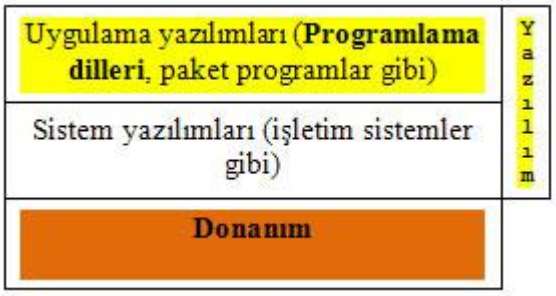
1.BÖLÜM
PROGRAMLAMA DİLLERİ ve JAVA DİLİ

GİRİŞ

Günümüzde yaygın olarak kullanılan programlama dillerinin neler olduğu, özellikleri, özelde ise Java programlama dili ve bu dillerin birbirlerinden farklarını anlatmadan önce genel olarak yazılım ve program nedir? Programcı ile bilgisayar arasında nasıl bir iletişim gerçekleşir? Gibi aklımıza ilk başta gelen soruların cevabını bulmaya çalışalım.

Bilgisayar donanım ve yazılım olmak üzere iki ana bileşenden oluşur. **Donanım (Hardware),** bilgisayarın fiziksel yapısını ifade eder. Fiziksel yapı, sadece bir metal yığınından ibarettir. Kısaca gözle görebildiğimiz elle dokunabildiğimiz parçalardır. Bilgisayarda kullandığımız her türlü programlara **yazılım (software)** denir. Kısaca donanımı kullanmak için gerekli programlara yazılım diyoruz. Bir nevi bilgisayara hayat veren, canlılık veren programlar yazılımlardır. Yazılımsız bilgisayarın ölü bir bedenden farkı yoktur.

Bu iki temel kavramdan sonra programlama dilleri yazılımın neresindedir? Sorusuna aşağıdaki şema ile cevap verebiliriz.



Programlama dillerini kurup, çalıştırabilmek için donanım üzerinde yüklü bir işletim sisteminin (sistem yazılımının) olması gerektiği açıktır.

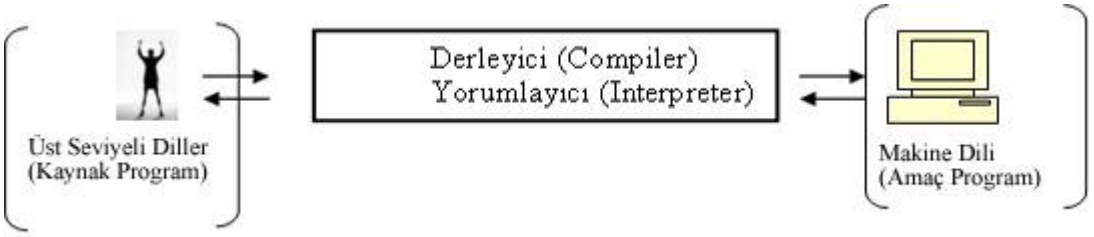
Program, bilgisayara ne yapması gerektiğini söyleyen bir grup komuta (kod topluluğuna) verilen isimdir. Bu komutları veren kişiye **programcı**, komutların bütününe ise **programlama dili** denir.

İnsan ya da programcı (yapay dil - konuşma dili bilen) ile bilgisayar (makine dili- "1,0" bilen) arasındaki iletişim nasıl gerçekleşir?

Bu sorunun cevabı için şu örneği verebiliriz. Birbirinin konuşma dilini bilmeyen iki yabancı devlet adamı bir araya gelip nasıl görüşür veya anlaşır? Doğal olarak bu görüşmenin ancak tercüman veya çevirmenler aracılığı ile gerçekleşebileceğini söyleyebiliriz. Benzer şekilde bilgisayar üreticileri, aynı sorunu insanın konuşma dilini bilgisayarın makine diline çevirecek, bir nevi tercümanlık yapacak "derleyici veya yorumlayıcı programlar" geliştirerek çözmüşlerdir.

DERLEYİCİ ve YORUMLAYICI KAVRAMLARI

Yazdığımız programları makine diline çeviren, makine dilinde elde edilen sonuçları bizim anladığımız biçime dönüştüren programlara **çevirici (derleyici veya yorumlayıcı)** programlar denir.



Kaynak Program (Source Program)

Bir programlama dili ile yazılmış programlardır. Kaynak program dosyasının uzantısı örneğin Visual Basic dilinde ise **.bas** veya **.frm** , C dilinde yazıldı ise **.c** , Java dilinde yazıldı ise **.java**, Assembly dilinde yazıldı ise **.asm** dir.

Amaç Program (Object Program)

Bir programlama dilinde yazılmış kaynak programın derlenmesi ile elde edilen program. Amaç program makine diline dönüştürülmüş ve çalıştırılmaya hazır programdır.
Programlama dilleri, kaynak kodun makine koduna (amaç programa) çevrilmesi ve çalıştırılması işleminde derleyici, yorumlayıcı veya her ikisini birlikte kullanabilir.

Derleyici (Compiler)

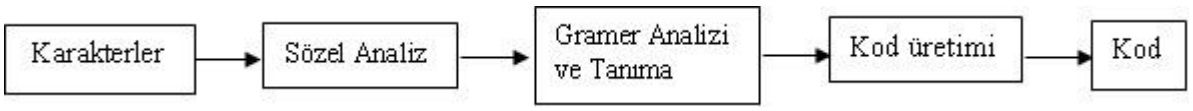
Programlama dili ile yazılmış bir programı, makine dili ile yazılmış amaç veya hedef programa çevirirler. Yüksek seviyeli bir dil ile yazılanlarda kurallarına ters düşen hata kontrollerini oluşturduktan sonra komut ve bilgilerin makine tarafından tanınarak işlemlerin yapılmasını sonuçların tekrar yüksek seviyeli program diline anlaşılır duruma çevrilmesini sağlarlar. Fortran, C, C ++, C#, Pascal, Delphi gibi diller derleyici kullanırlar.

Derleyicinin çevirme işlemi genellikle üç adımda gerçekleşir:

1. *Sözel Analiz (Lexical Analysis)*
2. *Gramer Analizi ve Tanıma (Syntax Analysis)*
3. *Kod Üreteci (Code Generator)*

Sözel analiz sırasında kaynak kodunu oluşturan kelimeler incelenir ve bu kelimelerin makine dilinde karşılıkları bulunup bulunmadığı kontrol edilir. **Gramer analizi**, kaynak kodun yazıldığı programlama dilinin kendi gramer kuralları içinde doğru yazılıp yazılmadığını kontrol eder. **Kod Üretimi** aşamasında tanınan komutlara karşılık gelen makine dili komutları üretilir. Bu adımlar

paralel olarak gerçekleştirilir ve kaynak kod, amaç koda çevrilmiş olur. Çevirme işlemi adımları aşağıdaki blok şemada görülmektedir.



Yorumlayıcı (Interpreter)

Program kodlarının ilk satırından son satırına kadar satır satır belirtilmiş komut ve işlemleri inceleyerek kaynak programın hatalarının düzeltilmesine imkân veren ve çalıştıran programdır. Herhangi bir komut satırının çevrilmesinde ya da çalıştırılmasında bir hatayla karşılaştığında çalışmayı durdurur ve hatalı satırı programcıya bildirir. Basic, PHP, Perl, Lisp gibi diller yorumlayıcı kullanırlar.

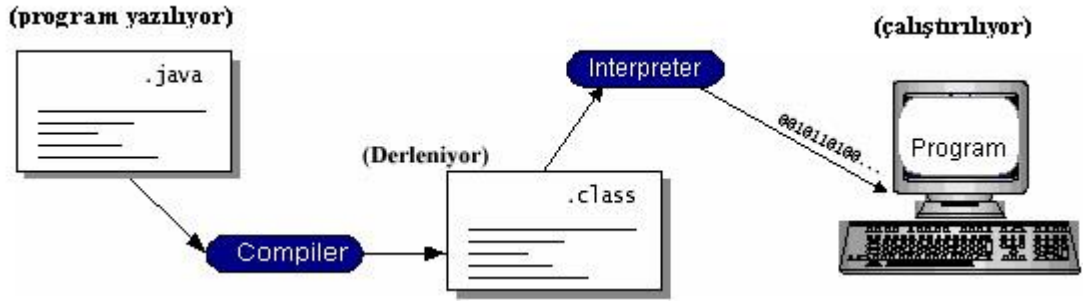
Derleyici ile Yorumlayıcının Farkı

Yorumlayıcıda kodun okunması ve çevrilmesi programın çalışması sırasında yapıldığından hız düşüktür. Ayrıca yorumlayıcı yalnızca karşılaştığı ilk hatayı rapor edebilir. Bu hata düzeltildiğinde sonraki çalışmada da program ancak bir sonraki hataya kadar ilerleye bilir. Oysa derleyici kaynak kodundaki bütün hataları bulabilir. Buna karşılık hata ayıklama işlemi yorumlayıcılarla daha kolaydır. Ayrıca derleyicilerle gelen bazı sınırlamaların kalkması nedeniyle daha esnek bir çalışma ortamı sağlanır. Son yıllarda iki yöntemin üstün yanlarını birleştiren karma diler (Java ve Python gibi) öne çıkmaya başlamışlardır.

