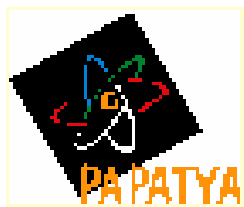
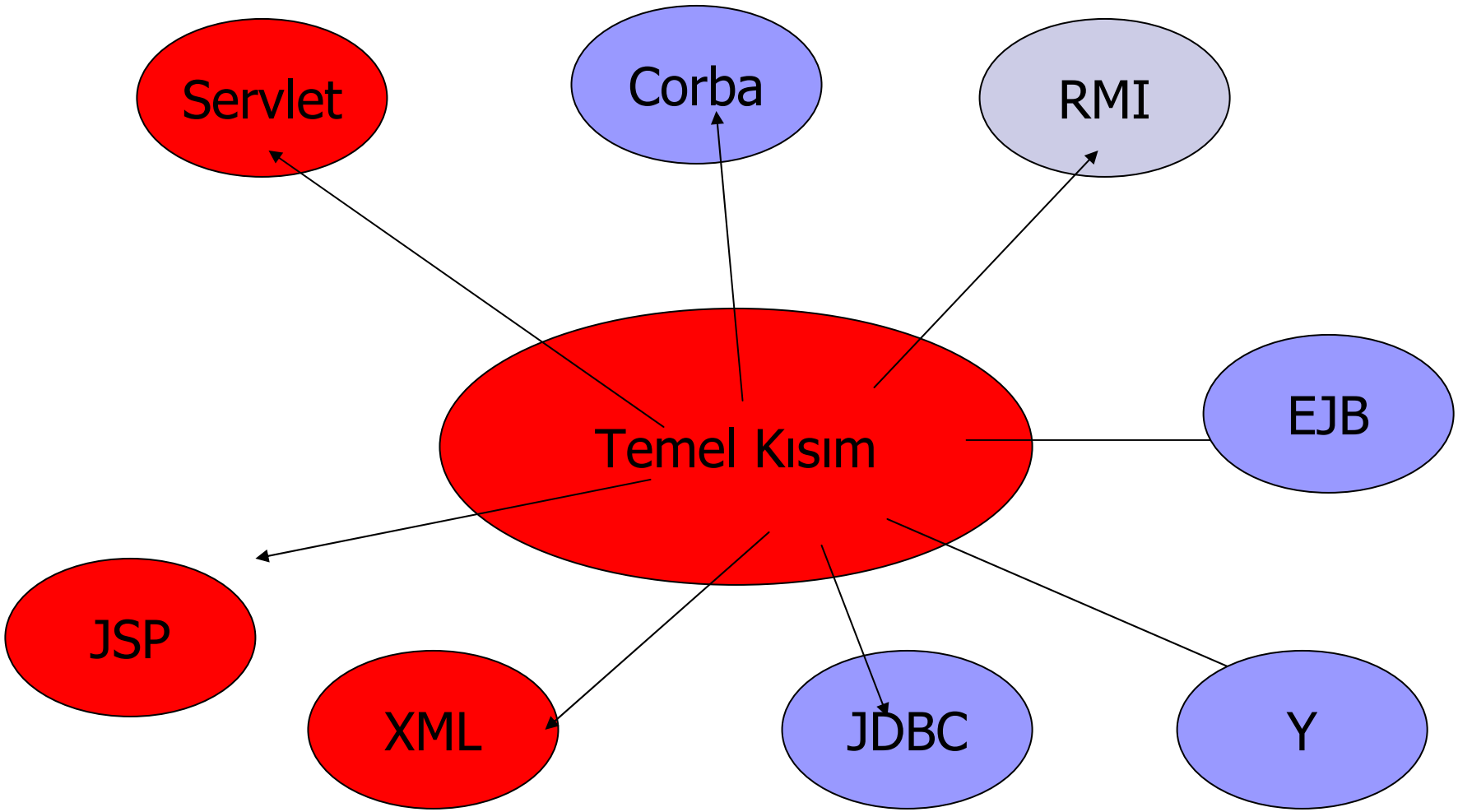
Java
ile
Nesneye Yönelik Programlama



Bu Dönem Hakkında



Java Nedir?

- Java ™ platformu , ağ(network) ‘ın önemi hesaba katılarak ve aynı yazılımın birçok değişik bilgisayar ortamında veya değişik tür makinalarda çalışması fikri ile geliştirilmiş yeni bir teknolojidir.

- Java teknolojisi kullanılarak aynı uygulamayı değişik ortamlarda çalıştırabiliriz – örneğin Pc’lerde , Macintosh bilgisayarlarda, hatta cep telefonlarında.

- Java diğer programlama dilleri gibi başlı başına bir ürün değildir**.**

- Java ve Java’ya bağlı alt teknolojiler, Sun Microsystems tarafından verilmiş belirtimlerden (specifications) oluşmaktadır.Eğer bu belirtimlere sadık kalınmaz ise hukuki olarak suç işlenmiş olur.

Java İle Neler Yapılabilir?

