BMB203. NESNEYE YÖNELİK PROGRAMLAMA

Ders 8: Java ile Nesneye Yönelik Programlama

Erdinç Uzun
NKÜ Çorlu Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Konular

- Sınıf / Nesne
- Encapsulation (Saklama, Paketleme)
- Constructor
- Static
- Polymorphism
  - Function Overloading
  - Constructor Overloading
- Kalıtım (Inheritance)
  - Abstract
  - Interface

Java

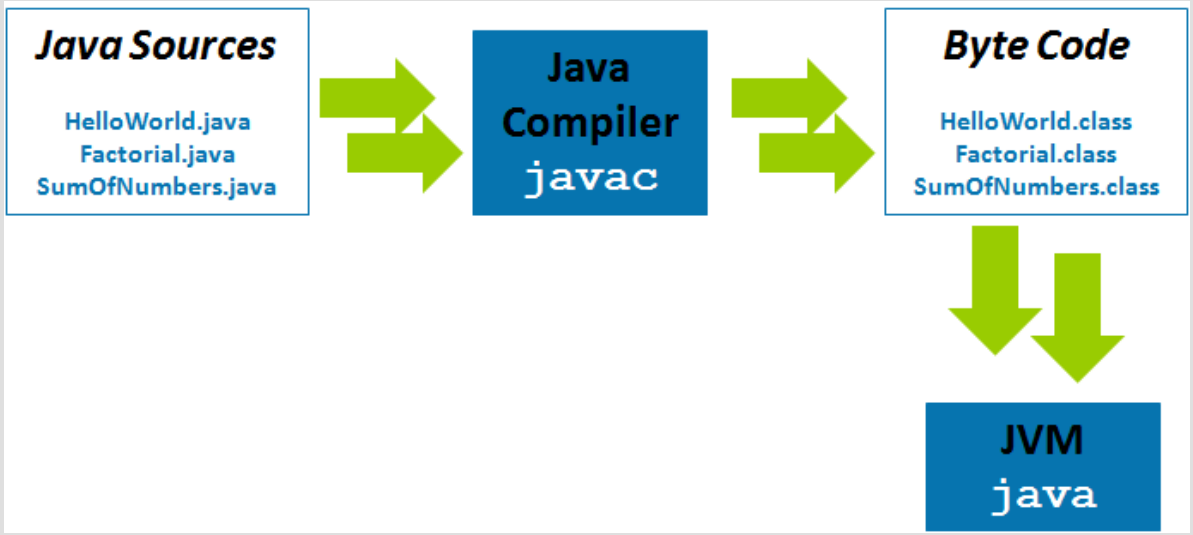
- Java, Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu, nesneye yönelik, zeminden bağımsız, yüksek verimli, çok işlevli, yüksek seviye, adım adım işletilen (yorumlanan-interpreted) bir dildir.
- Java, Sun Microsystems'den James Gosling tarafından geliştirilen bir programlama dilidir ve 1995 yılında Sun Microsystems'in çekirdek bileşeni olarak piyasaya sürülmüştür.
- Sun Microsystem'in şu anda Oracle Corporation ile bağlı ortaklığı bulunmaktadır.
- Bu dil C ve C++'dan birçok sözdizim türetmesine rağmen bu türevler daha basit nesne modeli ve daha az düşük seviye olanaklar içerir. Java uygulamaları bilgisayar mimarisine bağlı olmadan herhangi bir Java Virtual Machine (JVM)'de çalışabilen tipik bytecode'dur.