Derleyici, programın tamamını kontrol eder bir satırında hata varsa program satırları bittikten sonra hatayı gösterir. Yani kaynak kodu bir defa çevirir. **Yorumlayıcı** ise programın bir yerinde hata varsa çalışırken hatalı satıra geldiğinde durur ve hata mesajı verir. Hata düzeldikten sonra tekrar birinci satırdan itibaren kodu çevirir.

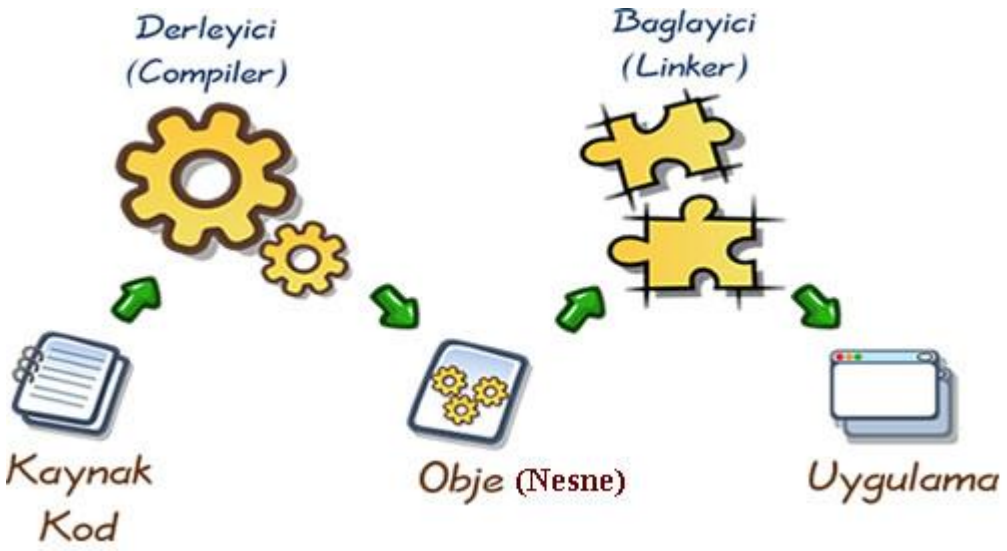
Hem derleyici hem de yorumlayıcı tekniği birlikte de kullanılabilir. Bu tip çalışmada kaynak kodu sanal bir bilgisayarın makine koduna (bytecode) çevrilir ve daha sonra bu sanal bilgisayarı gerçekleyen bir program yardımıyla yorumlanarak çalıştırılır. Örneğin Java dilinde yazılmış bir kaynak kodu önce Java derleyicisinden geçirilerek Java Sanal makinesinin (**Java Virtual**

Machine- JVM) makine koduna dönüştürülür; sonrada bu sanal makineyi gerçekleyen bir Java çalışma ortamı (**Java Runtime Environment** – JRE) yardımıyla çalıştırılır. Bu işlemin nasıl yapıldığı aşağıda grafiksel olarak gösterilmiştir.



Bağlama (Linking)

Günümüzde yazılan birçok program, genellikle programlama dili içerisinde bulunan kütüphane fonksiyonlarına ve yordamlara bağlantı içerdiğinden kaynak programdan üretilen amaç program kodunun (obje) bilgisayarda çalıştırılmaya hazır olması için bu amaç program (object code) bağlama (linking) işlemine tabi tutulur. Bu bağlama işlemini de Bağlayıcı (Linker) denilen yazılımlar yapmaktadır. Bu işlem sonucu eksik parçaların program kodu içerisine eklenmesi ile artık yazılan program (uygulama dosyası) çalışmaya hazır hale gelmiş olur.

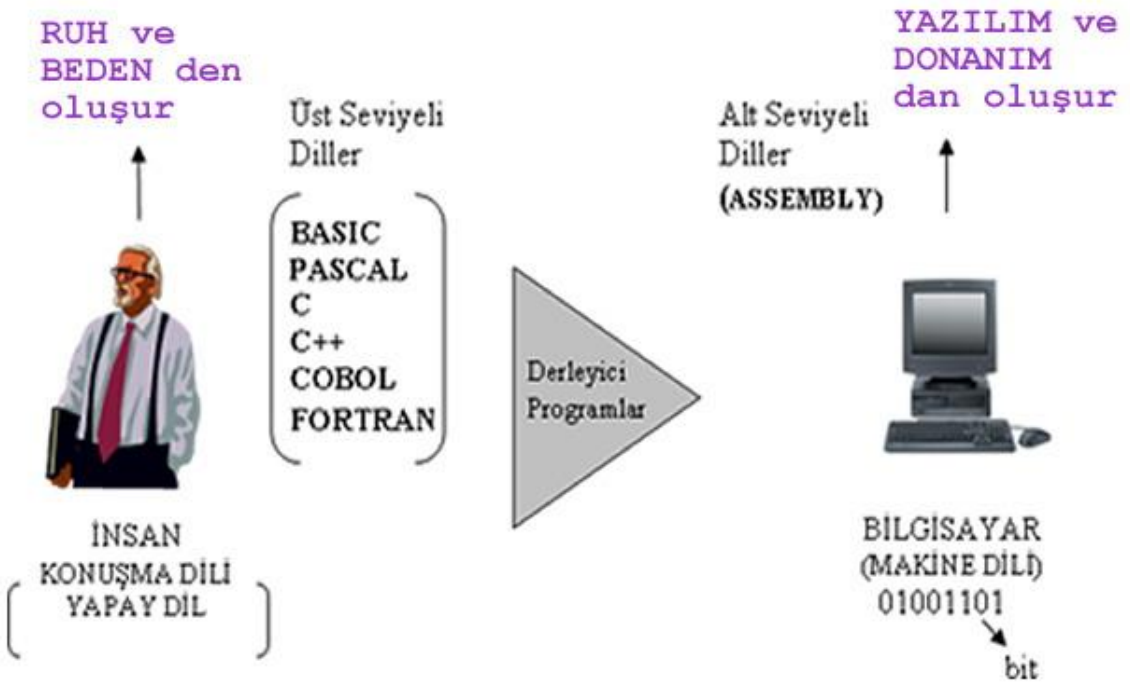


PROGRAMLAMA DİLLERİ VE ÖZELLİKLERİ

Bir **programlama dili**, programcının bir bilgisayara ne yapması gerektiğini anlatmasının standartlaştırılmış bir yoludur.

Bir programcı komutları farklı programlama dilleri kullanarak yazabilir. Günümüzde yüzlerce programlama dili mevcuttur. Bu dilleri Şekil 1. de gösterildiği gibi insanın algılamasına veya makineye yakın olmasına göre 3 gruba ayırabiliriz;

1. Alçak seviyeli diller (Assembly dili)
2. Orta Seviyeli Diller (C, Ada dili)
3. Yüksek Seviyeli Diller(Visual Basic, Pascal, Delphi, Cobol, FoxPro, Java gibi..)



Şekil 1. Programcı ile Makine (PC) arasındaki İlişki

NOT: Programlama, temel olarak çözüm algoritması geliştirme işlemi, bilgisayar programı ise bir programlama dili ile yazılmış algoritmadır.

Programlama dilleri kavramı, günümüzde kabına sığamamış cep telefonlarından kol saatlerine kadar tüm işlem birimlerinde (Mikroişlemci-CPU, Mikro denetleyici-MCU, Sayısal İşaret İşleyici-DSP …) bulunan elektronik cihazlarda kullanım alanı bulmuştur. Şüphesiz bu yönelim, programlanabilir bir elektronik aygıtın özelleştirilebilme büyüsünden ileri gelmektedir.

Programlama dilleri, daha sonraları yalnız programlama kavramı içerisine sıkışıp kalmadı. Grafik Kullanıcı Arabirimi (GUI) tanımlama, veritabanı sorgulama gibi alanlarda da kullanım bularak uygulama geliştiricilerin daha çok işlem yapabilen programları, daha kısa sürede yazmalarına imkân tanıdı. Programalama dilleri ise artık yalnızca uygulama geliştirmek için değil uygulamaları özelleştirmek için de sıkça kullanılır hale geldiler.

