sol bölmede “deneme.txt” oluşmuştur.

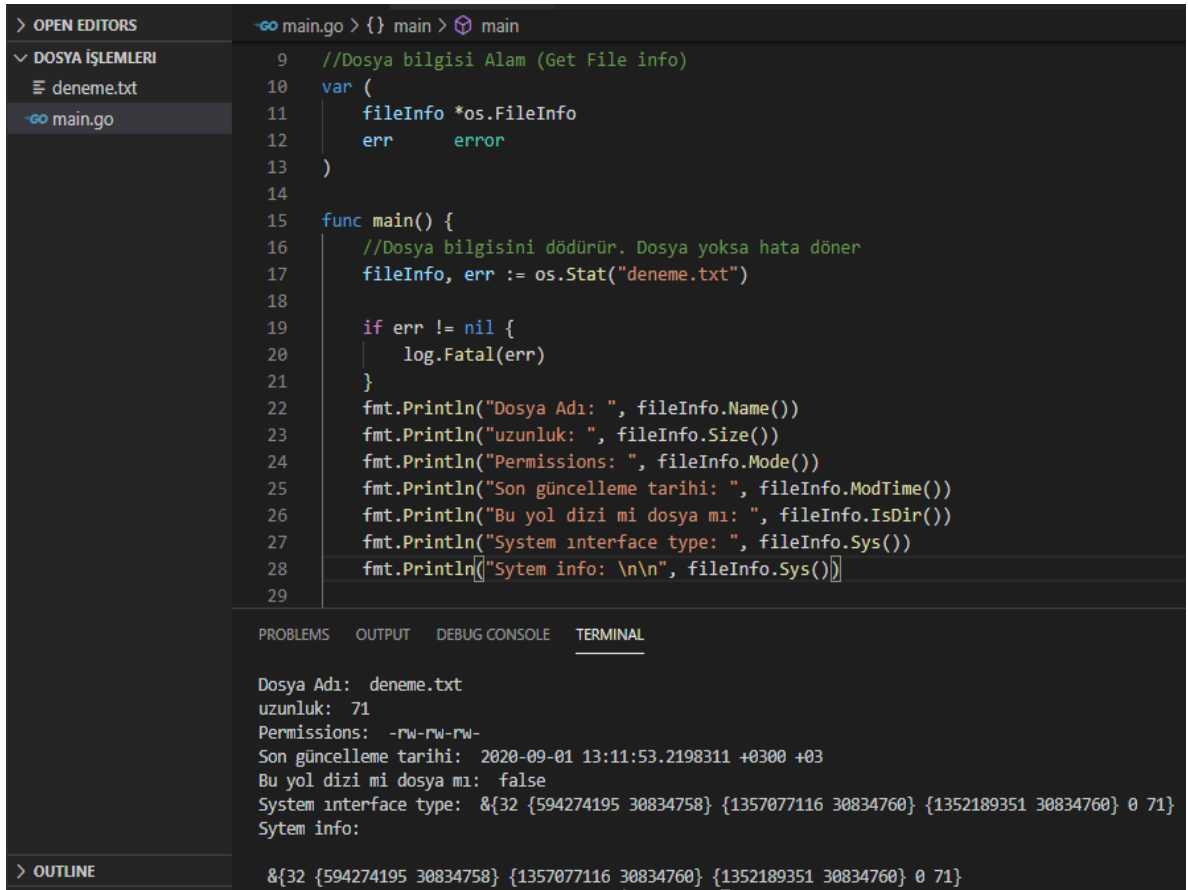


```
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 var (
9     newFile *os.File
10    err      error
11 )
12
13 func main() {
14     newFile, err = os.Create("deneme.txt")
15     if err != nil {
16         log.Fatal(err)
17     }
18 }
19
```

Şekil 35:Dosya Oluşturma

8.2 DOSYA BİLGİSİ ALMA

Bir dosya bildiği almadan önce öncelikle bir dosya oluşturmanız gerekmektedir. Dosya bilgisi almak içinde import edilen kütüphaneler "fmt", "log", "os". Dosya oluşturma kısmında açıkladığımız kütüphanelerdir. Farklı olan “fmt” paketidir o da zaten ekrana yazma işlemi yaptığımız için eklenmiştir. Bu kütüphaneleri elle eklemenize gerek yoktur kodları yazıp kaydettiğinizde Golang otomatik ekleme yapacaktır.