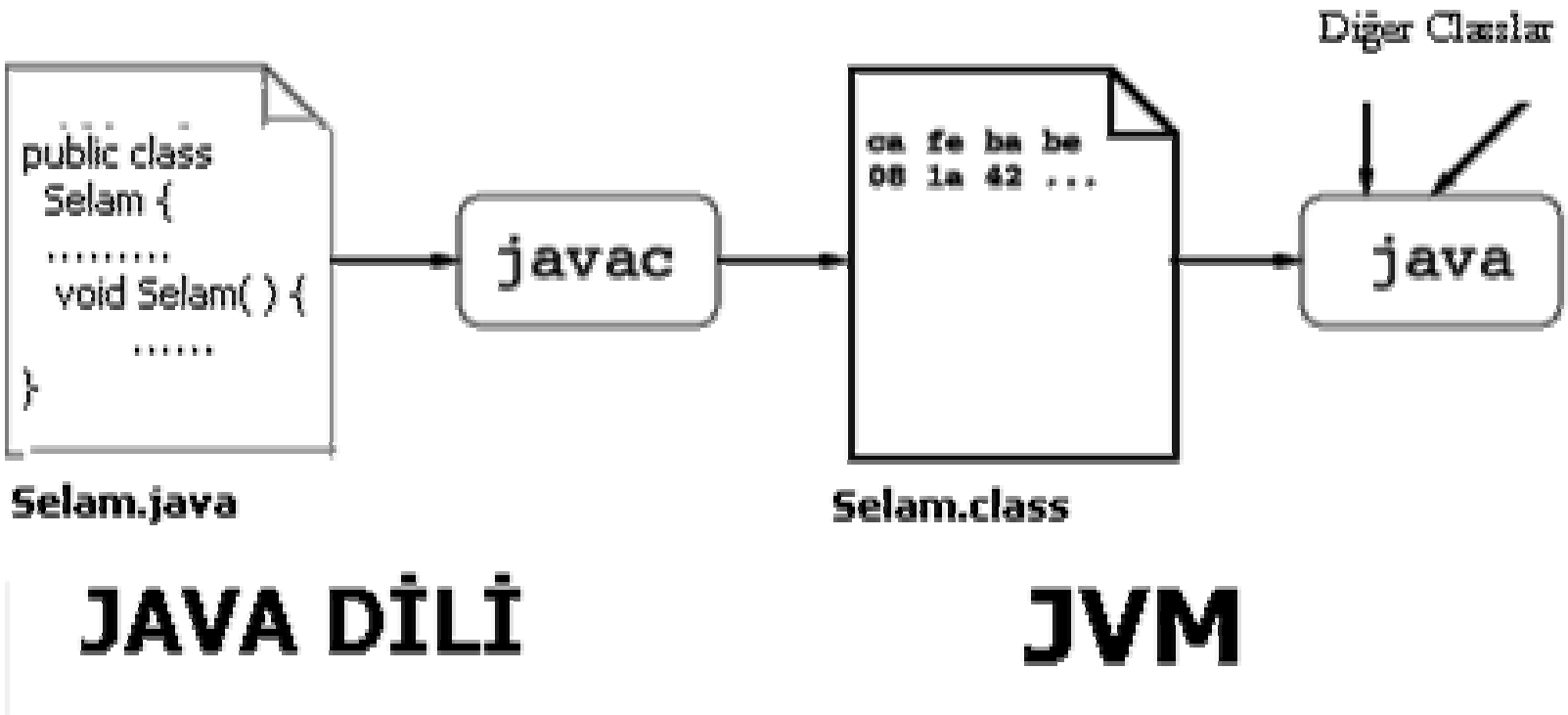
Java Programlama dili ile projelerimizi diğer programlama dillerine göre daha kolay ve sağlıklı bir şekilde yapmamız mümkündür . Kısaca göz atacak olursak , Java ile ;

- GUI (graphical user interface , grafiksel kullanıcı ara yüzü) uygulamaları , Appletler.
- Distributed components (ör . EJB, RMI, CORBA).
- Servlet, Jsp (web tabanlı uygulamalar).
- Veri tabanlarına erişim ile alakalı uygulamalar.
- Cep telefonları, Smart kartlar için uygulamalar .
- Ve daha niceleri… için uygulamalar yazmamız mümkündür.

Bir Kere Yaz Her Yerde Çalıştır

- Java uygulamaları JVM (Java Virtual Machine) tarafından yorumlanır(interpreted).
- JVM , işletim sisteminin en tepesinde bulunur
- Java uygulamaları değişik işletim sistemlerinde , herhangi bir değişiklik yapmadan çalışabilir, Java'nın felsefesi olan "bir kere yaz heryerde çalıştır" sözü gerçekleştirilmiştir.

Çalışma Evreleri



Çalışma Evreleri

Derleme anı (Compile time)

Çalıştırma anı (Run time)

Kategoriler

- Java platformunun ana grupları.
  - Standart Java
  - Enterprise Java
  - Gömülü cihazlar için Java (embedded devices)
  - XML Teknolojileri
  - Diğer Teknolojiler

Standart Java

- Java 2 SDK (J2SE)
- Java 2 Runtime Environment
- Java Plug-in
- Java Web Start
- Java HotSpot Server Virtual Machine
- Collections Framework
- Java Foundation Classes (JFC)
- Swing Components
- Pluggable Look & Feel
- Accessibility
- Drag and Drop
- Security
- Java IDL
- JDBC
- JavaBeans
- Remote Method Invocation (RMI)
- Java 2D