Günümüzdeki birçok modern programlama dilinin temeli, hiç şüphesiz Zuse'un Plankalkül hesaplama dili ve IBM tarafından geliştirilen Fortran'ın öncülüğünden sonra uygulama sahası bulmuştur. Programlama dillerinin tarihsel gelişimindeki FORTRAN, ALGOL, LISP, Pascal, APL, SIMULA, CPL, Ada, Smalltalk, BASIC, C gibi büyük atlama taşlarının bir çoğu günümüzde şekil değiştirmek suretiylede olsa güncelliğini korumaktadır. Bunların yanında daha yakın bir tarihte geliştirilmiş olan, bu dillerin torunları olarak gösterebileceğimiz Java, C++, C#, Python, Ruby, Perl, Delphi, Visual Basic, PHP gibi diller de sahnede kendilerini göstermiş ve programlama dili pastasında önemli bir pay sahibi olmuşlardır.

Bazı Programlama Dilleri Ve Tarihçesi

ASSEMBLY

Assembly programlama dili, çoğu zaman özel alanlarda geliştirilen yazılımlarda kullanılan alt düzey bir yazılım dili olarak tanımlanır. Bu dilin komutları, bilgisayarın doğrudan işlettiği makine dili komutlarının birebir karşılığıdır. Bu nedenle bu dil için makine dili de denilebilir. Her ne kadar uzman programcıların özel alanlarda kullandığı bir dil olarak tanımlansa da, programcılar istedikleri takdirde her türlü uygulamayı bu dil ile geliştirebilirler ya da kullandıkları üst düzey dil altından çağırabilecekleri procedure'ler yazabilirler. Assembly diliyle yazılmış bir program **assembler** derleyicisi ile makine diline çevrilir.

Makine dili kodları, işlemciye veya sanal makineye özel kodlardır. Okunulabilir (human readable) değildirler. Okunulabilir olmaları için bu dillere ait Assembly dilleri oluşturulur. Günümüzde birçok sanal makine de (Java Sanal Makinesi gibi) kendi makine kodlarını kullanmaktadırlar.

Aşağıdaki gösterimden de anlaşılacağı üzere, Makine dili gösterimi, hexadecimal (Onaltılı) sayılardan oluşurken, Assembly dili gösterimi ise komutların ingilizce kısaltmaları (mnemonic ifadelerden) oluşmaktadır.

Makine dili + Assembly gösterimi;

```
55        push    ebp
89 e5     mov     esp, ebp
5d        pop     ebp
c3        ret
```

ALGOL (ALGOrithmic Language)

ALGOL, evrensel bir dil oluşturma çabalarının sonucu olarak doğdu. 1950'li yılların ortasında FORTRAN'a ait yanılgıların tekrar yaşanmaması için tasarlandı. 1958'de Avrupalı ve Amerikalı bilgisayar bilimcilerin Zürih'te toplanması neticesinde geliştirildi. Zaman içerisinde birçok Algol türevi ve sürümü uyarlandı. ALGOL 58, ALGOL 60 ve ALGOL 68 gibi başlıca sürümlerinin yanında, 1974 yılına kadar 19 adet farklı ALGOL uyarlaması geliştirilmiştir.

ALGOL 68'e ait örnek “merhaba dünya” programı;

```
begin
    print(("Merhaba Dünya!", newline))
end
```

ALGOL'da ifade blokları begin ve end arasına alınmaktaydı. Bu yaklaşım daha sonra bir çok programlama dilinin temel felsefesi olacaktır.

ALGOL 58'in imla tanımlamalarını gerçekleştirmek için John Backus, BNF (Bachus Normal Form)'u geliştirdi. Daha sonra Peter Naur, ALGOL 60 için bu imla tanımlama şeklini gözden geçirip yeniledi. Bu güncellemenin üzerinde Donald KNUTH'un önerisi ile gösterim şeklinin adı Bachus-Naur Form olarak değiştirildi.

BASIC (**B**eginner's **A**ll **P**urpose **S**ymbolic **I**nstruction **C**ode)

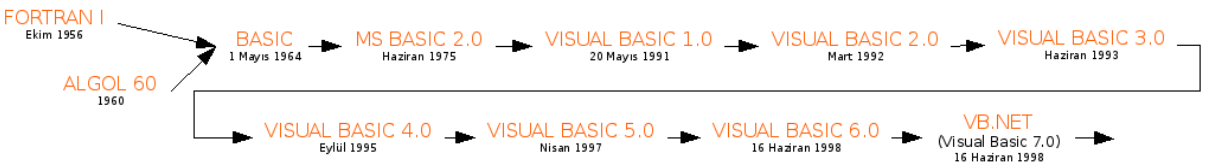
Basic dili 1964 yılında Jhon Kemeny ile Thomos Kurtz tarafından Darthmouth College Üniversitesinde geliştirilmiştir. Eğitim amaçlı bir programlama dilidir. Mikrobilgisayarlar için ilk Basic uyarlamasını Microsoft'un kurucuları olan Paul Allen ile Bill Gates yazmıştır. BasicA, GWBasic, QBasic, Commodore Basic V2, Turbo Basic gibi değişik sürümleri çıkmıştır.

BASIC dili 8 temel amacı gerçekleştirmek üzere tasarlanmıştır:

1. Yeni başlayan kullanıcılar için kolay olmalı.
2. Genel amaçlı bir programlama dili olmalı
3. Uzmanlar için gelişmiş özellikler de eklenebilmeli
4. Etkileşimli olmalı
5. Açık ve kullanıcı dostu olmalı (hata iletileri sağlamalı).
6. Küçük programlar için hızlıca cevap vermeli
7. Kullanırken bilgisayar donanımı bilgisine gerek olmamalı
8. Kullanıcıyı işletim sisteminden soyutlamalı

VISUAL BASIC

Microsoft tarafından, Basic programlama dili üzerinde geliştirilmiş, hareket bağımlı, nesne tabanlı ve görsel bir programlama dilidir. Visual Basic yapısal bir programlama dili olan Basic dilinden türetilmiş olmasına rağmen hareket bağımlı bir programlama (event-driven programming) dilidir. Yani bir kodun çalışması için bir olayın veya hareketin olması gerekir. İlk sürümü 1991 yılında Visual Basic 1.0 olarak çıkmıştır. Daha sonraki yıllarda 2.0 (1992 yılında),3.0 (1993 yılında), 4.0 (1995 yılında), 5.0 (1997 yılında) , 6.0 (1998 yılında) sürümleri çıkmıştır.



C

C dili, AT&T Bell laboratuarlarında 1969 ve 1973 yılları arasında, Ken Thompson ve Dennis Ritchie tarafından UNIX İşletim Sistemi için geliştirilmiş bir programlama dilidir. C, günümüzde neredeyse tüm işletim sistemlerinde kullanılan, dünyanın en çok kullanılan sistem programlama dilidir. Ancak, uygulama programları yazmak için de çok sık kullanılır. C ++ da, C den türemiş bir dildir.

C++

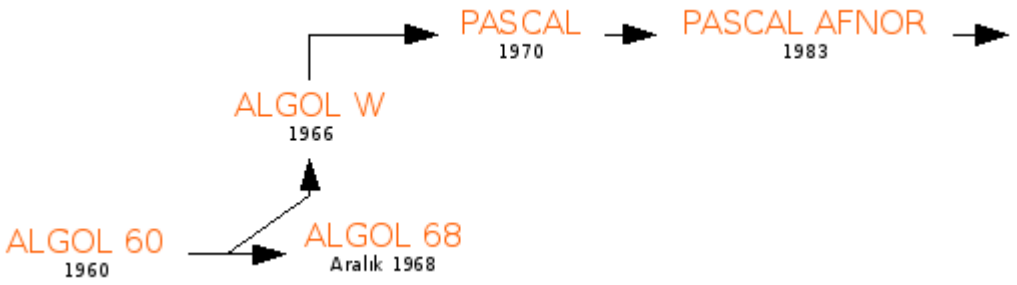
Nesneye yönelik bir programlama dilidir. 1980'lerin başında Bjarne Stroustrup tarafından geliştirilen C dilinin birçok temel özelliğini destekleyen ve nesne yönelimli programlamaya olanak sağlayan, sınıflar sayesinde yeni veri türlerinin oluşturulduğu çok yaygın olarak kullanılan programlama dilidir. C++ (si-plas-plas okunur) genel amaçlı bir programlama dilidir. İlk olarak C de bulunmayan sınıf kavramı eklendiği için ***C With Classes*** olarak adlandırılmış, daha sonra C'deki herhangi bir sayısal değişkenin değerini bir arttırmaya yarayan ve özellikle döngü yapılarında çok sık kullanılan 'i++' ifadesine benzer biçimde ***C***++ olarak adlandırılmıştır.