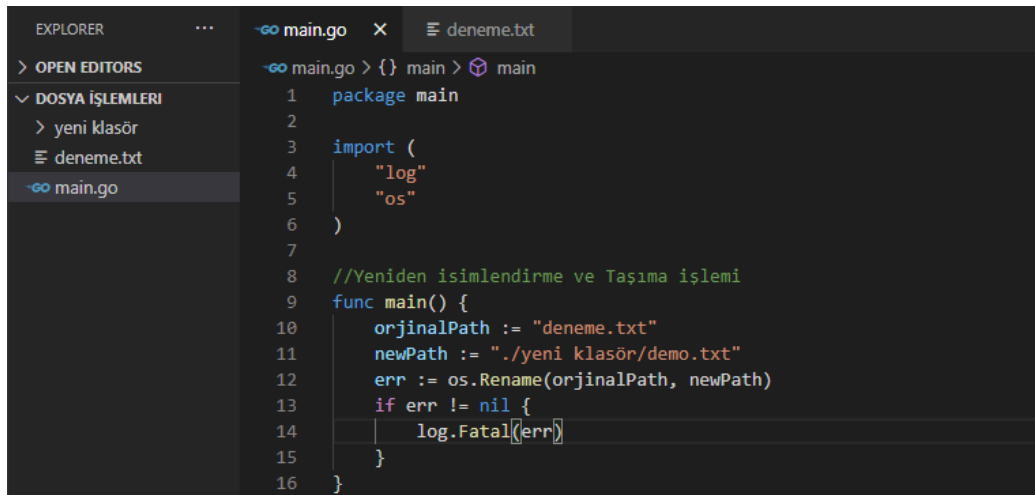
```
> OPEN EDITORS
DOSYA İŞLEMLERİ
  deneme.txt
  main.go
main.go
9 //Dosya bilgisi Alam (Get File info)
10 var (
11     fileInfo *os.FileInfo
12     err      error
13 )
14
15 func main() {
16     //Dosya bilgisini dödürür. Dosya yoksa hata döner
17     fileInfo, err := os.Stat("deneme.txt")
18
19     if err != nil {
20         log.Fatal(err)
21     }
22     fmt.Println("Dosya Adı: ", fileInfo.Name())
23     fmt.Println("uzunluk: ", fileInfo.Size())
24     fmt.Println("Permissions: ", fileInfo.Mode())
25     fmt.Println("Son güncelleme tarihi: ", fileInfo.ModTime())
26     fmt.Println("Bu yol dizi mi dosya mı: ", fileInfo.IsDir())
27     fmt.Println("System interface type: ", fileInfo.Sys())
28     fmt.Println("Sytem info: \n\n", fileInfo.Sys())
29 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Dosya Adı: deneme.txt
uzunluk: 71
Permissions: -rw-rw-rw-
Son güncelleme tarihi: 2020-09-01 13:11:53.2198311 +0300 +03
Bu yol dizi mi dosya mı: false
System interface type: &{32 {594274195 30834758} {1357077116 30834760} {1352189351 30834760} 0 71}
Sytem info:
&{32 {594274195 30834758} {1357077116 30834760} {1352189351 30834760} 0 71}
```

Şekil 36: Dosya Bilgisi Alma

8.3 DOSYAYI YENİDEN İSİMLENDİR VE TAŞIMA

Yapılan işlem “deneme. txt” dosyamızı yeni açtığımız “yeni klasör” kısmına taşımak. Dikkat edilmesi gereken sol kısımdır. “orjinalPath” orijinal dosyamızın adı iken “newPath” yeni dosyamızın yolu ve adıdır.



```
EXPLORER
  ...
  main.go
  deneme.txt
  yeni klasör
  deneme.txt
  main.go
main.go
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 //Yeniden isimlendirme ve Taşıma işlemi
9 func main() {
10     orjinalPath := "deneme.txt"
11     newPath := "../yeni klasör/demo.txt"
12     err := os.Rename(orjinalPath, newPath)
13     if err != nil {
14         log.Fatal(err)
15     }
16 }
```

Şekil 37:Taşımadan Önce

```
EXPLORER
> OPEN EDITORS
  DOS...
  yeni klasör
    demo.txt
  main.go

main.go
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 //Yeniden isimlendirme ve Taşıma işlemi
9 func main() {
10     orjinalPath := "deneme.txt"
11     newPath := "./yeni klasör/demo.txt"
12     err := os.Rename(orjinalPath, newPath)
13     if err != nil {
14         log.Fatal(err)
15     }
16 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
3: go
PS C:\Users\Kübra\Desktop\golang\Dosya İşlemleri> go run main.go
PS C:\Users\Kübra\Desktop\golang\Dosya İşlemleri>
```

Şekil 38:Taşımadan Sonra

8.4 DOSYAYI AÇMAK VE KAPATMAK

Dosyayı açıp işlem yaptıktan sonra kapatmak önerilir. Çünkü işletim sistemi sizin dosya ile işinizin bitip bitmediğini anlayamaz. Sizin başka bir zamanda dosyayı açmak istediğinizde(Mouse ile) size “bu dosya başka bir işlem tarafından kullanılıyor” gibi bir mesaj döndürecektir. Bu yüzden kapatmanız önerilir.