Java

- Java ilk çıktığında daha çok küçük cihazlarda kullanılmak için tasarlanmış ortak bir düzlem dili olarak düşünülmüştü. Ancak düzlem bağımsızlığı özelliği ve tekbiçim 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'nin 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 düzlemin gücü gözlendi ve tasarımında büyük değişiklikler ve eklemeler yapıldı. Bu büyük değişikliklerden dolayı geliştirilen yeni düzleme Java Platform 2 adı verildi ama sürüm numarası 2 yapılmadı, 1.2 olarak devam etti. 2004 sonbaharında çıkan Java 5, geçmiş 1.2, 1.3 ve 1.4 sürümlerinin ardından en çok gelişme ve değişikliği barındıran sürüm oldu. Java SE 7 ise (kod adı Dolphin) Sun'un üzerinde çalıştığı, Java teknolojisinin günümüz sürümüdür. 13 Kasım 2006'da Java düzlemi GPL ruhsatıyla açık kodlu hale gelmiştir.

Bir Java Kodu Nasıl Çalışır?

- Yazılımcı ve bilgisayar mühendisleri Java kodunu yazar.
- Bu kod bir Java derleyicisi ile derlenir. Sonuçta "bytecode" adı verilen bir tür sanal makine kodu ortaya çıkar. Düzlem bağımsızlığını sağlayan bytecode'dur. Çünkü bir kere bytecode oluştuktan sonra yazılım sanal makine içeren tüm işletim sistemlerinde çalışabilmektedir.
- Bu bytecode Java Sanal Makinesi (İng., Java Virtual Machine) tarafından işletilir. Bu aşama, her bir bytecode komutunun teker teker yorumlanması ile icra edilebileceği gibi, anında derleme kullanılarak da gerçekleştirilebilir.



Anında Derleme

- Java ilk çıktığında bytecode işletme hızı çok iyi değildi. Yerine göre sistemin öz yazılımlarından 5-10 kat yavaş çalışıyordu. Bu nedenle bazı yazılım geliştirmm şirketleri JIT yani "Just-in-time compile", "anında derleme" araçları üretmeye başladılar.
- Yapılan şey bytecode'u sanal makinenin kurulu olduğu gerçek sistemin diline anında derleme yaparak dönüştürmesiydi. Bu sayede verimde ciddi artışlar sağlandı. Ama 2000 yılından sonra geliştirilen sanal makinelerde (HotSpot gibi) JIT'in işlevi VM içinde yer almaya başlamış, işlemci hızı ve bellek miktarının dramatik biçimde artması ile dış JIT yazılımları popülerliğini kaybetmiştir.
- Bugün halen birkaç ürün (Excelsior JET gibi) pazarda bulunsa da genellikle bu yöndeki ihtiyaç azalmıştır.

Java API

- Java API, Java yazılımlarında kullanılan yazılım kütüphanelerine genel olarak verilen isimdir. Java API ile disk, grafik, ağ, veri tabanı, güvenlik gibi yüzlerce konuda kullanıcılara erişim imkânı sunulur. Java API J2SDK'nın bir parçasıdır.
- API (Application Programming Interface - Uygulama Programlama Arayüzü), herhangi bir uygulamanın belli işlevlerini diğer uygulamalarında kullanabilmesi için oluşturulmuş bir modüldür.

Gabrage Collector (Atık Veri Toplayıcı)

- Atık veri toplama teknolojisi Java'dan önce de var olan ama Java ile adını duyurmuş ve yaygın olarak kullanılmaya başlanmış bir kavramdır.
- C++, C gibi dillerin en büyük engellerinden birisi dinamik bellek yönetimidir. Yazılımda gösterici (işaretçi; İng., pointer) kullanarak dinamik olarak bellek ayırdıktan sonra o bellek ile işiniz bittiğinde mutlaka ayrılan belleği bellek yöneticiye özel altyordamlar yardımıyla (delete, free vs.) iade etmeniz gerekir. Yoksa bellek sızıntısı (İng., memory Leak) oluşur ve bu bir süre sonra yazılımın ve işletim sisteminin beklenenden farklı davranmasına yol açabilir.
- Sızıntıların saptanması oldukça güçtür ve bulunması zor hatalara yol açar. Bu nedenle bugünün tüm büyük C ve C++ yazılımları az da olsa bellek sızıntısı içerir (işletim sistemleri dahil).