Resim:tarih_cpp.png

PASCAL

1970 Yılında Niklous Wirth tarafından öğrencilerine programlama öğretmek için geliştirilmiştir. Bilgisayar bilimcisi Niklaus Wirth Pascal'ı 1970'te yapısal programlamayı derleyiciler için daha kolay işlenir hale getirebilmek amacıyla geliştirmiştir. Adını matematikçi ve düşünür Blaise Pascal'dan alan Pascal dili, Algol programlama dilinden türemiştir. Wirth, Pascal'dan başka Modula-2 ve Oberon programlama dillerini de geliştirmiştir. Bu diller Pascal'a benzerler ve ayrıca nesneye yönelik programlamayı da desteklerler.

İlk Macintosh işletim sistemi, büyük ölçüde Pascal kullanılarak yazılmıştır. Pascal dili derleyicilerinin çoğu yine Pascal programlama dili ile yazılmıştır.



DELPHI

Temeli Pascal dilidir. Özellikle nesne yönelimli programlama anlayışıyla yapılandırılmış Turbo Pascal dilinin görsel sürümü denilebilir. Nesne, sınıf, kalıtım, fonksiyon, aşırı yükleme (overloading) gibi temel nesneye yönelik programlama tekniklerini ve daha fazlasını içeren ve C++ den aşağı kalmayan güçlü ve esnek bir programlama dilidir

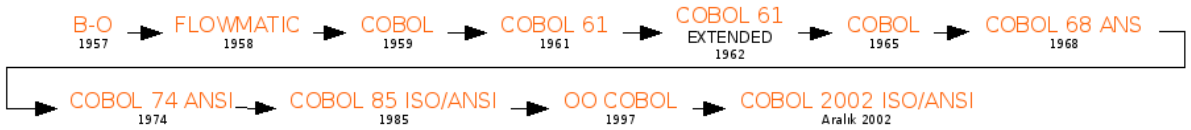
FORTRAN(**For**mula **Tran**slater)

1954'de IBM tarafından üretilen IBM 704 için ilk sürümü *John Backus* ve ekibi tarafından geliştirilmiştir. Fortran ilk yüksek düzey programlama dili olmasa da 1950'deki yüksek programlama dilleri derlenmeden, bir çevirici (interpreter) yardımıyla çalıştırılıyordu. Bu da makine koduyla yazılan programlardan en az 10 kat daha yavaş çalışmalarına sebep oluyordu. İşte bu noktada **Backus** ve ekibi hem yüksek programlama dilleri gibi kolay yazılabilen hem de makine kodunda yazılmış gibi hızlı çalışan bir programlama dili sözüyle **Fortran** 'ı tanıttılar. Her ne kadar ilk derlenebilir yüksek düzey dilin Fortran olup olmadığı hala tartışma konusu olsa da, Fortran geniş kitleler tarafından kullanılmış ilk yüksek düzey derlenebilir dildir. İlk Fortran sürümü Fortran 0 'dır. Fortran I, II, III,IV,77,90,95,2000 ve 2003 gibi farklı sürümleri bulunmaktadır. Matematiksel, Bilimsel İşlemlerde kullanılmak üzere tasarlanmıştır.

COBOL(COmmon Business Oriented Language)

1959 Yılında Ticari Amaçlar ile kullanıcı tarafından geliştirilmiştir. Ticari bir programdır veri tabanına dayanır.

COBOL 2002 standardı ile nesne modelli yaklaşımın yanında, birçok modern özellik bünyesine katılmıştır. COBOL'un temelleri 1959 yılında atılmış ve bu tarihlerden sonra COBOL-68, COBOL-74, COBOL-85 ve COBOL-2002 standartları ile günümüze kadar ulaşmıştır.



COBOL günümüzde popülerliğini nispeten yitirse de çağın gerisinde kalmamıştır. Çeşitli yazılımlarla .NET çatısı ile entegre çalışabilmektedir. Bu sayede bu çatının tüm kütüphanelerini de kullanabilmektedir.

LISP

1950'li yılların sonunda yapay zeka alanındaki artan uygulama gereksinimi sonucunda liste işleme özelliklerini sağlayan ilk işlevsel dil; **LISP** ortaya çıktı. IBM (international Business Machines), 1950'lerin ortasında yapay zeka alanına el attı ve Fortran'a ek olarak FLPL (Fortran List Processing Language) dilini geliştirdi. MIT'den John McCarthy, 1958'de IBM Araştırma Departmanı'nda sembolik hesaplamalar üzerine çalışıyor ve bu hesaplamaları yapmak için gerekenler listesi geliştiriyordu. Çalışma neticesinde, birçok eksiklik gözüne çarptı. Bu eksiklikler, ne Fortran ne de uzantısı olan FLPL dili ile gideriliyordu. McCarthy, MIT'ye döndüğünde yeni bir proje başlattı ve proje neticesinde LISP (LISt Processing) dili ortaya çıkmış oldu.

LISP, günümüze kadar ulaşmış olan köklü programlama dillerinden birisidir. Halen bir çok uygulama geliştirici bu dili kullanırken EMACS gibi bazı yazılımlar da uygulamaya yönelik betik bir dil olarak LISP'i kullanmaktadır. Halen çok kullanılan LISP türevleri, Common Lisp ve Scheme dilleridir.

LISP dilinin yapısı gereği hemen her işlev o alana özel tanımlanmış işlevler sayesinde yapılmaktadır. Çoğu işlev bloğu parantez içine alınmış değerlerden oluşur. Bu değerlerin ilki işlev belirtecidir. Devamındaki değerler ise işleve ait parametrelerdir.

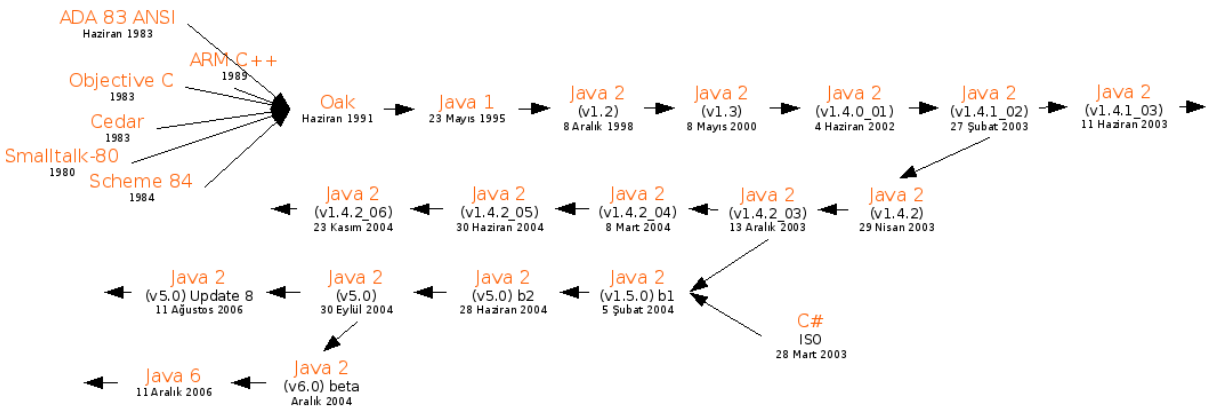
JAVA

Sun Microsystems mühendislerinden **James Gosling** tarafından geliştirilmeye başlanmış gerçek nesneye yönelik, platformdan bağımsız, yüksek performanslı, çok işlevli, yüksek seviye, adım adım işletilen bir dildir. İlk sürümü 1996 yılında çıkmış olup, 1.1,1.2,1.3,1.4 ve 5.0,6.0 gibi sürümleri geliştirilmiştir.

Java ilk çıktığında daha çok küçük cihazlarda kullanılmak için tasarlanmış ortak bir platform dili olarak düşünülmüştü. Ancak platform bağımsızlığı özelliği ve standart kütüphane desteği C ve C++'tan çok daha üstün ve güvenli bir yazılım geliştirme ve işletme ortamı sunduğundan, hemen her yerde kullanılmaya başlanmıştır. Şu anda özellikle kurumsal alanda ve mobil cihazlarda son derece popüler olan Java özellikle J2SE 1.4 ve 5 sürümü ile masaüstü uygulamalarda da yaygınlaşmaya başlamıştır.

