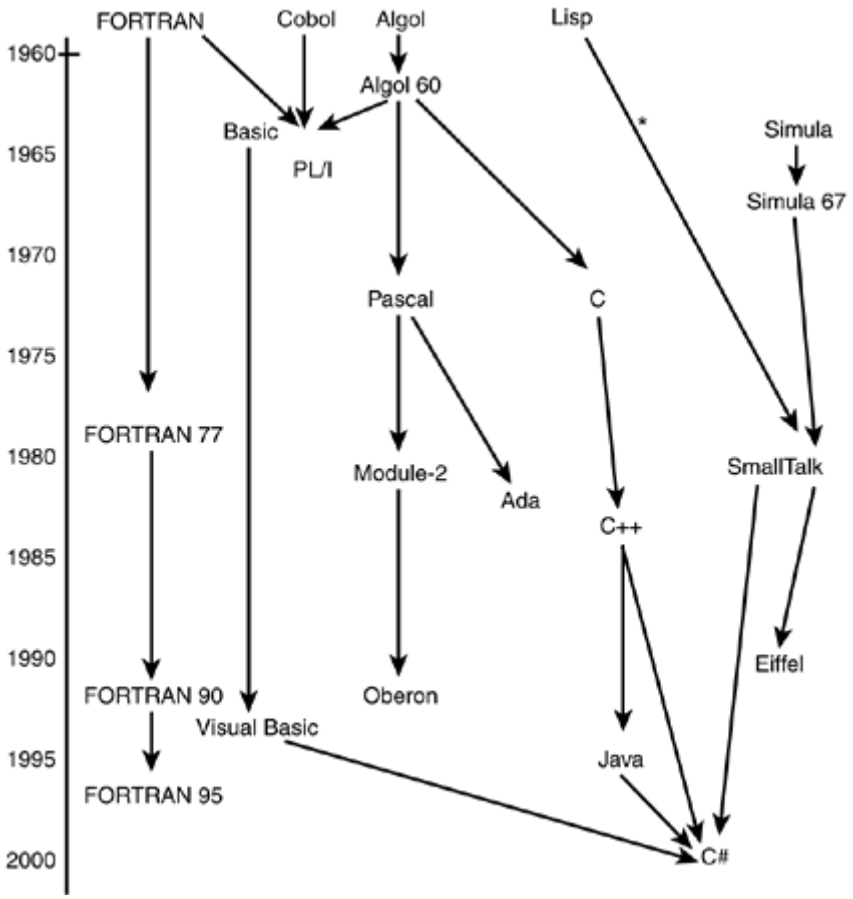
Programlama Dili Prensipleri

Lab Notları – 1

Programlama Dillerinin Tarihçesi:



C Programlama Dili

C dilinin ANSI standardı 1989'da onaylanmış fakat 1990 tarihinde yayınlanmıştır. C dili, hızlı ve düşük seviye özelliklere erişmek isteyen uygulamaların yazılması için popüler bir dildir. C programlama dili yapısal bir programlama dili olup nesne desteği yoktur. Aktif bellek yönetimini destekleyen C dilini kullanacak bir programcı bellek yönetimini çok iyi bilmelidir. Kaynak dosyaları genelde ".c" uzantısına sahiptir. Bu derste kodlar derlenirken bir GNU derleyicisi olan MinGW kullanılacaktır. MinGW kurulumunu yapmak için SABİS üzerinde paylaşılan dosya incelenebilir.

İlk Program

Girilen 4 basamaklı bir sayıyı basamaklarına ayırmak.

```
#include "stdio.h"
#include "locale.h"

int main()
{
  int sayi;
```

```
 setlocale(LC_ALL,"Turkish"); // Türkçe karakter desteği için
 do{
        printf("4 Basamaklı bir sayı girin:");
        scanf("%d",&sayi);
 }while(sayi<1000 || sayi>10000); //4 basamaklı sayı kontorlü
 short birler,onlar,yuzler,binler;
 binler=sayi/1000;
 yuzler=(sayi%1000)/100;
 onlar=(sayi%100)/10;
 birler=sayi%10;
 printf("\nBinler:%d\n",binler);
 printf("\nYüzler:%d\n",yuzler);
 printf("\nOnlar:%d\n",onlar);
 printf("\nBirler:%d\n",birler);
 getchar();
 return 0;
}
```

Nasıl Derlenir: Komut satırından kodun bulunduğu klasöre gelerek "gcc -o basamak basamak.cpp" girilip enter tuşuna basılırsa derleme gerçekleşip basamak.exe dosyası oluşacaktır.

Kod incelendiğinde yorum satırlarının // ifadesiyle başladığı görülecektir. Birden çok satırda yorum yapmak için /* yorumlar… */ şeklinde bir kullanım yapılmalıdır. Yorum satırları derleyiciler tarafından görülmez ve o satırlar atlanır. Bu satırlar programcılar için açıklama mahiyetindedirler.

karakteri ile başlayan ifadeler, ön işlemci komutlarıdır. Derleme işleminin hemen öncesinde çalıştırılırlar. Bir başka dosya veya kütüphane ekleme işlemleri bu şekilde yapılmaktadır.