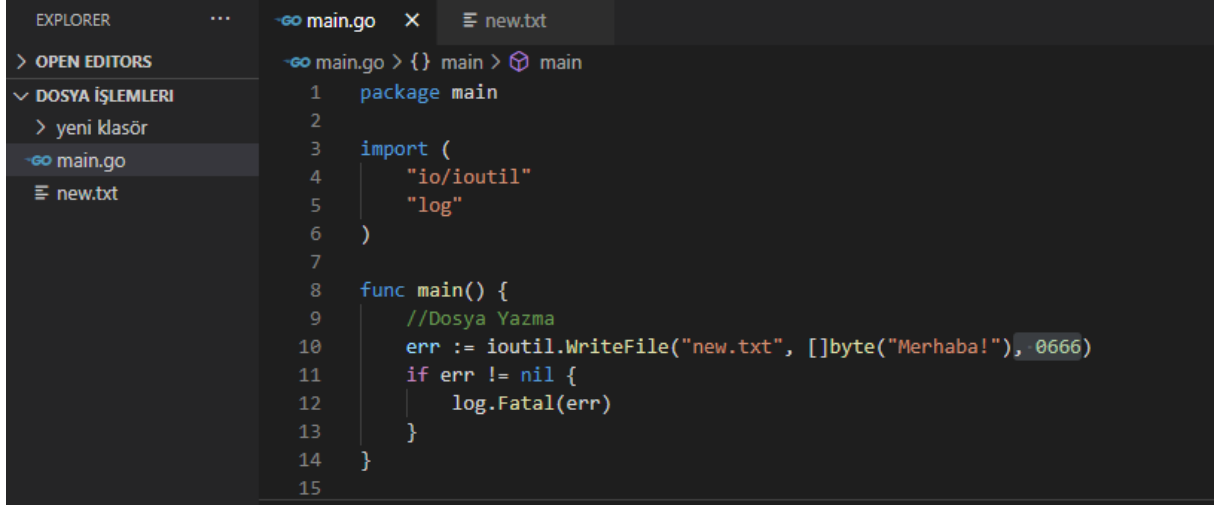
```
EXPLORER
> OPEN EDITORS
  DOSYA İŞLEMLERİ
    yeni klasör
  main.go
  new.txt

main.go
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 //Dosyayı Açma ve Kapama
9 func main() {
10     file, err := os.Open("new.txt")
11     if err != nil {
12         log.Fatal(err)
13     }
14     //Veri yazma kısmı
15     file.Close()
16 }
```

Şekil 39:Dosya açma ve kapatma

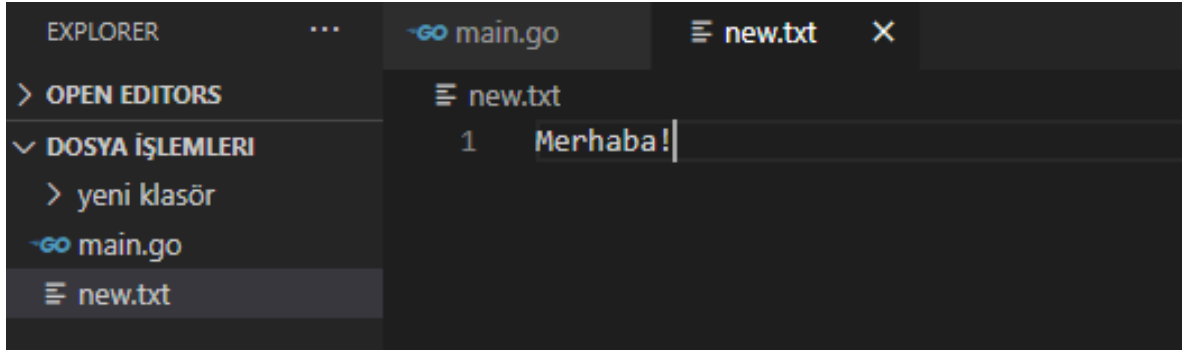
8.5 DOSYA YAZMA İŞLEMİ

Go 'da dosya yazma işlemini basit bir kısa yolu vardır. Bu ioutil kütüphanesidir. Bu sayede direk yazma işlemi gerçekleştirmemizi sağlar.



```
EXPLORER    ...    -Go main.go X    new.txt
> OPEN EDITORS
✓ DOSYA İŞLEMLERİ
  > yeni klasör
  -Go main.go
  new.txt
-Go main.go > {} main > main
1  package main
2
3  import (
4      "io/ioutil"
5      "log"
6  )
7
8  func main() {
9      //Dosya Yazma
10     err := ioutil.WriteFile("new.txt", []byte("Merhaba!"), 0666)
11     if err != nil {
12         log.Fatal(err)
13     }
14 }
15
```