Java, 13 Kasım 2006 da Java platformu GPL lisansıyla açık kodlu hale gelmiştir. Java'nın ilk sürümü olan Java 1.0 (1995) Java Platform 1 olarak adlandırıldı ve tasarlama amacına uygun olarak küçük boyutlu ve kısıtlı özelliklere sahipti. Daha sonra platformun gücü gözlendi ve tasarımında büyük değişiklikler ve eklemeler yapıldı. Bu büyük değişikliklerden dolayı geliştirilen yeni platforma Java Platform 2 adı verildi ama sürüm numarası 2 yapılmadı, 1.2 olarak devam etti. 2004 sonbaharında çıkan Java 5, geçen 1.2, 1.3 ve 1.4 sürümlerinin ardından en çok gelişme ve değişikliği barındıran sürüm oldu. **Java SE 11**(JDK 11.0) ise Oracle firmasının Eylül 2018 tarihli çıkardığı ilk LTS (**Long-term support**) sürümüdür.

Java dilinin farklı sürümleri ve çıkış tarihleri aşağıdaki şekilde gösterilmiştir.



JAVA PROGRAMLAMA DİLİNİN ÖZELLİKLERİ

Java dilinin temel özelliklerini sıralarsak;

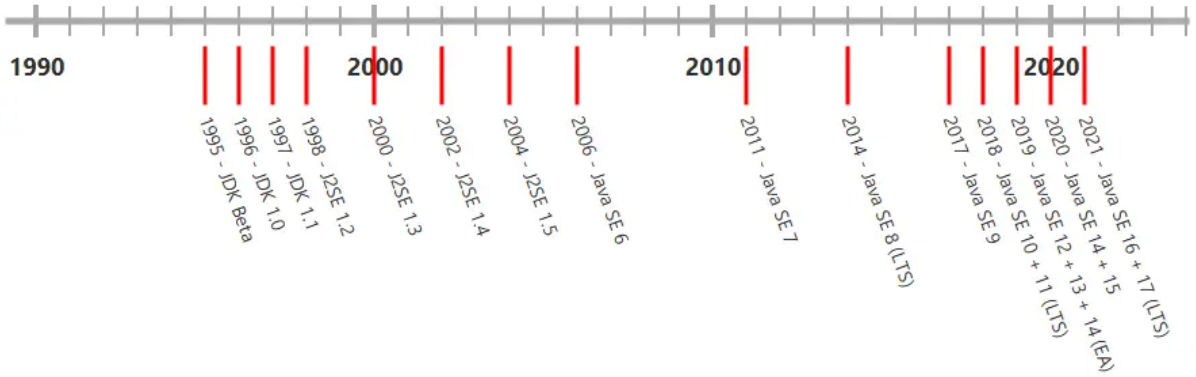
Basit (Simple)	Java dili kısa ve özlü komut yapısına sahip, öğrenmesi kolay bir dildir. Ayrıca C,C++ diline benzerliği nedeni ile bu dillere aşina olanlar için Java'da kod yazmak kolay olacaktır.
Güvenli (Secure)	Java programlama dili ile hazırlanan programlar güvenliğin çok önemli olduğu uygulamalarda (internet uygulamaları gibi) tercih edilmektedir. C, C++ gibi dillerden daha güvenlidir. Sadece derleme esnasında değil aynı zamanda yükleme(çalıştırma) esnasında da Byte-code doğrulaması yapılır.
Taşınabilirlik (Portable)	Java teknolojisinin en önemli özelliği her ortamda çalışabilmesidir. JVM (Java Sanal Makinesi) nin kurulduğu her ortamda (Java platformunu destekleyen her türlü ortamda) çalışır.
Nesne Yönelimli (Object Orient)	Nesne yönelimli bir programlama dilidir. Java'da her şey ya nesnedir ya da bir nesnenin parçasıdır.

Sağlam (Robust)	Java sağlam bir dildir. Programlamadaki hataların çoğu daha yazılım aşamasında anlaşılabilmektedir. Yazılım aşamasında yakalanamayan hatalar, çalışma zamanında (run-time) düzeltilebilmektedir.
Çoklu kullanım (Multithreaded)	Java dili çoklu kullanım (multithreaded) programlama desteği sağlar. Çoklu kullanım, program içinde birden fazla işlemin çalışabilmesi demektir.
Mimariden bağımsız (Architecture-neutral)	Bir belirli makine ya da işletim sistemi mimarisine Java programlama dili bağlı değildir.
Yorumlayıcı (Interpreted)	Java dili hem derleyici hem yorumlayıcı kullanır.
Yüksek performans (High performance)	Java diğer dillere nazaran nispeten yavaş olmakla birlikte, Java bytecode'u çalışma hızı için en iyi şekilde optimize edilmektedir.
Dağıtık (Distributed)	'Dağıtık' birden fazla bilgisayarda çalışan programların bir biriyle uyumlu çalışabilmesidir. Bir yazılım parçasının bir kısmının bir makinede diğerinin başka makinede aynı anda çalışması mümkündür. (Özellikle internet uygulamalarında Java tercih edilmektedir.
Dinamik (Dynamic)	Java da bir programın birimlerinin (kütüphaneler, modüller veya sınıfların) birbirine bağlanması çalışma zamanında (run-time) yapılır. Kullanılan birimlerin içyapısı değiştirildiğinde, bu birimleri kullanan programın değişmesi gerekmez
Ücretsiz	Java programını ve her türlü yazılımını internet adresinden (www.java.sun.com) ücretsiz edinebilirsiniz.

Java, kendini çok hızlı bir şekilde yenileyen bir dil, sürekli yeni versiyonları çıkmakta ve her çıkan sürümle birlikte dile yeni özellikler eklenmektedir. Bu programcılar için hem dezavantaj hem de avantaj sağlamaktadır. Dezavantajı, programcının sürekli yeni özellikleri öğrenmek zorunda kalması ama bu dezavantajı java dili geriye uyumluluk ile avantaja dönüştürmektedir.

Yani 1.4 sürümündeki bir komut, 1.8 sürümünde de çalışmaktadır. Java'nın mevcut sürümleri ve aralarındaki farkları aşağıdaki linkten inceleyebilirsiniz;

https://javaalmanac.io/

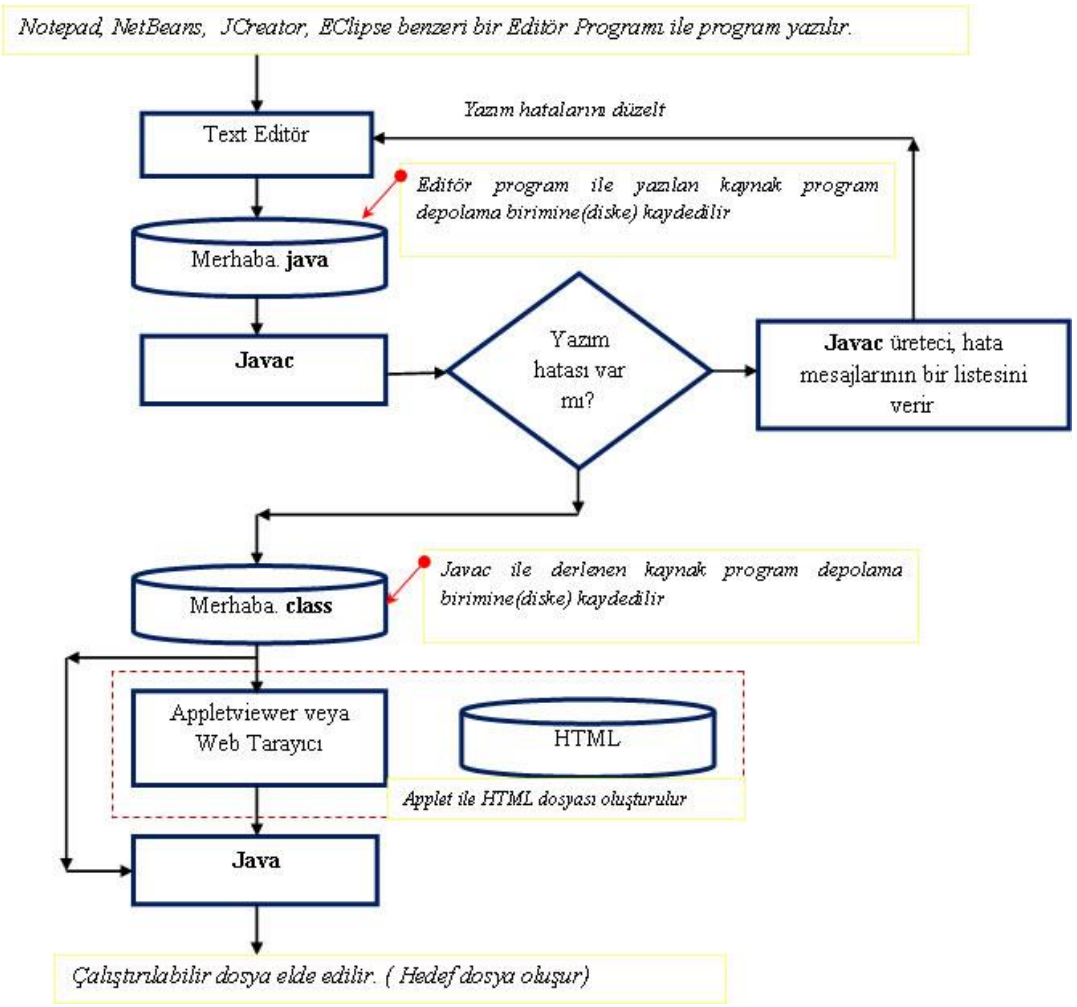


(Kaynak: https://foojay.io/today/openjdk-vs-openjfx-release-cycles/)

JAVA PROGRAMLAMA DİLİNİN YAPISI

Bir Java programını çalıştırabilmek için Sun'ın web adresinden (http://java.sun.com/javase/downloads) en son sürümünü indirebilirsiniz (download edebilirsiniz). Örnek olarak Java 6.0 sürümünü indirebilirsiniz.