Enterprise Java

- J2EE (Java 2 Enterprise Edition)
- CORBA Teknolojisi
- ECperf Teknolojisi
- Enterprise JavaBeans Teknolojisi
- Kontaynerler için Java Yetkilendirme Kontratı (Java Authorization Contract for Containers) (Java ACC)
- Java IDL
- JavaMail API
- Java Mesajlaşma Servisi (Message Service) (JMS) API
- JavaServer Faces
- JavaServer Pages
- Java Servlets
- JDBC Teknolojisi
- J2EE Bağlayıcı Mimarisi (Connector Architecture)
- Hareketler (Transactions)

Gömülü Cihazlar İçin Java *(Embedded Devices)*

- Java 2 Platform, Micro Edition (J2ME technology)
- Java 2 Platform, Micro Edition (J2ME Teknolojisi)
- Bağlı Aygıt Konfigurasyonu (Connected Device Configuration) (CDC)
- Sınırlı Bağlanmış Aygıt Konfigurasyonu (Connected Limited Device Configuration) (CLDC)
- C Sanal Makinası (Virtual Machine) (CVM)
- K Sanal Makinası (Virtual Machine) (KVM)
- PersonalJava
- Java Card
- JavaPhone API
- Java TV API
- Jini Network Technology
- Mobil Bilgi Aygıt Profili (Mobile Information Device Profile) (MIDP)

XML Teknolojileri

- XML İlişkilendirilmesi için Java Mimarisi (Java Architecture for XML Binding) (**JAXB**)
- XML-Tabanlı RPC için JAVA API'si (Java API for XML-Based RPC) (**JAX-RPC**)
- XML Mesajlaşması için JAVA API'si (Java API for XML Messaging) (**JAXM**)
- XML İşlemleri için JAVA API'si (Java API for XML Processing) (**JAXP**)
- XML Kayıtları için JAVA API'si (Java API for XML Registries) (**JAXR**)

Diğer Teknolojiler

- Araç Ürünler
  - MIF Doclet
  - Sun ONE Stüdyo (Studio)
- Ağ (NetWork) Ürünleri
  - Sertifikalı JAIN API Ürünleri (JAIN API Certified Products)
  - Java Dynamic Management Kit
  - Java Yönetim Uzantısı (Java Management Extensions) (JMX)
  - Java MetaData Arabirimi (Java Metadata Interface) (JMI)
  - Java Paylaşılan Veri Araç Takımı Java Shared Data Toolkit
  - Java Spaces Teknolojisi
  - Servis Sağlıyıcılar için Java Teknolojisi (Java Technology for Service Providers)
  - Jini Ağ Teknolojisi (Network Technology)
  - JXTA Projesi
  - J2ME Platformu için JXTA Projesi (Project JXTA for J2ME Platform)
  - Sun Chili!Soft ASP

Java'nın Gelişim Evreleri

1995	**Java teknolojisinin ilk çıkış yılı ; ilk olarak Applet teknolojisinin dikkat çektiği seneler.**
1996	**Java Development Kit (JDK) v1.0 çıkartıldı. Temel seviyeli işlevleri içeren bir versiyon (ör. soket programlama, Girdi/Çıktı (Input/Output), GUI (Graphical User Interface- Grafik Kullanıcı Arabirimi)**
1997	**JDK 1.1 çıkartıldı. Bu sürümde Java GUI , veritabanı erişimi için JDBC , dağınık nesneler için RMI ve daha birçok yeni gelişmeler eklendi .**
1998	**JDK 1.2 çıkartıldı .** **JFC/Swing yayınlandı- aynı sene içersinde http://java.sun.com internet adresinden 500,000+ adet indirme(download) gerçekleştirildi.**
1999	**Java teknolojisi J2SE, J2EE ve J2ME olarak 3'e bölündü .** **Java HotSpot (performans arttırıcı) yayınlandı .** **JavaServer Pages (JSP) teknolojisi yayınlandı.** **J2EE platform'u yayınlandı .** **Linux üzerinde J2SE platformu yayınlandı .**
2000	**JDK v1.3 çıkartıldı .** **Java APIs for XML teknolojisi yayınlandı .**
2002	**JDK v1.4 versiyonu çıkarıldı (Merlin projesi).** **Java API for XML binding yayınlandı.**
2003	**2003 yılının sonuna doğru JDK v1.5 versiyonun çıkarılması planlanmaktadır (Tiger projesi).**

Java'nın Başarılı Olmasındaki Sebepler

- **Nitelikli bir programlama dili olması**
  - C++ da olduğu gibi bellek problemlerinin olmaması .
  - Nesneye yönelik (Object - Oriented) olması
  - C/C++/VB dillerinin aksine dinamik olması .
  - Güvenli olması .
  - Internet uygulamaları için elverişli **(Applet, JSP, Servlet, EJB, Corba, RMI).**
- **Platform bağımsız olması : bir kere yaz her yerde çalıştır**

Çöp Toplayıcı *(Garbage Collector)*

- Bir programın çalışma durumunda ortaya çıkan ve sonradan kullanılmayan (gereksiz) nesneleri bulur ve onları yok eder (*destroy*).
- Bellek yönetiminin (*memory management*) yükü, kodu yazan kişiden Java'ya geçmiş olur
- Diğer dillerde, örneğin C++ da , oluşturulan nesnelerin yok edilme sorumluluğu kodu yazan kişiye aittir.
- Çöp toplayıcısı(*garbage collector*) JVM'in yazılışına (*implementation*) göre değişkenlikler gösterebilir.

