Hafta-1 Java Giriş

Batuhan Düzgün - Mayıs 2020

Java Tarihçesi

Java Nedir?

Java platform bağımsız, **Nesneye Dayalı Programlama**'yı (Object Oriented Programming) tamamıyla destekleyen bir yazılım geliştirme dilidir.

Java aynı zamanda geliştirme platformunun kendisidir.

Java Programlama Dilinin Özellikleri

Java platform bağımsız bir dildir. (Windows, Linux, Mac OS)

Java Nesneye Dayalı programlamaya tamamıyla uygun bir dildir.

Java öğrenmesi basit bir dildir.

Java kararlı bir programlama dilidir.

Java, Multi-Thread (Çok Kanallı) programlamayı varsayılan olarak destekler.

Dağıtık sistemler ve Web programlama yapabilmenize imkan tanır.

Mobil ve Gömülü sistemlerde programlama imkanı sağlar.

Java Tarihçesi

1991 yılında Green Project ile Sun Microsystem’de başlar.

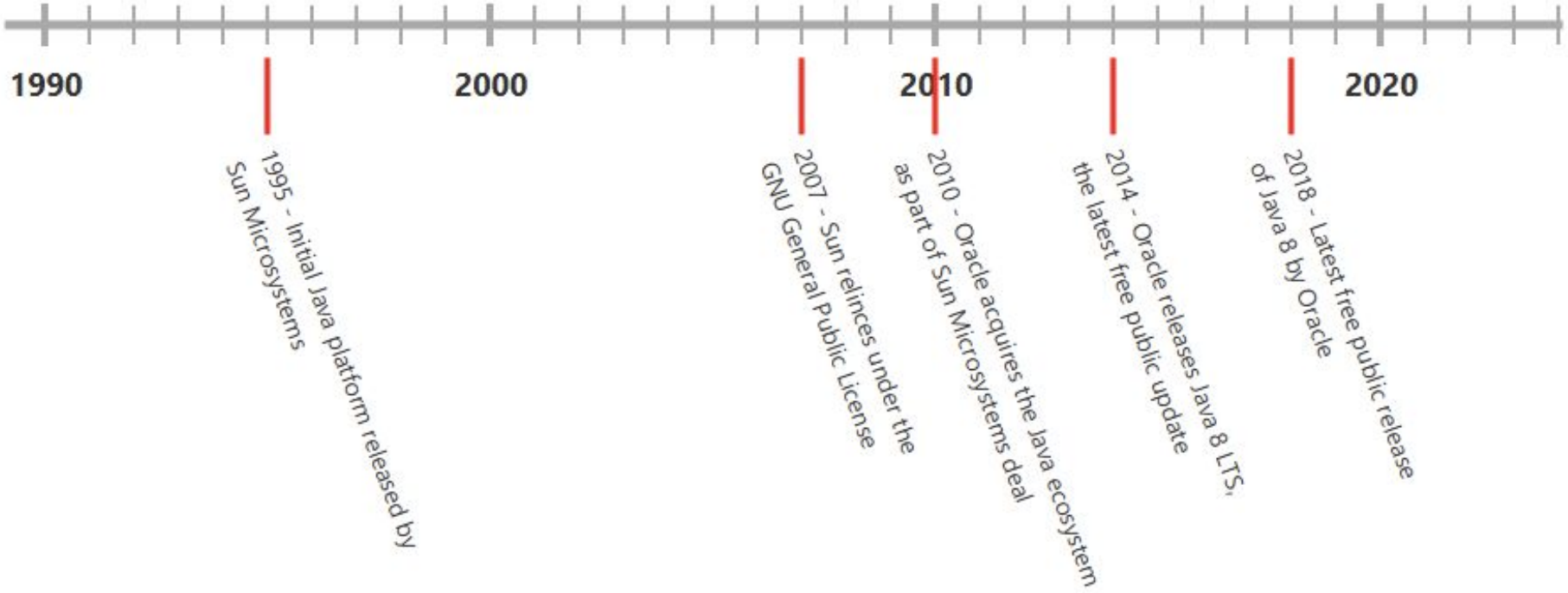
İlk ismi Oak (Yeşil Meşe Ağacı)

1995’te ilk Java Beta duyurulur.