{ } ifadeleri C dilinde kod bloklarının açılma ve kapanma ifadeleridir. Fonksiyonlar, yapı tanımlamaları döngüler, kontrol ifadelerinin hepsi birer kod bloğu olduğu için bu ifadeler ile başlayıp biterler. Fakat bu ifadeleri kullanmak için illaki bu bahsi geçen yapıların olmasına gerek yoktur. Aşağıdaki örnek kod C dilinde derlenip çalışacaktır.

```
#include "stdio.h"
int main(){
     int a=10;
     {
          printf("%d\n",a);
          {
               printf("Merhaba\n");
          }
     }
    return 0;
}
```

stdio.h kütüphanesinin eklenmesinin nedeni printf ve scanf gibi girdi ve çıktı fonksiyonlarının tanınması içindir. scanf girdi alma ifadesidir. printf ise çıktı verme için kullanılır.

Çalıştırılabilir bir C programı olabilmesi için bir main yani giriş fonksiyonuna ihtiyaç vardır. Bu derleyicinin kodun hangi satırından başlayacağını bilmesi içindir. main fonksiyonu tamamlandığında program kapanmış demektir. main fonksiyonun sonudaki return 0; ifadesi programın, işletim sistemine "her şey yolunda gitti ve işlem başarıyla tamamlandı" demesidir.

C dilinde çalıştırılacak tüm ifadeler ; ile biter. '\n' ifadesi ise bir alt satıra getirme karakteridir. C programlama dilinde kaynak kodda bırakılan boşluk miktarının bir önemi yoktur örneğin aşağıdaki program başarıyla çalışır. Sadece okunabilirliği düşüktür.

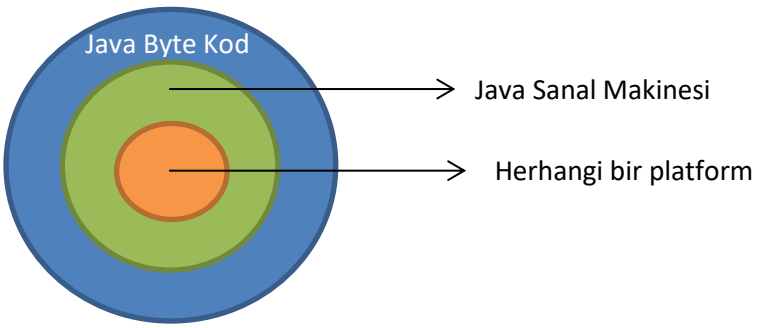
```
#include "stdio.h"
int main(){int i; for(i=0;
i<10;
i++)printf("%d\n",i); }
```

Java Programlama Dili

İlk başlarda **Oak** ismiyle tasarlanan Java programlama dili gömülü uygulamalar için kullanılıyordu. 1995 yılında Java ismini aldı. Internet uygulamaları geliştirmek için yeniden tasarlandı. Bundan sonra Java gittikçe popülerleşti. Java'nın hızlı bir şekilde yükselişinin ve bu kadar popüler olmasının ardındaki en büyük etken verdiği sözde yatmaktaydı. "Kodu bir kere yaz ve istediğin yerde çalıştır." Java dilinin desteklediği özellikleri madde halinde yazacak olursak

- Tasarımı kolay
- Nesne Yönelimli
- Taşınabilen
- Yorumlanabilen
- Güvenli
- Yüksek Performanslı
- Çoklu-Thread destekleyen
- Dinamik

Günümüzde Java, sadece web uygulaması için değil bilgisayar uygulaması için de kullanılmaktadır. Yüksek seviyeli bir dilde yazılan programa kaynak program denir. Bilgisayar kaynak programı anlayamaz bunun için bir dönüştürücüye ihtiyaç vardır. Kaynak koddan makinenin anlayacağı asambly koda çevirenlere **derleyici** denir. Derleyiciler ekstra kütüphane ve nesne dosyalarını da **linker** yardımıyla derleme sürecine dahil ederler. Uygun derleyici yardımıyla bir program kodu herhangi bir makine koduna çevrilebilir. Fakat bir makine kodu sadece ilgili özel makinede çalışacaktır. İşte Java bu engeli ortadan kaldırmak için herhangi bir platformda çalışabilecek şekilde tasarlamıştır. Bunun yaparken Java, kaynak kodu özel bir makine koduna çevirir. Bu makine koduna **Byte kod** (bytecode) denir. Bu Byte kod istediğiniz herhangi bir makinede Java sanal makinesi (Java Virtual machine) yardımıyla çalıştırılabilir.



Bir Java programı birçok yolla yazılabilir. Bir Java uygulaması, bir Applet veya bir Servlet olabilir. Uygulamalar herhangi bir bilgisayarda Java Sanal Makinesi yardımıyla çalıştırılabilen programlardır. Appletler web üzerinden çalışan özel uygulamalardır. Servlet'ler ise server üzerinden çalışan ve

dinamik web içeriği üreten programlardır. Bu ders kapsamında sadece bilgisayar üzerinde çalışan Java uygulamaları üzerinde durulacaktır.