Java'da Yorum Satırı

- Java kaynak kodunun içerisine istediğiniz yorumları yazabilmeniz için belli yol izleminiz gerekmektedir.
- Java'da yorum satırlarını belirtme iki şekilde mümkün olur
  1. /* **yorum** */ , slash - yıldızdan , diğer yıldız-slash arasına kadar istediğiniz yorumu yazabilirsiniz . Uzun satırlı yorumlarda bu yöntemi kullanabilirsiniz.
  2. // **yorum** , tek satırlık yorum yapmak için idealdir. Kısa yorumlarınız için bu yöntemi kullanabilirsiniz.

Herşey Nesne - 1

- Java'da herşeye nesne olarak davranırız. Herseyin nesne olmasına rağmen nesneleri yönetmek için " referanslar" kullanılır .

Örnek : Diyelim ki elimizde bir maket uçak (nesne olarak düşünün) ve bu maket uçağa ait bir de kumanda (referans) olduğunu düşünelim.

Bu maket uçağı havada sağa sola döndürmek için elimizdeki kumanda cihazını kullanmak zorundayızdır; benzer şekilde havalandırmak veya yere indirmek için de kumanda cihazından faydalanırız. Burada dikkat edilmesi gereken unsur kumanda cihazından çıkan emirlerin maket uçağı tarafından yerine getirilmesidir.

Herşey Nesne - 2

- Elimizde uzaktan kumandanın (referans) olması, maket uçağımızın (nesne) olduğu anlamına gelmez .
- Uzaktan kumandamız (referans) da tek başına hayatı sürdürebilir.

```
String kumanda ; // kumanda referansı şu an için
                 //String bir nesneye bağlı değil.
```

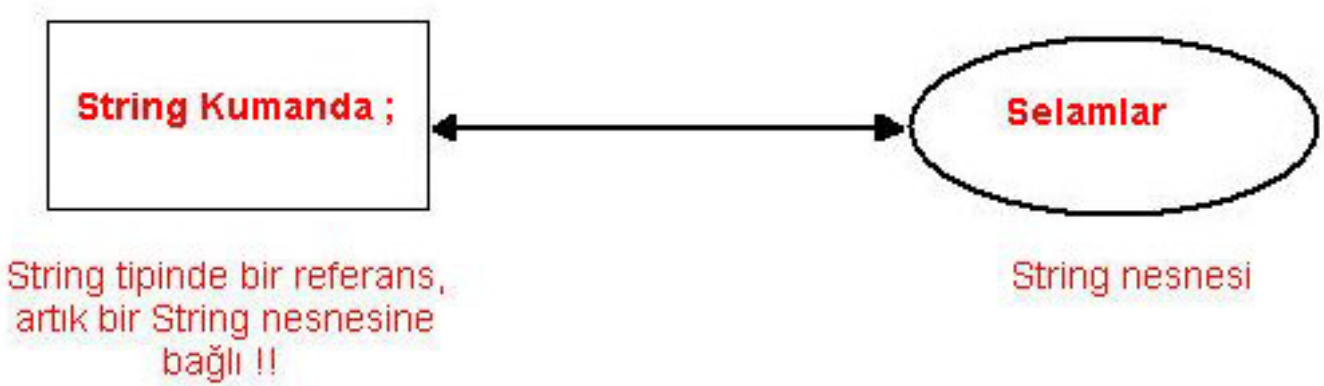
String tipinde bir referans, şu an için herhangi bir String nesnesine bağlı değil