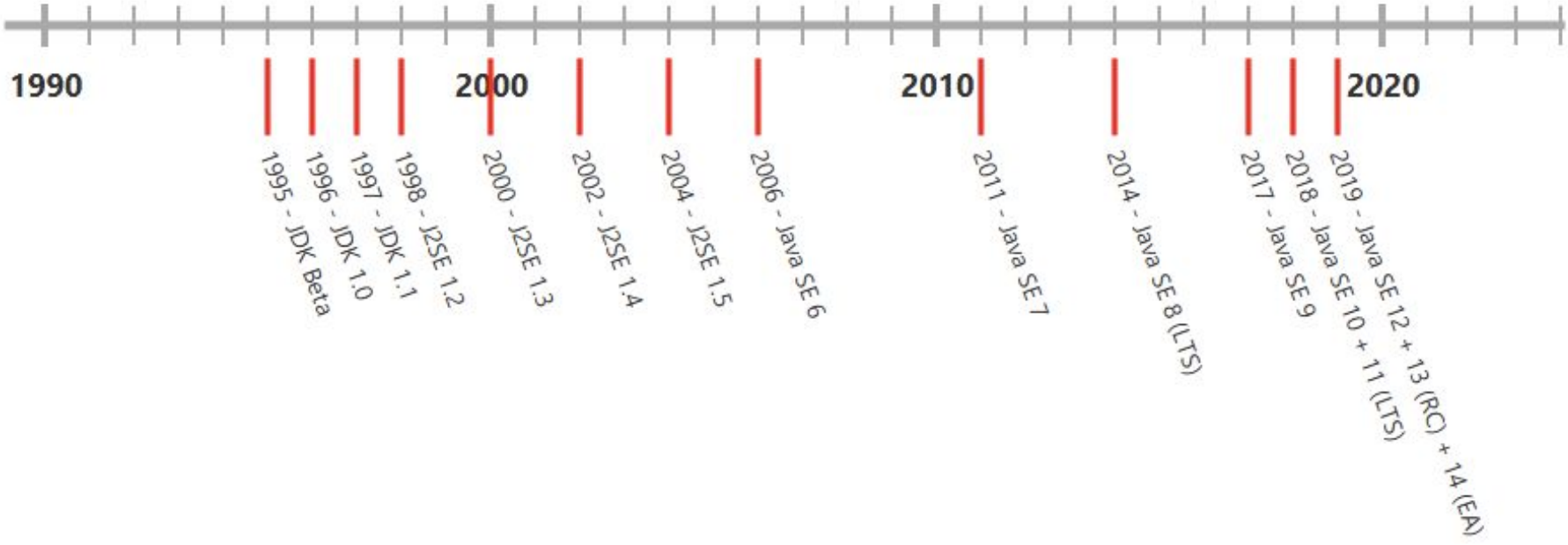
1996’da Java 1.0 adıyla resmen duyurulur.

2010 yılında Oracle firması tarafından satın alınır.

Java Tarihçesi



Java Sürüm Tarihçesi



Java ile Uygulama Geliştirme Yapabileceğiniz Alanlar

- Web uygulamaları geliştirebilirsiniz.
- Masaüstü uygulamaları geliştirebilirsiniz. (Eclipse IDE gibi)
- Kurumsal uygulamalar geliştirebilirsiniz. (www.sahibinden.com gibi)
- Mobil uygulamalar geliştirebilirsiniz.
- Gömülü sistem uygulamaları geliştirebilirsiniz.
- Robotik projelerde kullanabilirsiniz.
- Oyun programlamada kullanabilirsiniz. (Android ile oyun geliştirme gibi)

Java Uygulama Geliştirme Altyapıları

Java SE (Java Standard Edition)

Java EE (Java Enterprise Edition)

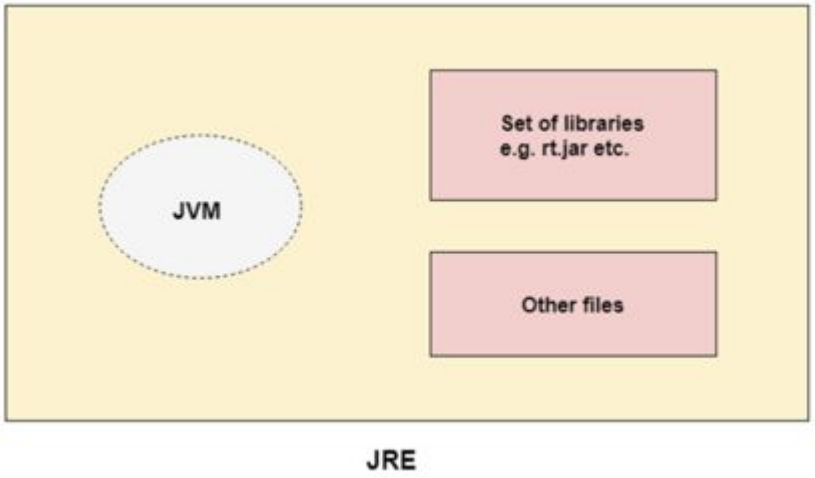
Java ME (Java Micro Edition)