Şekil 40:Dosya Yazma



```
EXPLORER    ...    -Go main.go    new.txt X
> OPEN EDITORS
✓ DOSYA İŞLEMLERİ
  > yeni klasör
  -Go main.go
  new.txt
new.txt
1  Merhaba!
```

Şekil 41:Dosya Yazma-2

8.6 DOSYA SİLME İŞLEMİ

Şekil 42'deki 10.satıra silinecek dosyamızın adı yazılır. Eğer dosyanız ana klasör içinde değilse bunu dolunu yine aynı yerde belirtmeniz gerekmektedir. Örneğin başka bir klasördeki dosyayı silecekseniz yolu ile birlikte bunu belirtmelisiniz.

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar has a tree view with 'DOSYA İŞLEMLERİ' expanded, showing 'yeni klasör', 'main.go', and 'new.txt'. The 'main.go' file is selected. The editor window shows the Go code for file deletion. The code is as follows:

```
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 //Dosya Silme İşlemi
9 func main() {
10     err := os.Remove("new.txt")
11     if err != nil {
12         log.Fatal(err)
13     }
14 }
```

Şekil 42:Dosya Silme Önce

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar has a tree view with 'DOS...' expanded, showing 'yeni klasör', 'main.go', and 'new.txt (deleted)'. The 'main.go' file is selected. The editor window shows the Go code for file deletion. The code is as follows:

```
1 package main
2
3 import (
4     "log"
5     "os"
6 )
7
8 //Dosya Silme İşlemi
9 func main() {
10     err := os.Remove("new.txt")
11     if err != nil {
12         log.Fatal(err)
13     }
14 }
```

The bottom of the screenshot shows the terminal output:

```
PS C:\Users\Kübra\Desktop\golang\Dosya İşlemleri> go run main.go
PS C:\Users\Kübra\Desktop\golang\Dosya İşlemleri>
```

Şekil 43:Dosya Silme Sonra

9. GO’NUN SEKTÖRDEKİ YERİ VE WEB PROGRAMLAMA

9.1 GOLANG VE SEKTÖR

Öncelikle Golang yeni gelişmekte olan birdir ve sizin projeniz varsa bunu geliştirme istiyorsanız diğer dillere oranla biraz daha çaba göstermeniz gerekecektir. Dil gelişmekte olduğu için bazı kısıtlamalar olsa da şuan birçok marka Golang dilini benimsemiştir. Docker, Koding, Google, Apple, Twitter, Github, Ubuntu, Amazon, Facebook, Medium, Dropbox örnek olarak verilirken ülkemizde ise yazılım şirketlerinin ve oyun programcılarının yanında Trendyol gibi yerlerin yazılımında da kullanılmaktadır. Golang sistem ve sunucu geliştirme olarak programlanmıştır, Google ilk zamanlar go dilini yazılımlarında oluşan hataların çözümünde kullanmaktaydı. Bulut depolama paylaşım hizmeti olan Dropbox, ağındaki 500 milyondan fazla kullanıcıyı kontrol altında tutmak için Golang’ı kullanmaktadır. Ayrıca ayda milyonlarca trafiği işleyen e-ticaret siteleri oluşturmak için de ideal bir programlama dilidir [12].

9.2 GOLANG İLE WEB PROJESİ GELİŞTİRME

Golang ile html üzerinde çalışmak için localhostlardan faydalanırız. Şekil 44’de localhost:5555 üzerinden çalışan bir web sunucusu örneği vardır [12]. “net/http” paketi, HTTP taleplerini dinlemek ve çıktı üretmek için gerekli pakettir. Net/Http uygulamamızda kullanılmak üzere HTTP istemcisi ve sunucu uygulamaları sağlayan pakettir.

```

- main.go > ...
1  package main
2
3  import (
4      "fmt"
5      "io/ioutil"
6      "net/http"
7  )
8
9  func loadFile(fileName string) (string, error) {
10     bytes, err := ioutil.ReadFile(fileName)
11     if err != nil {
12         return "", err
13     }
14     return string(bytes), nil
15 }
16
17 func handler(w http.ResponseWriter, r *http.Request) {
18     var body, _ = loadFile("index.html")
19     fmt.Fprintf(w, body)
20 }
21
22 func main() {
23     http.HandleFunc("/", handler)
24     http.ListenAndServe(":5555", nil)
25 }
26

```

Şekil 44:HTML Örnek

```

<> index.html > html > body
1  <!DOCTYPE html>
2  <html lang="tr">
3  <head>
4  |   <title>Sayfa Başlığı</title>
5  </head>
6  <body>
7  |   Sitemize Hoş Geldiniz! :)
8  </body>
9  </html>