Herşey Nesne - 3

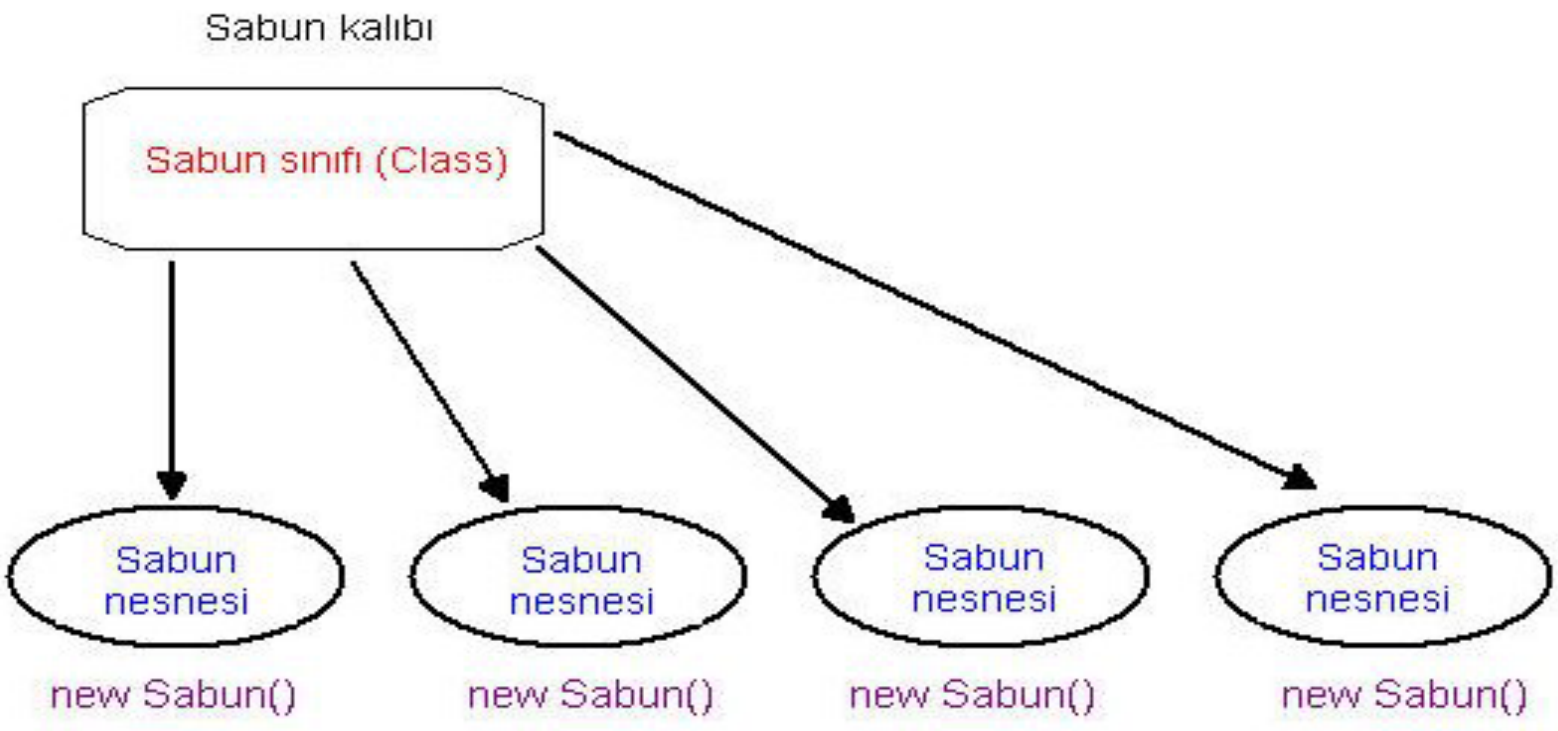
- Bir referansa mesaj göndemek istiyorsak onu bir nesneye bağlamamız gerekir.

```
•String kumanda= new String("Selamlar") ;
```

```
•String kumanda="Selamlar" ;
```



Sınıf Nedir? Nesne Nedir?




```
Sabun   s = new Sabun();
```

Java'da Depolanan Veriler Nerede Durur - 1

- **Stack** : Bulunduğu nokta RAM'dır... Stack üzerinde referansların kendileri bulunur.
- **Heap** : Burası genel amaçlı bir havuzdur . Nesnelerin kendisi bu alanda durur.
- **Statik Alan** : Bu alan RAM'de bulunur. Statik alanda yer alan veriler , programın çalışması süresince orada yaşarlar. Nesnelerin kendileri bu alanda yer almazlar.

Java'da Depolanan Veriler Nerede Durur - 2

- **Non-RAM Bellek :** Bazı durumlarda uygulamaların içerisinde oluşturduğumuz nesnelerin,uygulama sonlandıktan sonra bile varlıklarını sürdürmelerini isteriz.
  1. **Akışkan Nesneler (streamed objects) :** Bu nesneler, genellikle ağ(network) üzerindeki başka bir makineye gönderilmek üzere bayt ırmaklarına dönüştürülürler.
  2. **Kalıcı Nesneler (persistent objects) :** Bu nesneler kendi durumlarını(*state*) saklarlar ve diskimizde saklanırlar. Kendi durumlarını saklamaktan kasıt ise özelliklerinin (*attribute*) değerlerinin korunmasıdır.

Temel *(Primitive)* Tipler

- Temel tipler stack alanında saklanırlar.

Temel tip	Boyut	Minimum	Maximum	Sarmalıyıcı Sınıf Tipi
`boolean`	–	–	–	`Boolean`
`char`	16- bit	`Unicode 0`	`Unicode` $2^{16}-1$	`Character`
`byte`	8- bit	`-128`	`+127`	`Byte`
`short`	16- bit	-2^{15}	$+2^{15}-1$	`Short`
`int`	32- bit	-2^{31}	$+2^{31}-1$	`Integer`
`long`	64- bit	-2^{63}	$+2^{63}-1$	`Long`
`float`	32- bit	`IEEE754`	`IEEE754`	`Float`
`double`	64- bit	`IEEE754`	`IEEE754`	`Double`
`void`	–	–	–	`Void`

Sarmalayıcı *(Wrapper)* Sınıflar

- Temel tiplerin birer adet sarmalıyıcı (*wrapper*) sınıfları bulunur.

```
char c = 'x' ; // temel tip
```

```
Character C = new Character(c); // sarmalayıcı sınıf
```

Geçerlilik Alanı *(Scope)* - 1

```
{
  int a = 177;
  /* sadece a mevcut*/
  {
    int b = 196;
    /*  a ve b mevcut */
  }
  /* sadece a mevcut */
  /* b "geçerlilik alanının dışına çıktı " */
}
```

Geçerlilik Alanı *(Scope)* - 2

- C ve C++ doğru ama Java'da yanlış olan bir ifade

```
{ // dış alan

  int a = 12;

  { // iç alan

    int a = 96; /* java'da yanlış ama C ve C++ doğru */

  } // iç alanın sonu

} //dış alanın sonu
```

Nesneler İçin Geçerlilik Alanı *(Scope of Objects)*

```
if (true){

  String s = new String("Selamlar");

}  /* geçerlilik alanının sonu*/
```

• Geçerlilik alanının sonunda **String** nesnesi "Çöp Toplayıcısı" (*Garbage Collector*) tarafından bellekten silinecektir.