Gabrage Collector (Atık Veri Toplayıcı)

- Atık veri toplayıcı sayesinde Java'da bir nesne oluşturulduktan sonra o nesne ile işiniz bittiğinde hiçbir şey yapmanız gerekmez: Sanal makine akıllı bir biçimde kullanılmayan bellek bölümlerini belirli aralıklarla ya da uyarlamalı yöntemlerle otomatik olarak temizler ve sisteme iade eder. Bu işleme çöp toplama (İng., garbage collection) adı verilir. Çöp toplama sistemlerinin yapısı oldukça karmaşıktır ve geçen yıllar içinde büyük gelişmeler kaydedilmiştir. Çöp toplayıcının varlığı Java'da bellek sızıntısı olmayacağı anlamına gelmez, ama bellek sızıntıları daha ender olarak ve farklı şekillerde karşınıza çıkar ve genellikle tedavi edilmesi daha kolaydır.

JAR

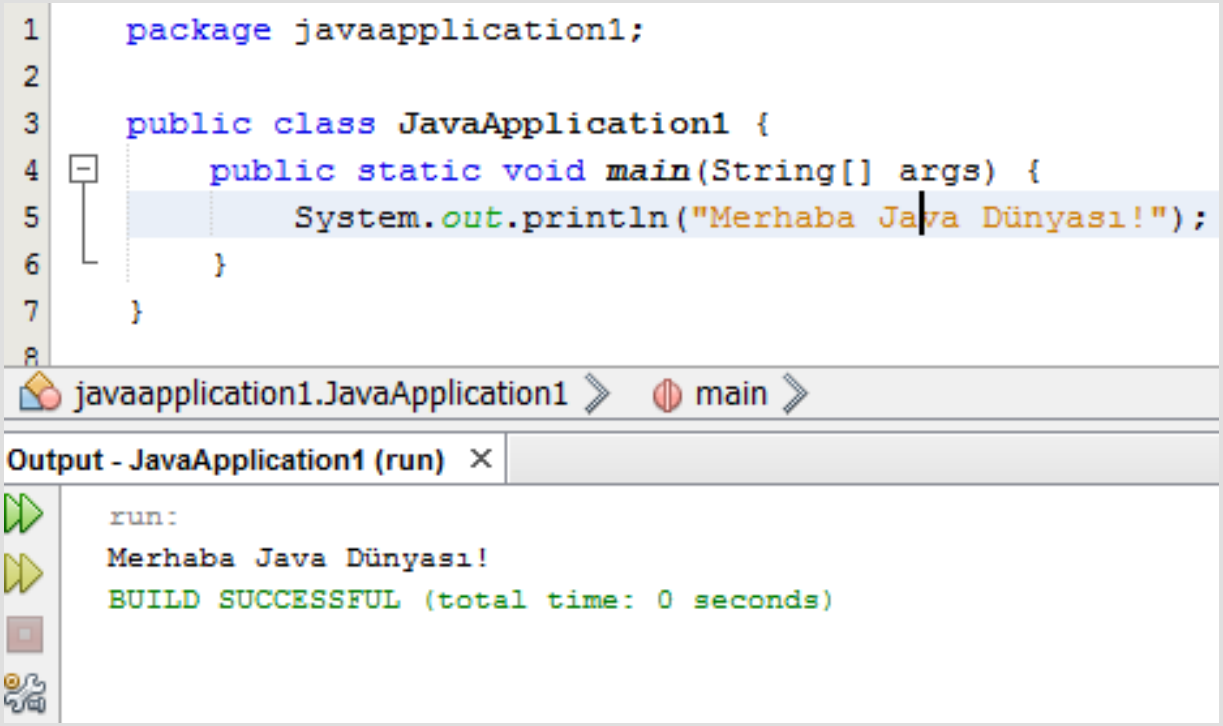
- Jar (İng. Java Archive), aslında bir tür sıkıştırma formatıdır. Jar ile derlenen Java kodları ile oluşan yazılımın paketlenip taşınması kolay bir hale getirilir. Jar dosyaları temelde bytecode blokları içerir. Jar dosyaları genellikle kütüphane oluşturmada ya da uygun biçimde hazırlanırsa işletim sisteminden doğrudan çalıştırılabiliecek bir şekilde kullanılabilir (Executable jar, işletilebilir jar) jar dosyalarının içeriğini sıkışıtırma yazılımları ya da java yazılım geliştirme araçları ile inceleyebilirsiniz.
- Java 1.5 ile yeni bir tür jar oluşturma metodu da kullanıma girdi. Pack200 adı verilen hiper-compression algoritması ile jar dosyaları daha küçük boyutlara indirilebiliyor. Ancak bu teknoloji daha çok ağ üzerinden yapılan transferlerde kullanılıyor.