Aşağıdaki şema Java programlama dilinde yazılan bir programın derleme aşamalarını göstermektedir.



İlk Java Programımız;

Bir Java programını (örneğimizde **Merhaba.java** isimli) farklı yöntemlerle çalıştırabiliriz.

I.Yol : Not defteri ile..

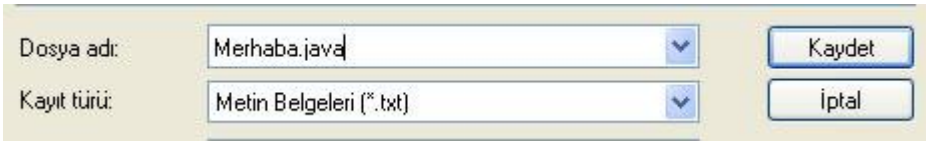
1.adım: Metin editöründe (Not defteri) ilk Java programını yazıyoruz.

Merhaba - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

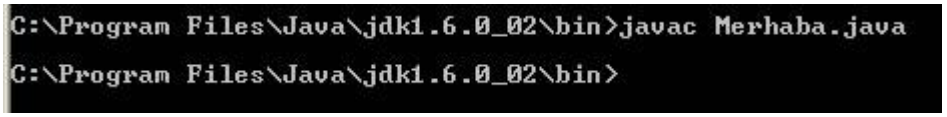
```
public class Merhaba {

  public static void main(String[] args) {
      System.out.println ("Merhabalar,bu İlk JAVA Programımız");
}
}
```

2.adım: Yazdığım programı **javac** derleyicisinin bulunduğu klasör içerisine **“.java”** uzantılı kaydediyoruz. Bunun için;
Dosya -> Farklı Kaydet -> **“C:\Program Files \ Java\jdk1.6.0_02\bin”** klasörünü seçiyoruz **ve Merhaba. java** ismiyle kaydediyoruz.



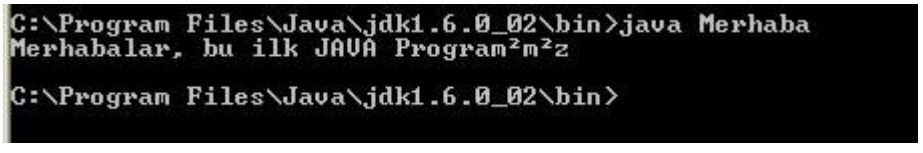
3.adım: MS-DOS komut istemine geçiyoruz (Windows için**, Başlat-> Run -> CMD** veya **Linux** için **Terminal** ile). Programı derlemek için **javac** isimli derleyicinin bulunduğu yolu yazıyoruz. (Örnek olarak “**C:\Program Files \ Java\jdk1.6.0_02\bin”** klasörünün içerisinde**)**

4.Adım: javac Merhaba.java komut satırı ile java uzantılı programı derliyoruz.



Programda bir yazım hatası yoksa derlenecektir. Bu aşamada “**Merhaba.class**” dosyası oluşmuş olur.

4.adım: class dosyasını oluşturduğumuz dosyayı son aşama olarak java komutu ile (***java Merhaba***) çalıştırabiliriz.

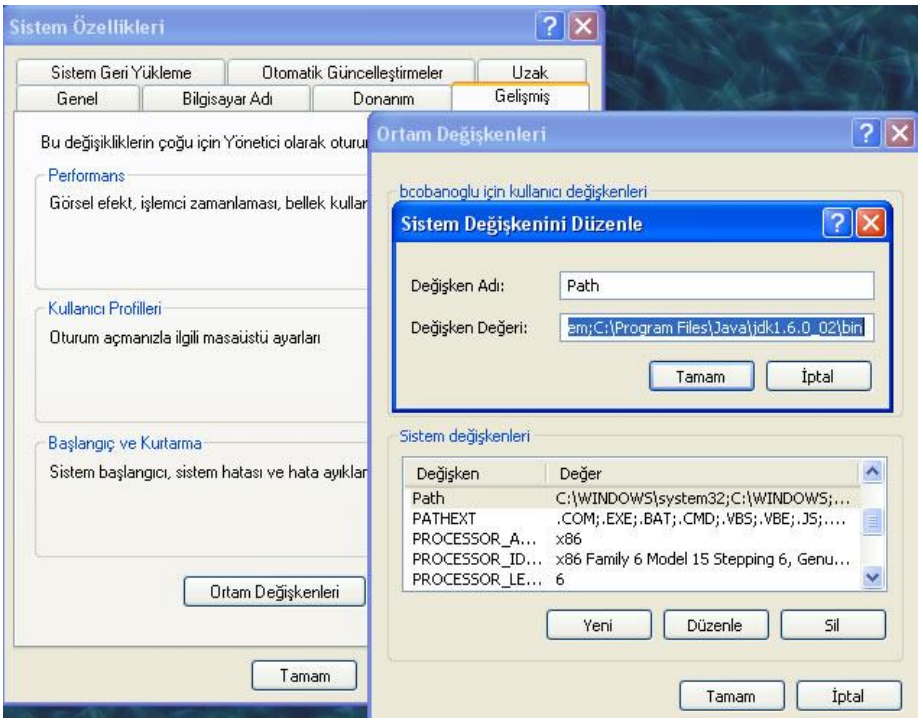


Programı çalıştırdığımızda ekrana “**Merhabalar, bu ilk JAVA Programımız**” mesajını aldık

Not: (Windows kullanıcıları için)

Her defasında MS-DOS komut isteminde programı derlemek için javac isimli derleyicinin bulunduğu yolu yazmamak (“C:\Program Files \ Java\jdk1.6.0_02\bin”) için ;

Masaüstündeki Bilgisayarım (My Computer) simgesine sağ(ters) tıklama yapılır ; Özellikler (Properties) -> Gelişmiş (Advanced) -> Ortam Değişkenleri (Environment Variables) seçilir ve PATH (yol) tanımı “**; C:\Program Files \ Java\jdk1.6.0_02\bin**” şeklinde yapılır. Artık sadece MS-DOS komut isteminde “c:\> **javac dosya_adi.java”** yazarak programınızı derleyebilirsiniz.



2.yol: Bir Java Editör programı ile

Eğer MS-DOS komut istemi ile uğraşmak istemiyorsanız, yazdığım kod direkt çalışsın diyorsanız, bir Java editörü kullanmanızı tavsiye ederim. Sık kullanılan bazı Java editörleri ve ücretsiz indirebileceğiniz web adresleri aşağıda verilmiştir.

- Eclipse (http://www.eclipse.org/downloads)
- NetBeans (http://www.netbeans.info/downloads)
- IntelliJ IDEA (https://www.jetbrains.com/idea/)
- BlueJ (http://www.bluej.org/download/download.html)
- jGRASP (https://www.jgrasp.org/)

Eclipse Editörünün Kurulumu ve Program Yazımı

Eclipse, Java geliştiricilerinin en çok tercih ettiği editörlerden biridir. Windows, Linux ve Mac OS gibi tüm işletim sistemlerine kurulabilir. Sadece yapmanız gereken; öncelikle www.eclipse.org adresine girip, **Downloads** sekmesinden işletim sisteminiz için uygun olan en güncel Eclipse sürümünü indirmeniz ve sıkıştırılmış haldeki dosyayı açmanızdır. Tabii Eclipse programını kurmadan önce öncelikle **JDK**(Java Development Kit-Java Geliştirme Kiti) programının en son sürümünü bilgisayarınıza kurmanız gerektiğini unutmayınız.