Yeni Sınıf Oluşturma

```
public class YeniBirSinif  {

   .....



}
```

Alanlar - 1

- Alanlar, temel bir tip veya sınıf tipinde olabilir.

```
public class YeniBirSinif  {
   public int i;
   public double d;
   public boolean b;
}
```

Alanlar - 2

Temel (*primitive*) Tip	Mevcut değer (*Default value*)
`boolean`	`false`
`char`	`'\u0000' (null)`
`byte`	`(byte)0`
`short`	`(short)0`
`int`	`0`
`long`	`0L`
`float`	`0.0f`
`double`	`0.0d`

Alanlar - 3

```
public class YeniBirSinif  {
     public int i = 5 ;
     public double d = 3.23;
     public boolean b = true ;
}
```

Alanlar - 4

```
YeniBirSinif ybs = new YeniBirSinif();
```

Alanlara Ulaşım

- **Nesnenin alanlarına ulaşmak için “.” (nokta) kullanılır**.
- Bu alanların erişim belirleyicileri
  - public
  - private
  - protected
  - friendly

  olabilir.

```
YeniBirSinif   ybs = new YeniBirSinif();
ybs.i ;
ybs.d ;
ybs.b ;
```

Alanlara Değer Atama

```
YeniBirSinif ybs = new YeniBirSinif();
ybs.i = 5;
ybs.d = 5.3;
ybs.b = false;
```

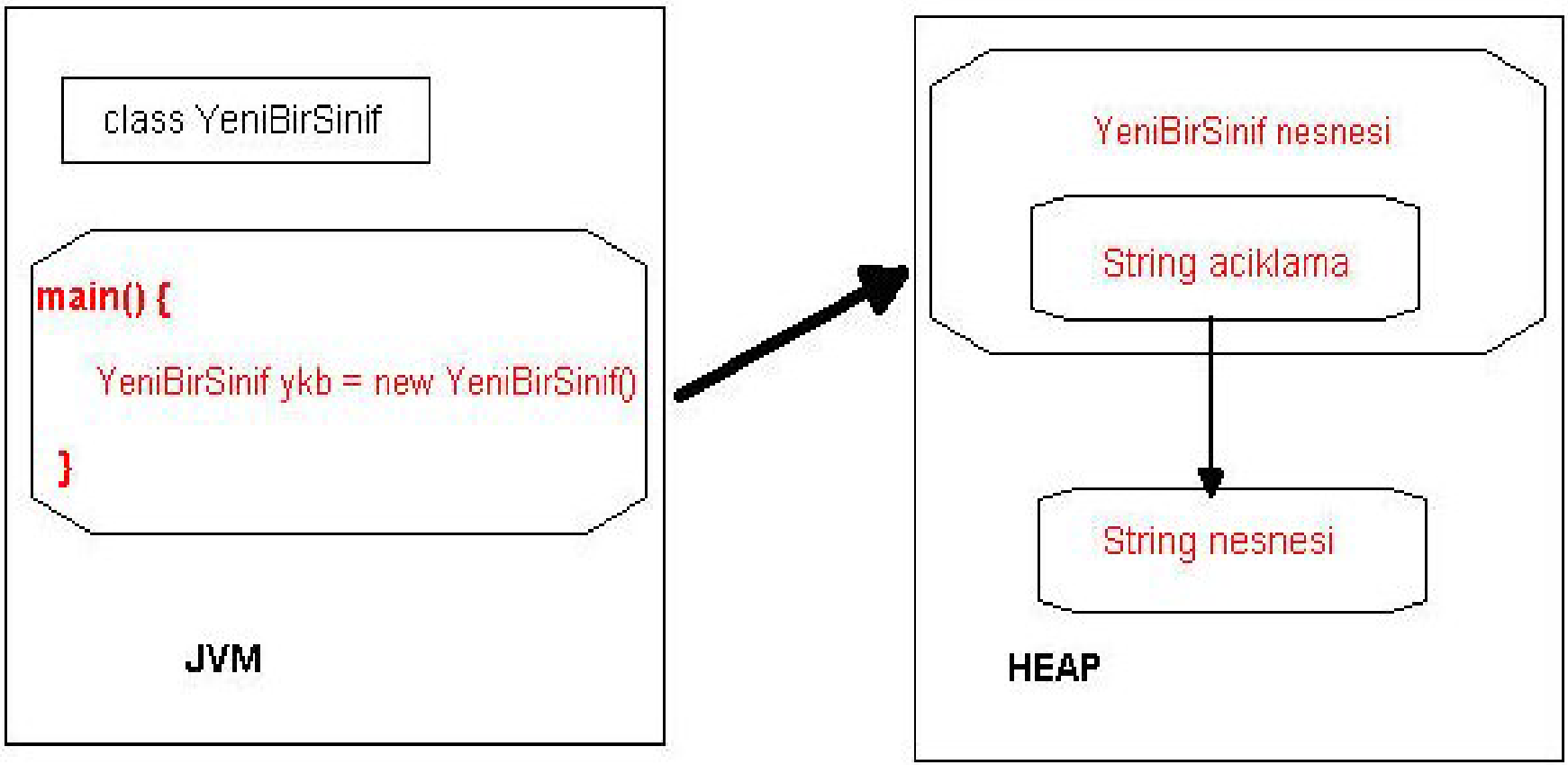
Sınıf Tipindeki Alanlar - 1

```
public class YeniBirSinif  {

  public int i;
  public double d;
  public boolean b;
  public String aciklama = new  String("aciklama");

}
```