Her Java programı en az bir sınıfa sahiptir. Sınıf metot ve verileri içeren bir yapıdır. Sınıflar ".java" uzantılı dosyalardır. Nesne dosyası oluşturulduğunda ".class" şeklinde bir dosya oluşur. Bu ders kapsamında Java derleyicisi olarak **NetBeans** kullanılacaktır. NetBeans kurulumu için aşağıdaki dosyadan yardım alabilirsiniz.

https://dosya.sakarya.edu.tr/Dokumanlar/2015/BSM208/352180945_netbeans_kurulumu.pdf

Örneğin aşağıdaki programı NetBeans'ta yazarsak. Program girilen ifadeyi ekrana yazmaktadır. Programdan da görüleceği üzere Java, C++ ile aynı altyapıdan geldiği için birçok yönden benzerlikler bulunmaktadır.

```
package ilkproje;

import java.util.Scanner;

/**
 *
 * @author M.Fatih
 */
public class IlkProje {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner scan = new Scanner(System.in);
        String girilen = scan.nextLine();
        System.out.println(girilen);
    }
}
```

Yorum açısından C ve C++ ile aynıdır. /** */ bu şekilde bir yorum tarzı dokümantasyon için kullanılır. İlk satırda bulunan package, paket tanımlaması için kullanılır. Amacı isim benzerliğinin ve bundan doğacak problemlerin önüne geçmektir. Paket ismi klasör ismi ile aynı olmalıdır. Örneğin yukarıdaki IlkProje sınıfı ilkProje klsörünün içerisindedir. Farklı bir paket ismi de olabilirdi. Buradaki önemli nokta paket ismi ile klasör isminin aynı olmasıdır. Java'da kütüphanelerin, sınıfların veya farklı dosyaların yazılan projeye eklenmesi import kelimesi ile olmaktadır. Yukarıdaki kodda girdi almak için kullanılan Scanner sınıfı import kelimesi ile projeye eklenmiştir.

import java.util.*; * karakteri de kullanılabilir. * karakteri paketin hepsini dahil eder. util paketindeki herşey dahil edilecektir. Eğer import yazılmadan kullanılmak isteniyorsa bu durumda her kullanışta tam paket yolu yazılmalıdır.

java.util.Scanner scan = new java.util.Scanner(System.in); gibi

C ve Java'da ayrılmış kelimeler (her ikisi içinde geçerlidir)

break case catch char struct const continue default do double else float for goto if int long malloc return short switch throw try void volatile while

Bu kelimelerden const ve goto artık Java için tanımsız olup kullanılmamaktadır.

Aşağıdaki kelimeler ise sadece Java için özeldir.

abstract boolean byvalue cast extends final finally future generic implements import inner instanceof interface native null operator outer package rest super synchronized this throws transient var
Yukarıda kırmızı ile yazılanlar şuanda Java'da tanımsız olup kullanılmamaktadır.

Java'da en basit program yazmak için bile sınıf tanımlamak bir zorunluluktur. Fakat Java'da main metodunun başında static ifadesi bulunmaktadır. Bu ifadelerin daha detaylı bir şekilde anlatımı ileriki bölümlerde açıklanacaktır. static konmasının sebebi main metodu nesne oluşumu başlamadan önce çalışmaya başlıyor. Eğer static olmasaydı nesnesi oluşturulup metodun çağrılması gerekecekti.

```
/* Dene.java  */
public class Dene {
   public void F(){
      System.out.println("Cagrildi...");
   }

}
```

```
Dene d = new Dene();
d.F();
```

Eğer F metodu static tanımlansa idi aşağıda görüldüğü gibi Dene sınıfının nesnesi oluşturulmadan çağrılabildi.

```
/* Dene.java  */
public class Dene {
   public static void F(){
      System.out.println("Cagrildi...");
   }

}
```

```
Dene.F();
```

Komut Satırı Parametreleri

Komut satırı parametreleri, daha program çalışmaya başlamadan önce programa belli parametreleri girmeyi sağlar. Bu C ve Java'da main metodunun parametreleri ile sağlanır. Örneğin aşağıdaki iki kod sırasıyla C ve Java'da komut satırından aldığı parametreyi ekrana yazar. C'de 1. İndekstekini almamızın sebebi 0. İndekste programın kendisini tutmaktadır. Bunun dışında C'de argc isminde bir başka parametrede bulunmaktadır. Bu argc, komut satırından kaç adet parametre girildiğini gösterir. Java'da Netbeans ortamında komut satırı parametresi girebilmek için proje adına ters tıklayıp "**Properties**" kısmına gelinir. Buradan da Run alt alanına gelindikten sonra "**Arguments**" yazan kısım komut satırı parametrelerini ifade eden kısımdır. Birden çok parametre girmek için parametreler arası boşluk bırakılmalıdır.

```
// Java
public static void main(String[] args) {
```

```
        // TODO code application logic here
        System.out.println(args[0]);
    }
```

```
// C Dili
#include "stdio.h"
int main(int argc,char *argv[]){
        printf("%s\n",argv[1]);
        return 0;
}
```

Hazırlayan

Arş. Gör. Dr. M. Fatih ADAK