```

Şekil 45:HTML Örnek devam

← → ↺ ⓘ localhost:5555

Sitemize Hoş Geldiniz! :)

Şekil 46:Localhost Çıktısı

9.2.1 Paket inceleme: net/http

Web uygulaması geliştirirken kullanılan en temel paket “net/http” paketidir. Web programlama tabanlı bütün işlemler bu pakette gerçekleşir. Get, Head, Post ve PostForm, HTTP (veya HTTPS) isteklerini oluşturabilir [18].

```
err: = http.Get (" http://example.com/ ")

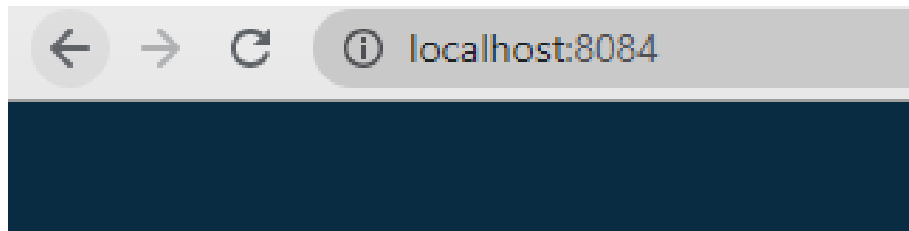
err: = http.Post (" http://example.com/upload ", "image / jpeg" ve buf)

err: = http.PostForm (" http://example.com/form ",
    url.Values {"key": {"Value"}, "id": {"123"}})
```

Şekil 47: net/http

9.3 GOLANG BİLGİ SİTESİ

Bizim geliştirmek üzerine web sitemiz için html ve css dosyalarımızdan oluşuyor. “localhost:8084” üzerinden çalışan bir “index.html” dosyamız var.



Şekil 48:Çağırdığımız adres



Şekil 49:Anasayfamız

Bizim geliştirdiğimiz web sitemiz localhost ile index dosyamıza bağlandık. Golang programlama dilinin web uygulaması için en uygun bir dil olarak görülmesinin sebebi hem main.go dosyanızın içinde bir web sitesi geliştirebilirken hem de başka bir html dosyanıza bağlanabilirsiniz. Go'nun web programlama için diğer dillerden avantajları fazladır. Örneğin Golang derleme olarak diğer dillerden çok hızlıdır ve ortaya çıkan dosyanın kapladığı alan oldukça azdır. Söz dizimi olarak öğrenilmesi kolaydır. Web projenizin temel kodu aynı kalıp sadece geliştirmek istediğiniz kısımlara yoğunlaşarak ilerleme kaydetmeniz diğer dillerden daha fazladır. Golang statik bir dil olsa da yazılımcılara dinamik bir dil avantajlarını da sağlar. Eşzamanlılık bu dili ayıran bir başka özelliklerden biridir. Hızlı gelişmesi, modern bir dil olması, statik kod analizi sağlaması gibi birçok avantajları vardır. Aşağıdaki şekillerde sitemizin tasarımı gösterilmiştir.

GOLANG

Golang diğer bir ismiyle Go dili, programlama verimliliğini arttırmak ve geliştiricilerin yaşadığı problemlerden yola çıkılarak ortaya çıkarılmıştır. Google tarafından geliştirilen bu dil statik olarak yazılmasının yanında bellek yönetimini kendisi sağlayan ve ertelenmiş fonksiyonları destekleyen bir çöp toplama sistemine sahiptir. Golang resmi web sitesinde bu dil için "Go basit, güvenilir ve verimli yazılım oluşturmaya kolaylaştıran açık kaynaklı bir programlama dilidir" ifadeleri kullanılmıştır. Golang ilk zamanlar Google'ın arka plandaki sistem performansını arttırmak amacıyla geliştirilmiş bir sistem programlama dili iken zamanla az kaynak tüketmesi, web uygulamalarının kendi kendini host ediyor olması ve yüksek performans özellikleri sayesinde Web programlayıcıları tarafından dikkat çekmeye başlamıştır. Golang 2007 yılında Google ekibinden Robert Griesemer, Rob Pike ve Ken Thompson tarafından geliştirilen programlama diliydi fakat ilk kez Kasım 2009 açık kaynak olarak piyasaya tanıtıldı. İlk sürüm 1.0, Mart 2012'de yayınlandı şuan kullanılan son sürüm ise 2020 yılında güncellenmiş olan 1.15'tir. Neden yeni bir dil çıktığı kısmına değinecek olursak bazı kaynaklarda dili geliştiren tasarımcıların köklü dillerin(C++,Java, python vb.) büyük projelerde performans, derleme gibi sorunlarından sıkılıp yeni bir geliştirmek için motive olup bunu başardıkları yazmaktadır. Bu nedenle Golang'a eklenen her özellik yılların verdiği deneyimlerle eklenmiş olup herkesin anlayabileceği tarzda geliştirilmektedir. Golang diline özel olarak geliştirilmiş maskot Gopher ayırt edici özelliklerinden biridir. Go Gopher daha önceden bağış toplama etkinliği için bir projede tasarımcısı Renee French tarafından yayınlanmıştır. Go projesine başlanıldığı zamanda Renee French tasarımcısı olmayı istemiştir ve Gopher'ı tasarlamıştır. Sincaba benzeyen bu maskotun adı "Go Gopher" olarak kalmıştır. Daha sonradan mavi renge boyanmıştır ve kullanılmaya başlanmıştır. Bu yüzden Golang yazılımcılarına Gopher denmektedir.