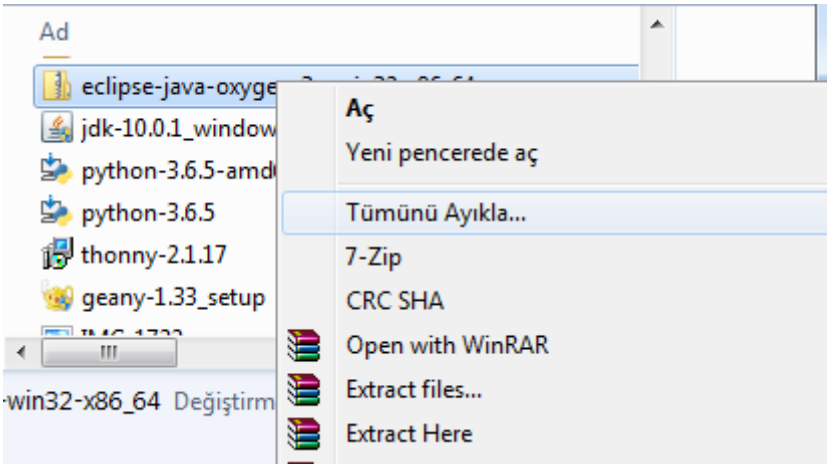
Eclipse Editörünün Kurulumu

1. Eclipse programını kurmadan önce Java için öncelikle JDK(Java Development Kit-Java Geliştirme Kiti) programının en son sürümü bilgisayara kurulmalıdır. Örneğin JDK nın **en son sürümü**nü (*Haziran 2018 itibari ile **Java SE 10.0.1** idi*) aşağıdaki linkten (bilgisayarınızda kurulu olan **işletim sistemine dikkat ederek)** indirip bilgisayarınıza kurabilirsiniz; **http://www.oracle.com/technetwork/java/javase/downloads/index.html**

2. Eclipse kurulumu için **www.eclipse.org** adresinden, **Downloads** sekmesine tıklamak gerekir.

3. Download(İndirme) sayfasında, **İşletim sisteminize uygun (Windows 64 bit gibi) olan** "**Eclipse IDE for Java Developers**" seçilir. Linki: **http://www.eclipse.org/downloads/eclipse-packages/** .

4. Bilgisayara indirilen sıkıştırılmış dosya açılır. Açmak için aşağıdaki gibi dosya üzerine fare(mouse) ile sağ tuş yapılır "**Tümünü ayıkla…**" ya da Winzip, Winrar, 7-Zip gibi bir program ile "**Extract Here/Burada çıkart**" seçenekleri tercih edilebilir.

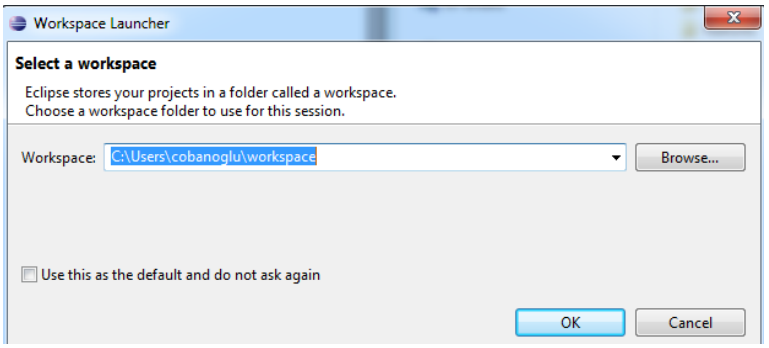


5. Sıkıştırılmış dosyayı açtıktan sonra artık klasör içerisindeki "**eclipse**" isimli dosyaya tıklayarak programı çalıştırabilirsiniz.

Eclipse ile Merhaba Dünya Uygulaması

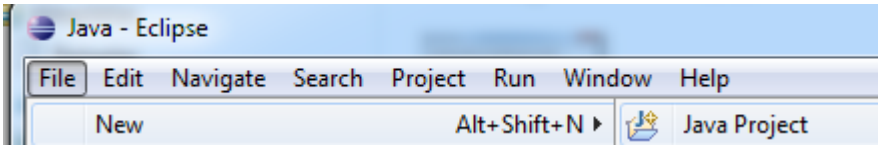
1. Bilgisayarınıza indirmiş olduğunuz sıkıştırılmış haldeki Eclipse dosyasını açtıktan sonra artık klasör içerisindeki "**eclipse**" isimli dosyaya tıklayarak programı çalıştırabilirsiniz.

2. İlk olarak yazacağınız Java uygulamalarının nerede saklanması gerektiğini gösteren bir menü çıkacaktır. "**OK**", tamam diyebilirsiniz veya değiştirebilirsiniz.

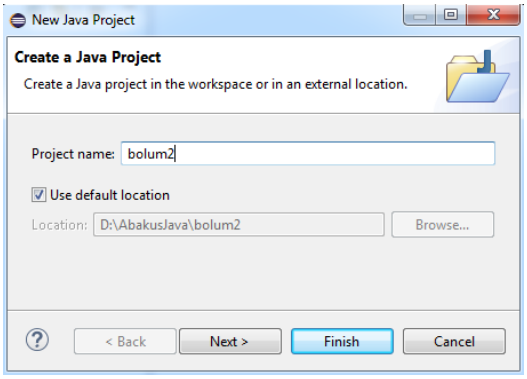


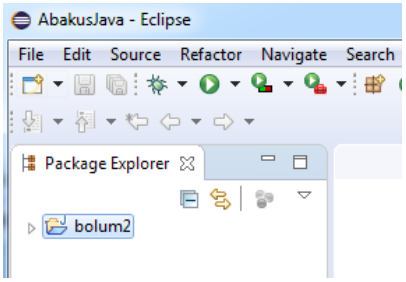
3. İlk kurulumda sizi bir "**Welcome**-Hoşgeldiniz" penceresi karşılayacaktır.

4. Bu pencereyi kapattıktan sonra ilk programımızı artık yazabiliriz. **File >> New >> Java Project** menüsünden proje klasörümüzü belirliyoruz.



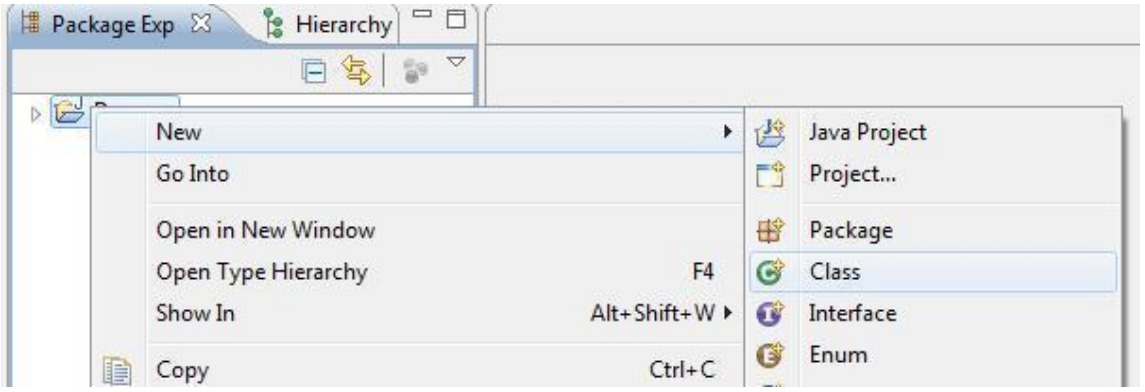
5.“**Project Name**” kısmına projenizin ismini (**bolum2** gibi) yazıp, “**Finish**” e tıklıyoruz.