İlk Java Kodu

```
package javaapplication1;

public class JavaApplication1 {
    public static void main(String[] args) {

    }

}
```

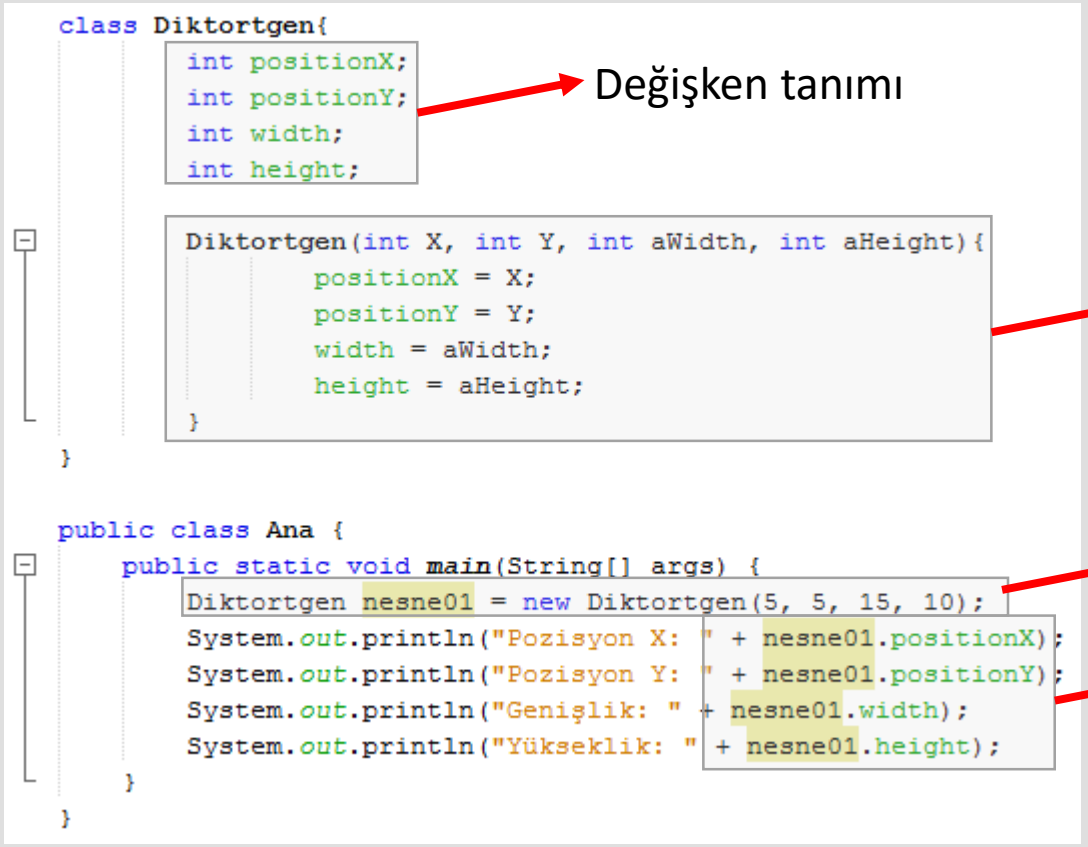
- Package = NameSpace
- Diğer kısımlar C# ile aynı…

Merhaba Java Dünyası



- System.out.println = System.Console.WriteLine
- C#'a çok benzer
- Döngüler, if kullanımı, değişkenler gibi konularda C# benzerlik gösterir. Bu günkü dersin amacı sadece C# göre farklılık gösteren kısımları incelemektir.