GOLANG VS DİĞER DİLLER

Şekil 50: Sayfa örnek

GO PROGRAMLAMA

Golang Programlama Dili

Dille ilgili çalışmalar yapmadan önce dil hakkında bilgiler edinilmelidir. Bu araştırma öncelikle Go dilinin temel özellikleri belirtilmiş, veri yapıları üzerinde durulmuş, uygulamalar geliştirilmiş (Kodlanmış) ve derlenerek nasıl bir sonuç ürettiği ortaya çıkartılmıştır. Son olarak görsel bir web ara yüzü ile dilin anlaşılması kolay gele getirilmiştir. Golang dilinin bazı özellikleri aşağıda sıralanmıştır.

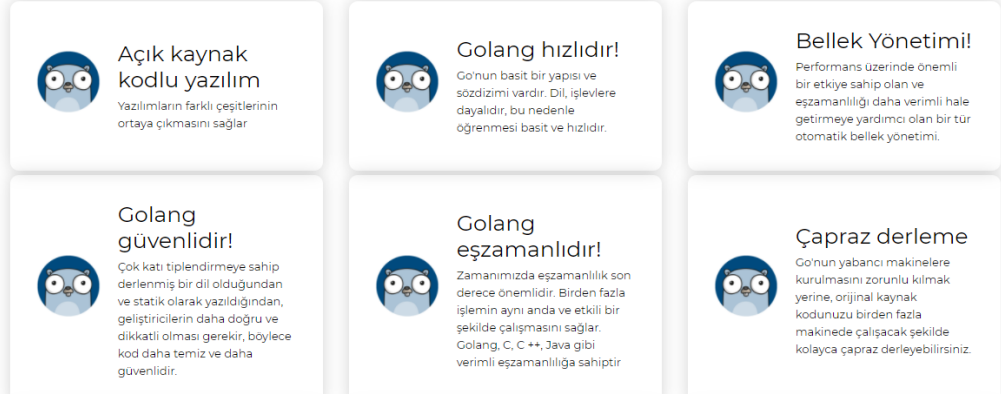
- Açık kaynak kodlu dildir.
- Derleme süresi hızlıdır.
- Modern bir dildir.
- Modern teknolojinin ihtiyaçları göz önünde bulundurularak tasarlandı.
- Yüksek performans ve hızlı gelişebilme özelliği ile Java ile kıyaslanabilecek seviyededir.
- Çoğu iş sadece standart kütüphaneler ile yapılabilirdi için dil yapısı basittir.
- Eş zamanlı programlamada Go basit fakat performansı yüksek imkânlar sunmaktadır.
- Eşzamanlılık(Birden fazla işlemin eşzamanlı ve etkili bir şekilde çalışmasını sağlar) özelliği vardır.

Dil, neredeyse 2009'da piyasaya çıkar çıkmaz yükselmeye başladı. Hızla 65. dilden dünyanın en üst sıralarına doğru kavmaya başladı. Ardından Business Insider "Go'yu 2016'nın en sıcak