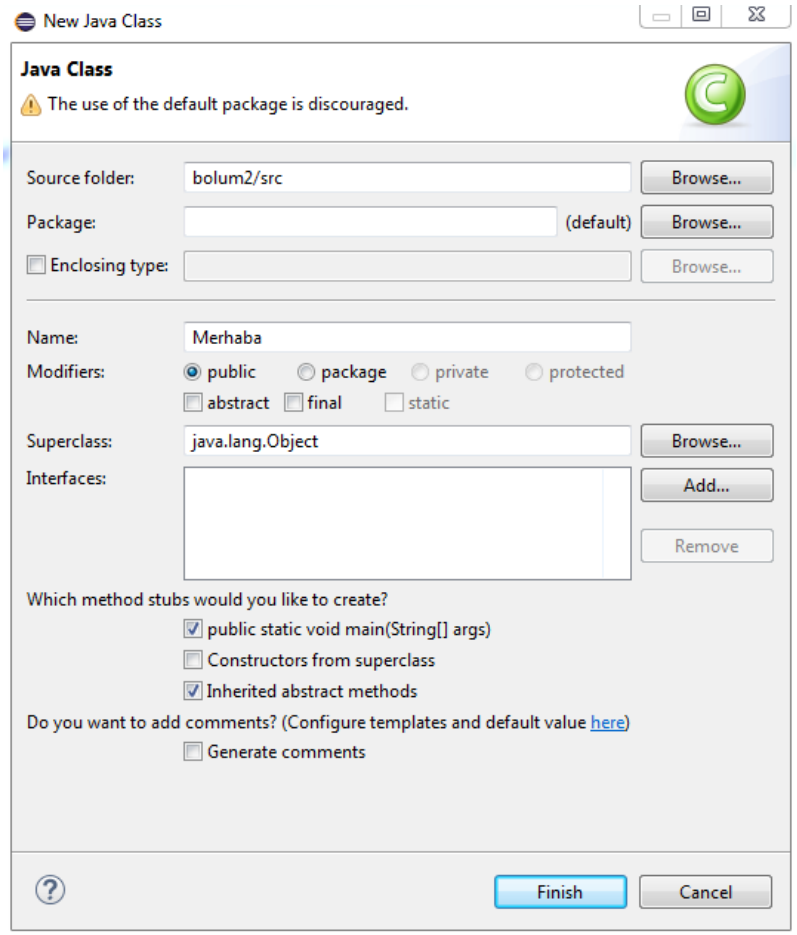


Bu işlemden sonra sol panelde ‘Package Explorer’ başlığı altında “**bolum2**” klasörünün oluştuğunu görmemiz gerekir.

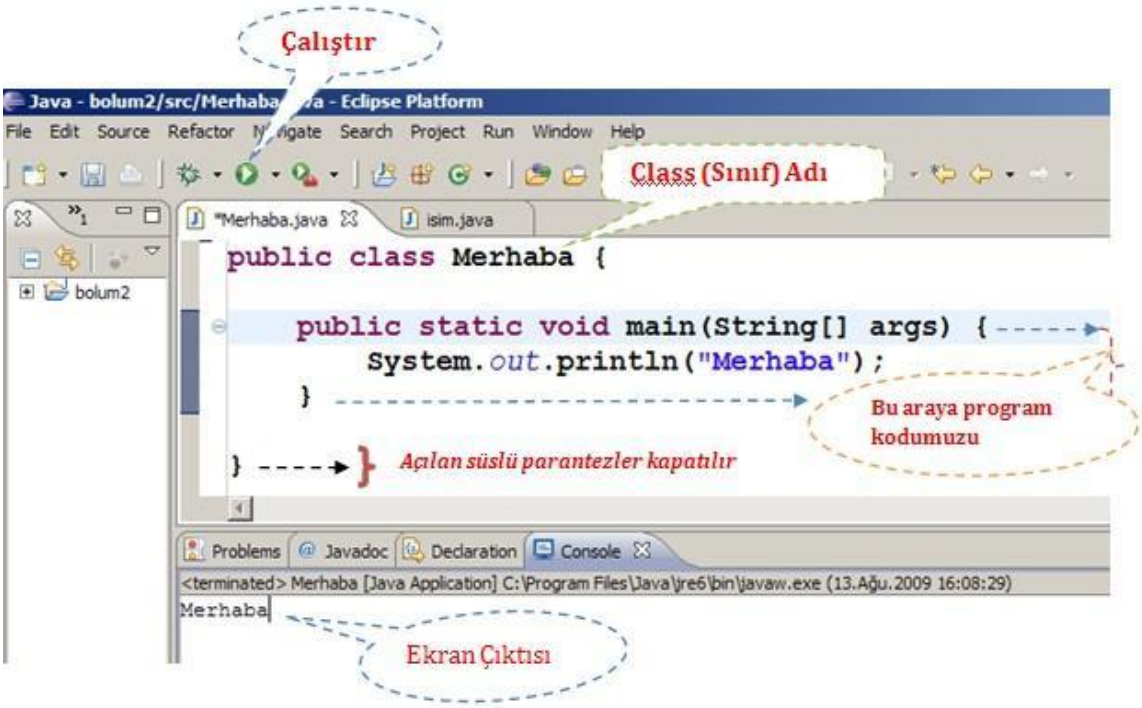
6. Proje klasörü üzerine gelip fare ile **sağ tuş** yapıp, "**New" >> "Class"** menülerini seçiyoruz.



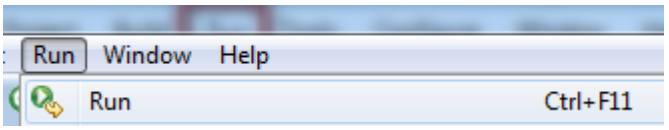
7. "**Name "** kısmına dosya/sınıf adını ("Merhaba") yazıp, "**public static void main**" e çek işareti koyuyoruz.



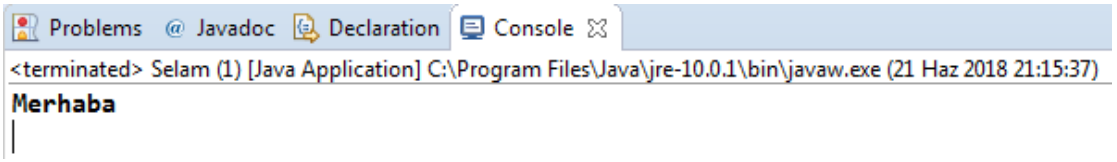
8. Gelen pencereye artık kodumuzu {`System.out.println("Merhaba");`} yazabiliriz.



9. Yazdığınız programı 'Çalıştır' simgesinden veya **Run** menüsünden '**Run'** ya da '**Ctrl+F11**' kısayolları ile çalıştırabilirsiniz.



10. Programı çalıştırdığımızda konsolda "Merhaba" mesajını görürüz.



Not. Tüm bu işlemlerin adım adım gerçekleştirimini gösteren uygulama videosuna aşağıdaki adresten erişebilirsiniz;
https://youtu.be/UIg5wv4LFb4

BASİT BİR JAVA PROGRAMININ KALIBI

Her programlama dilinde olduğu gibi Java programlama dilinde de komutlar (sınıfları, nesneleri içeren) , sabitler, mesajlar, değişkenler, operatörler ve açıklama satırları vardır. Bu kavramları açıklamadan önce Java dilinin yapısını (ne nereye yazılıyor?) basit bir program (Merhaba.java) üzerinde açıklayalım.

```
public class Merhaba{

        public static void main(String[] args) {

                System.out.println("Merhabalar…");

        }

}
```

Buraya Java sınıfının (class) adını yazıyoruz.(Sınıf adı dosya adımızla aynı olmalıdır) Sol ayraç {, her sınıfın gövde kodları için başlangıç noktasıdır.

*Burası komutlar için başlangıç noktası (**main**). Sadece bir metodun ismi **main** olabilir. **Sol** ayraç {, metodun başlangıç noktasıdır.*

İşlem kısmı, (Ekranda görmek istediğimiz mesajların, çıktıların yazıldığı yer)

} Sağ ayraç metot gövdesinin sonu

} Sağ ayraç sınıf gövdesinin sonu

? Herhangi bir Java programı derlendikten sonra çalıştırılabiliyorsa, programın belli bir metodundan çalıştırılmaya başlanır. Bu metot nedir?

Cevap: public static void **main**(String[] args) { ... }

Açıklama;

public deyimi ile yöntemin herkese açık olduğunu, **static** deyimi ile sınıf tarafından paylaşıldığı, **void** deyimi ile geriye değer döndürmediğini, **main** yöntemine sahip olduğunu, **String [] args** ile yöntemde kullanacağımız parametreleri belirtiyoruz. Parametre belirtirken sınıfını ve parametre adını veriyoruz. **String** deyimi sınıf adı, **args** ise parametre adını belirtir

Java Dili Gramer Yapısı

Java programlama dili C,C++ dilinin gramer yapısını kullanır. Her komut satırı **';'** karakteri ile sonlandırılmalıdır. Komutlar küçük harflerle yazılır (**class** deyimini **Class** şeklinde yazdığınızda program hata verir). Açıklama (Yorum) satırları içinde;

Eğer tek satırlık bir açıklama yapılacaksa **'//'** karakteri,

```
// Bu karakterlerden sonra yazılanlar ekranda görünmez
```

Birden fazla satırlık açıklama yapılacaksa **' /* ……. */ '** karakterleri kullanılır.

```
/* Bu karakterlerden sonra yazılanlar ekranda görünmez ve
   Derleyici tarafından dikkate alınmazlar */
```