Sınıf - Nesne



- Constructor kullanmada fark yoktur.
- Ancak, hiçbir erişim belirteci kullanılmadığında değişken, metotlar ve constructor'lar private değil (C# böyleydi), public olmuştur.

Sınıf - Nesne

- Ayrıca bir sınıf için aşırı yükleme kurallarına uyulduğu sürece birden fazla yapılandırıcı bildirilebilir. Üstelik, **this** anahtar kelimesini şu şekilde **this(eger varsa parametreler)** kullanarak bir yapılandırıcı içinden baska bir yapılandırıcı çağrılabilir. Pratikte derleyici sınıfın geçerli örneğine işaret eden yöntemlere, **this** diye bir gizli değişken geçirir. Bu **this** anahtar kelimesi geçerli nesnenin referansını açıkça kullanmanız gerektiğinde (örneğin yapılandırıcıları yapılandırıcılar içinde çağırırken ve geçerli nesneye yöntemden bir referans döndürürken) kullanılabilir.
- Eğer bir sınıf için hiç yapılandırıcı bildirmezseniz, derleyici otomatik olarak hiç parametresi olmayan bir yapılandırıcı yaratır. Bu yapılandırıcı varsayılan yapılandırıcı olarak bilinir.
- Eğer parametreli bir yapılandırıcı bildirirseniz derleyici varsayılan yapılandırıcıyı oluşturmayacaktır, bu durumda varsayılan yapılandırıcıyı çağırarak örnek oluşturamazsınız.

Finalize

- C# ve C++'tan farklı olarak Java'da yok edici tanımlama (destructor) yapılamamaktadır. Buna karşın bir nesnenin yok edilmesi durumunda bir şeylerin yapılması gerekiyorsa finalize() metodu kullanılabilmektedir. Bu metot çalışma esnasında nesne yok edileceği zaman çağırılacaktır. Bu metot bir sona erdirme (finalization) metodudur.
- Bir nesne referansı kullanılarak kapsamı içinde ve artık kullanılmadığı ve çöp toplayıcı tarafından alındığı her yerde erişilebilirdir. Bir nesne ona ilgisi olan hiçbir değişken kalmadığında çöp toplayıcıya alınır. Çöp toplayıcı otomatik olduğundan nesneyi silmeye gerek yoktur. Fakat bazı durumlarda bazı nesne için ayrılmış bazı kaynakları bırakmak için özel bir silme yöntemine ihtiyaç duyabilirsiniz.

Finalize

```
class Diktortgen{
        int positionX;
        int positionY;
        int width;
        int height;

        Diktortgen(int X, int Y, int aWidth, int aHeight){
                positionX = X;
                positionY = Y;
                width = aWidth;
                height = aHeight;
        }

        public void finalize() {
        System.out.println("Diktortgen finalize()");
        }
}
```

- Bu durumda özel finalize() yöntemini kullanmalı ve silme kodunu bu yöntemin içine koymalısınız. Fakat, bu yöntemi kullanırken dikkatli olmalısınız, çünkü finalize() sadece bir nesne çöpe toplandığında yada çöpe hemen toplanamadığında kullanılabilir. Bu yüzden eğer gerçekten özel bir silme yöntemi bildirmek istiyorsanız bu yöntemin nesneyle işiniz bittiğinde kendiniz çağırmalısınız.