Şekil 51: Sayfa örnek-2

GOLANG AVANTAJLARI



Şekil 52: Sayfa örnek-3

10.GOLANG VS DİĞER DİLLER

10.1 GOLANG VS C++

Golang 2007 yılında Google ekibinden Robert Griesemer, Rob Pike ve Ken Thompson tarafından geliştirilen programlama diliydi fakat ilk kez Kasım 2009 açık kaynak olarak piyasaya tanıtıldı. C++ ise 1983 yılında Bjarne Stroustrup tarafından çıkarılmıştır. İki dilde statik olarak yazılmıştır fakat Golang dili daha basit ve kolay olduğundan C++’dan daha çabuk öğrenilebilir [13]. Golang, C++’dan farklı olarak parantez ayrışımı yapar. İki dili birbirinden ayıran diğer bir özellikte derleme zamanlarıdır. C++ yavaş derleme süresine sahip iken Golang çok daha hızlı derleme gerçekleştirir. C++ nesneye yönelimli bir dil iken Golang açık olarak nesneye yönelimli değildir. C++ sınıflar varken Golang dilinde sınıf kavramı yoktur. C++’da güvenlik açıkları Golang’a oranla daha fazladır. C++ açık bir dildir yani programcı veya program, kaynak kodun her bölümüne ve onu çalıştıran makineye erişebilir. Fakat Golang daha kapalı birdir. İki dili kütüphaneleri bazında kıyasladığımızda Golang’ın kütüphane sayısı daha azdır onu daha basit kılan özelliklerden biride budur. Her iki dilin birbirinden üstün özellikleri vardır fakat unutulmamalıdır ki Golang modern standartlara uyacak şekilde yazılmıştır [14].

10.2 GOLANG VS PYTHON

Kıyaslanan bu iki dilde aslında web programa için hız ve sadelik konusunda her zaman tartışma konusu olmuştur. Golang ve Python arasındaki dilin sadeliği kıyaslandığında Golang daha baskın gelmektedir. Golang python’a oranla eşzamanlılık modeli daha hızlı performans sağlar. Go statik, python ise dinamik bir dildir [15]. Golang arayüzleri desteklerken python desteklemez bu yüzden Go daha fazla ayrıntılıdır. Python kodlardaki istisnaları kısmen görmezden gelirken Golang hata ile direk bunu hemen belirtir. Son olarak bu iki dilinde kullanım amacı farklıdır. Python web geliştirme, oyun geliştirmede, Linux temelli uygulama yönetimi için kullanılırken Golang daha çok sistem dili olarak kullanılmaktadır. Fakat go ile web geliştirmekte mümkündür.

10.3 GOLANG VS JAVA

Golang ile Java'yı kıyaslamak aslında köklü ve günümüzde en çok kullanılan bir dil ile sektörde yeni olan, yeni tanınan bir arasındaki kıyaslamadır. Kod yazmak konusunda ikisi de C dilinden temel olsa da farklılıklar oldukça fazladır. Go, Java gibi nesne yönelimli değildir. Performans bakımından Go ile java arasında fark vardır. Go daha hızlıdır. Java'nın aynı zamanda çoklu okuma yetenekleri de vardır, ancak bunlar Go'nun modern, en son teknolojisiyle aynı seviyede değildir. Go'da Java'ya oranla herhangi bir çözüme sahip olmayan bir problemle karşılaşılma olasılığınız oldukça düşüktür. Çünkü sizin yaptığınız hatayı mutlaka biri yapmıştır ve çözümü vardır [16].

10.4 GOLANG VS PHP

Golang programlama dili statik olarak yazılmıştır fakat PHP ise dinamik olarak yazılmıştır. Bu iki dilin derleme hızı konusunda yazılar okurken bir grup API geliştiricisi API'yi PHP dilinde yazmışlar daha sonra ilerde oluşabilecek sorunlardan endişe duydıkları için başka bir dil daha araştırmaya başlamışlar. Golang ile tanışan ekip API'yi go dilinde yazıp iki dil arasında hız testine tâbi tutmuşlar. Sonuç olarak Golang php'den daha hızlı çalışma ve performans göstermiştir [17]. Go dilinde PHP'de olmayan kanallar ve gorutinler gibi özellikler vardır. Go temelde makine dili programlaması için geliştirilirken, php ise daha çok web geliştirmek amaçlı kullanılmaktadır. Go kaynak dosyalarını otomatik olarak biçimlendirirken PHP'de böyle bir özellik yoktur. Bu iki dil arasındaki fark oldukça fazladır.

10.5 GOLANG VS SWIFT

İkili arasında en bilinen farklılık Golang basit bir dil iken Swift karmaşık bir dildir. Swift'in temelde kitaplık desteği Apple merkezli iken Golang açık kaynaktır ve bir işletim sistemine özgü olmak gibi kısıtlaması yoktur. Swift, Go ile kıyaslandığında derleme süresi yavaş kalmaktadır. TIOBE endeksine göre Golang 11.sırada ve yükselirken Swift 12.sıradadır.