Sınıf Tipindeki Alanlar - 2



Yordamlar *(Methods)* - 1

```
dönüşTipi  YordamIsmi( /* parametre listesi */ ) {
  /* Yordamın gövdesi */
}
```

Yordamlar *(Methods)* - 2

- **`dönüşTipi`**= Yordamların iki şansı vardır:
  - Değer döndürürler
    - Temel (*primitive*) bir tipde değer (int, double, short vb..)
    - Sınıf tipinde bir değer (String, Double, Short vb...)
  - Değer döndürmezler = **`void`**

Yordamlar *(Methods)* - 3

- **`yordamIsmi`** = Java'nın kendisine ait olan sözcükler (**if , else , import , class , return.**.vb) ve Türkçe karakterler haricinde istenilen isim kullanılabilir. Ancak, yordamlar bir eylem içerdikleri için, yordam isimlerinin de bir eylemi belirtmesi tercih edilir.

- Örneğin:
  - **`sayiSirala()`**
  - **`enBuyukSayiBul()`**
  - **`sqlCalistir()`**

Yordamlar *(Methods)* - 4

- **`parametre listesi`**= Yordam içerisinde işlemler yapabilmek için gerekli olan parametreler. Bu parametreler temel tipte veya sınıf tipinde olabilirler.

Yordamlar *(Methods)* - 5

- **`Yordam gövdesi`** = Bu kısım kodu yazan kişinin yaratıcılığına bağlı olarak değişir.

Yordam *(Method)* Örneği - 1

```
int boyutDondur(String kelime) {
   return kelime.length() ;
} // yordamın sonu
```

Yordam *(Method)* Örneği - 2

```
String elmaHesapla(int elmasayisi) {
   return  new String("elma sayisi = "
                                + elmasayisi*2);
}  // yordamın sonu
```

Yordam İçindeki Yerel Değişkenlerin İlk Değerlerini Alması

```
void hesapla(String kelime , int kdv ) {
  int sondeger = 0;
  int kelimeboyut = 0 ;
  int toplamboyut ; // Hatalı !!
  toplamboyut++ ;   // Hatalı !!
  kelimeboyut = kelime.length();
  sondeger = kelimeboyut + kdv ;
}
```

İlk Java Programı - 1

```
public class Selam {
   public static void main(String args[]) {
       System.out.println("Selamlar !");
    }
}
```

İlk Java Programı - 2

- **`public class Selam :`** Bu kısım da yeni bir sınıf oluşturuyor...

İlk Java Programı - 3

- **`public static void main(String args[])`**
  - Java'da bir sınıfın tek başına çalışması isteniyorsa (*standalone*) bu yordam yazılmak zorundadır . Bu yordam sınıflar için bir başlagıç noktasıdır.
  - **`static`** yordamlar nesneye bağımlı olmayan yordamlardır. Bu yordamı kullanmak için, ilgili sınıfa ait bir nesne oluşturma zorunluluğu yoktur.

Diziler *(Arrays)*

- **`main()`** yordamı parametre olarak *String* sınıfı tipinde dizi alır, bu *String* sınıfı tipindeki dizinin içerisinde, konsoldan Java uygulamasına gönderilen parametreler bulunur .
  - `args[0]` **:** konsoldan girilen 1. parametre değerini taşır …
  - `args[1]` **:** konsoldan girilen 2. parametre değerini taşır …
  - `args[n-1]` **:** konsoldan girilen n. parametre değerini taşır …
- Java'da diziler sıfır'dan başlarlar. Diziler ilerleyen bölümlerde yoğun bir şekilde incelenecektir.

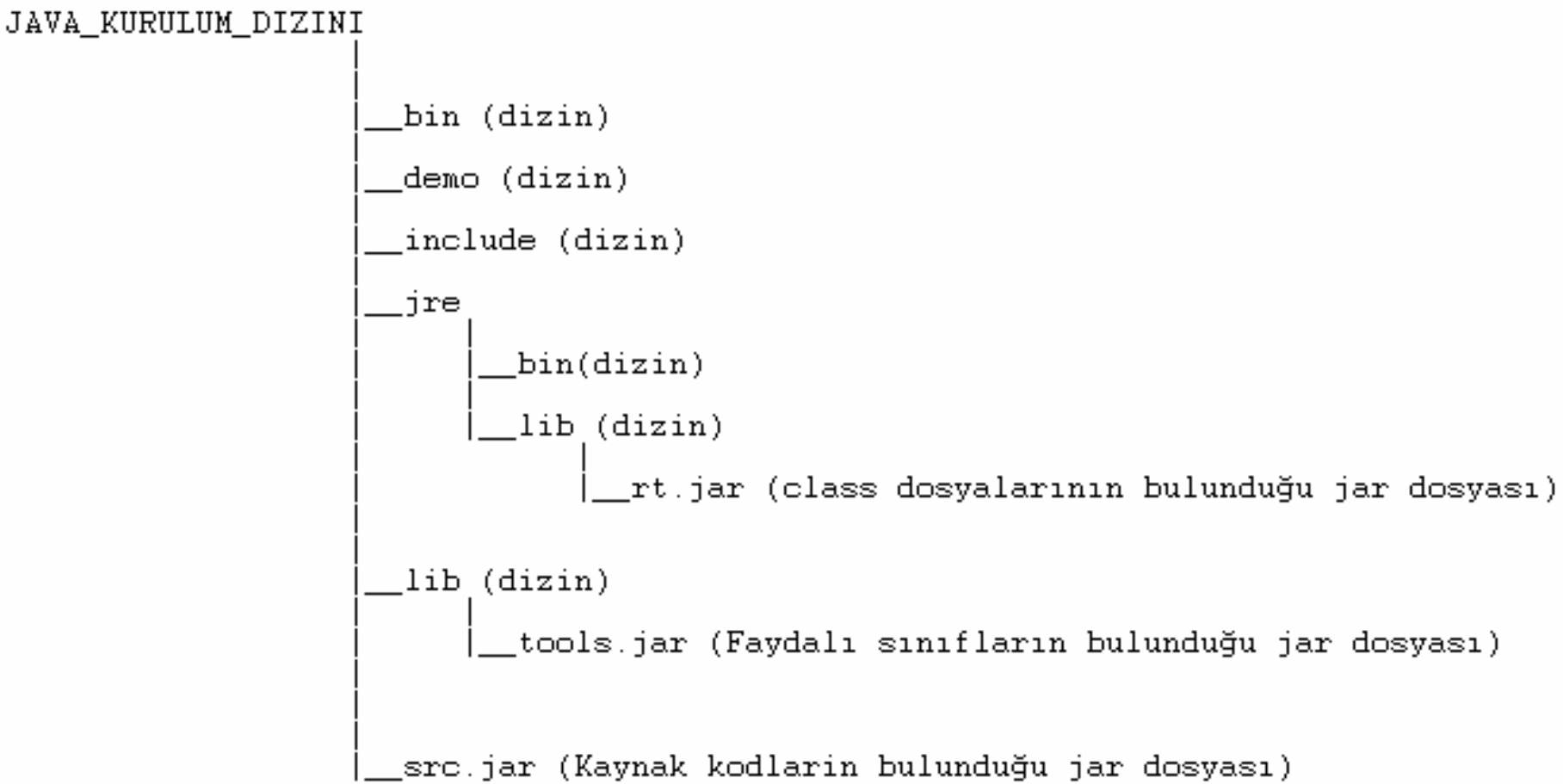
İlk Java Programı - 4

- **`System.out.println("Selamlar !")`**
  - Bu komut satırı, bilgileri konsola (ekrana) basmamızı sağlar. Java'nın dokümanlarına bakarsak;
  - ***System*** sınıfı altında static bir alan olan **out** alanının mevcut olduğunu görüyoruz. Bu yüzden ***System*** sınıfını oluşturmak zorunda değiliz (**`new System()`**).
  - **out** alanı bize **P*rintStream*** nesnesi oluşturur ve ***PrintStream*** nesnesinin **`println()`** methodu ile bilgileri konsola(ekrana) bastırırız.

Kurulum

- Kurulumlar dökümanlardan incelenebilir.

Dizin Yapısı



Nedir bu `args[]`? Ne İşe Yarar?

```
public class ParametreUygulamasi {
  public static void main(String[] args) {
    System.out.println("Girilen Parametre = "+args[0]);
  }
}
```

```
bash#   javac ParametreUygulamasi.java

bash# java  ParametreUygulamasi  test
```

Girilen Parametre = test

Hata Durumu

```
public class ParametreUygulamasi {
  public static void main(String[] args) {
    System.out.println("Girilen Parametre = "+args[0]);
  }
}
```

```
bash#   javac ParametreUygulamasi.java

bash# java  ParametreUygulamasi
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException at
ParametreUygulamasi2.main(ParametreUygulamasi.java:3)
```

Javadoc – Yorum İle Dökümantasyon Oluşturmak

- Dökümantasyon oluşturma yükünü ortadan kaldırır.
- Oluşturulan dökümanlar HTML formatında olur.

```
/** Ilk Java Kodumuzun Dokumantasyonu
Ekrana Selamlar diyen bir uygulama
* @author Altug B. Altintas (altuga@kodcu.com)
* @version 1.0
* @since 09.01.2002
*/
public class SelamDoc {
  /**sayiyi artirmak icin ,
    *degiskenler icin bir ornek
    */
    public int sayac = 0 ;
   /** siniflarda & uygulumalarda giris
     * noktasi olan yordam
     * @param args disaradan girilen
     * parameterler dizisi
     * @return donen deger yok
     * @exception Hic istisna firlatilmiyor
     */
   public static void main(String[] args) {
        System.out.println("Selamlar !");
   }
}
```

Sorular ...