11.SONUÇLAR VE ÖNERİLER

Go programlama dilinin diğer dillerden hızlı olması, dil yapısının basit ve kolay öğrenilebilir olması, hata ayıklama sisteminin olması ve işlemci hızını zorlamaması, kütüphanesinin karışık olmaması ve kısacası hızla gelişen bir dil olması sebebiyle aslında yeni bir dil öğrenmek veya bu alanda gelişmek isten yazılımcılar için go programlama doğru bir tercih olabilir. Ülkemizde neredeyse hiç tercih edilmemektedir fakat 2018 yılında TIOBE endeksinde 19. sırada yer alırken 2020’ de 11. sıraya yükselmiştir. Bu tez çalışmasında, Go programlama dilinin temel özellikleri açıklanmış, diller ile karşılaştırılması yapılmış ve bir web projesi geliştirilmiştir. Golang programcılar için farklı bir başlangıç oluşturabilir günümüzde programlama dillerinde zaman alıcı birçok yazım tekniği mevcuttur. Kod yazan geliştirici için anlaşılması kolay kodlar ve kolay yönetilebilirlik oldukça önemlidir. Google, Facebook, Youtube gibi büyük bir teknoloji şirketinin arkasında bulunması, bu dilin sektörden kolay silinmeyeceğinin göstergesidir.

12. KAYNAKLAR

- [1] A. Yusuf, A. Mustafa, A. Ahmet Ve A. Mehmet, Hgs Sjglksg Gasglşk, Cilt 5, İstanbul: Ieee Pub, 2012, Pp. 450-489.
- [2] M. E. Kaymaz Ve Ö. Öztürk, «Orca: Go Programlama Dili İçinOrm/Odm Kütüphanesi,» *Avrupa Bilim Ve Teknoloji Dergisi Özel Sayı*, Pp. 310-317, 2020.
- [3] M. Özalp, Go Programlama.
- [4] B. Çobanoğlu, Herkes İçin Python, İstanbul: Pusula , 2019.
- [5] Y. D. H. H. Balık, «“C” İle Programlamaya Giriş,» Elazığ, 2003.
- [6] D. N. H. 6, *Diziler (Arrays)*.
- [7] «A Tour Of Go,» [Çevrimiçi]. Available: <https://tour.golang.org/moretypes/7>.
- [8] Ş. E. Şeker, «<http://bilgisayarkavramlari.sadievrenseker.com/>,» 8 Kasım 2007. [Çevrimiçi]. Available: <http://bilgisayarkavramlari.sadievrenseker.com/2007/11/08/olusum-composition-ve-struct-yapilar/>.
- [9] Y. D. D. İ. Aydın, «Bmü-112 Algoritma Ve Programlama-I Inheritance (Kalıtım),» Ders Notu, [Çevrimiçi]. Available: http://web.firat.edu.tr/laydin/bmu112/week_5_soyutsiniflar_kalitim_arayuz.pdf.
- [10] R. Kara, «Ders Notu,» [Çevrimiçi]. Available: <http://akademik.duzce.edu.tr/content/dokumanlar/resulkara/dersnotlari/df81494c-3876-48e5-99ef-0ccd69105d70.pdf>.
- [11] E. Yakut, «<https://www.yakuter.com/>,» [Çevrimiçi]. Available: <https://www.yakuter.com/go-dilinde-concurrency/>.
- [12] K. Kuşçu, «[Go.Kaanksc.Com](http://go.kaanksc.com/),» [Çevrimiçi]. Available: <https://go.kaanksc.com/veritabani/sqlite3>.
- [13] «Uptech,» [Çevrimiçi]. Available: <https://uptech.team/blog/why-use-golang-for-your-project>.
- [14] J. Stjernberg Ve J. Annebäck, «A Comparison Between Go And C++,» *Bachelor's Thesis In Computer Science(Yüksek Lisans Tezi)*.
- [15] «[Careerkarma.Com](https://careerkarma.com/),» [Çevrimiçi]. Available: <https://careerkarma.com/blog/go-vs-c-plus-plus/#go-vs-c++-speed-and-readability>.
- [16] «[Qarea.Com](https://qarea.com/),» [Çevrimiçi]. Available: <https://qarea.com>.
- [17] «[Careerkarma.Com](https://careerkarma.com/),» [Çevrimiçi]. Available: <https://careerkarma.com/blog/go-vs-java/#:~:Text=Java%20is%20older%2C%20object%2Doriented,Used%20for%20server%2Dside%20programs..>
- [18] M. Design, «[Medium.Com](https://medium.com/@mtrdesign/php-or-go-7d9ee1a86e17),» [Çevrimiçi]. Available: <https://medium.com/@mtrdesign/php-or-go-7d9ee1a86e17>.
- [19] Golang.Org, «Golang,» [Çevrimiçi]. Available: <https://golang.org/pkg/net/http/>.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Kübra Kılıç
Doğum Tarihi ve Yeri :28/01/1998 Beykoz/İstanbul
Yabancı Dili :Orta Seviye
E-posta :kubraklc149@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh.	Düzce Üniversitesi	okuyor
Lise	Web programlama	Üsküdar Cumhuriyet KTML	2016