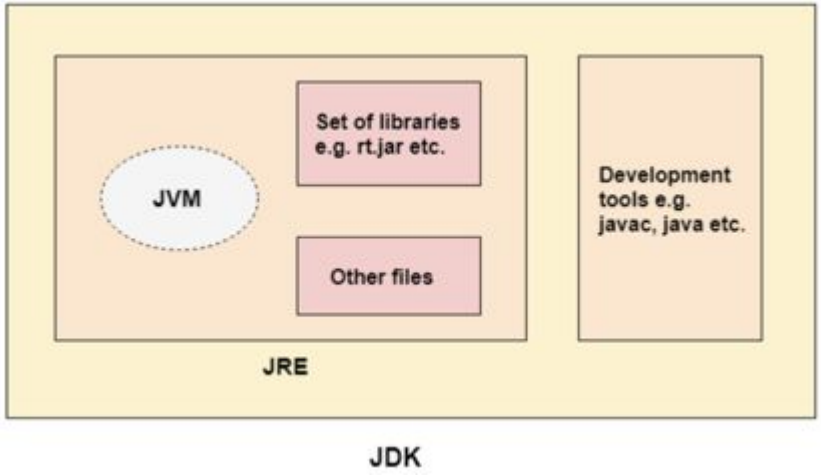
JavaFX

Java Temel Kavramlar

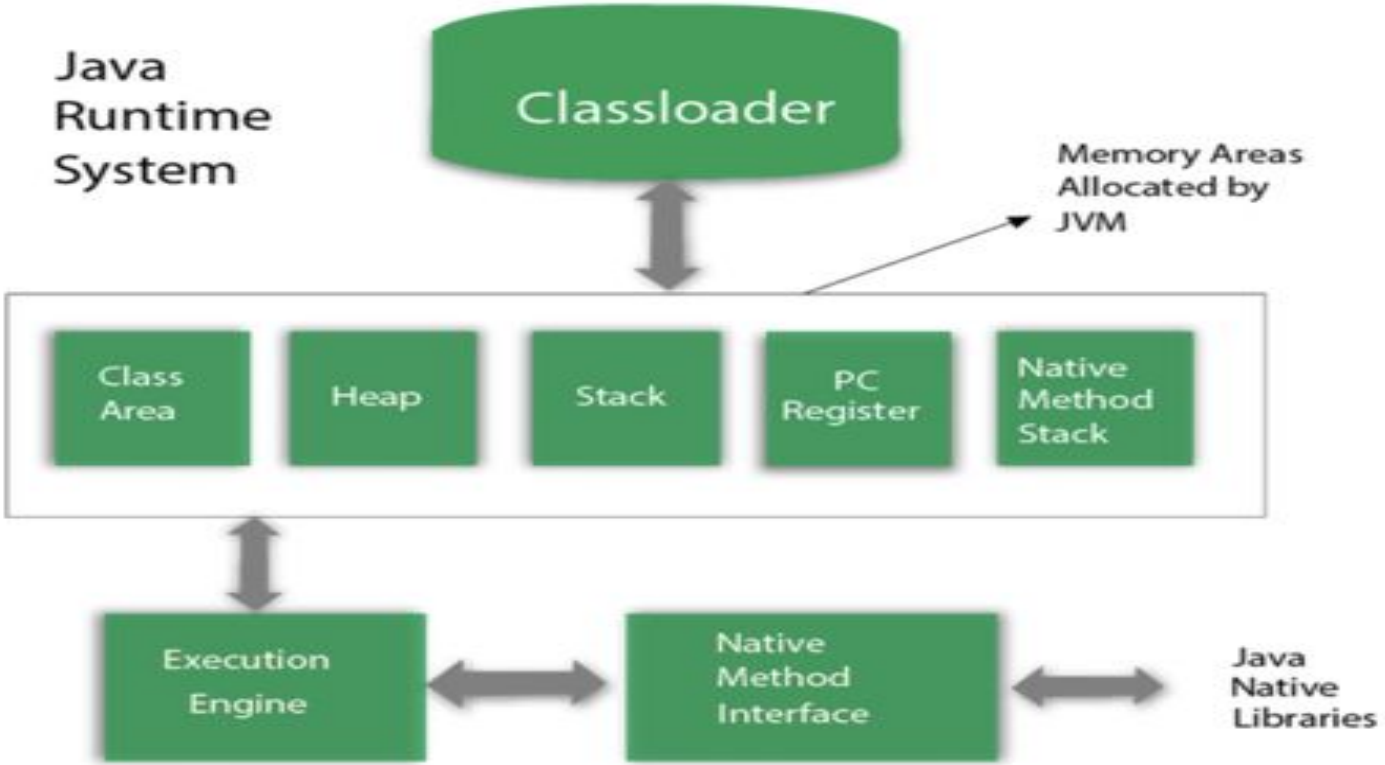
JDK, JRE, JVM Kavramları

JVM, Java Byte koda dönüştürülebilen her yazılım geliştirme dilini çalıştırabilme yeteneğine sahiptir.

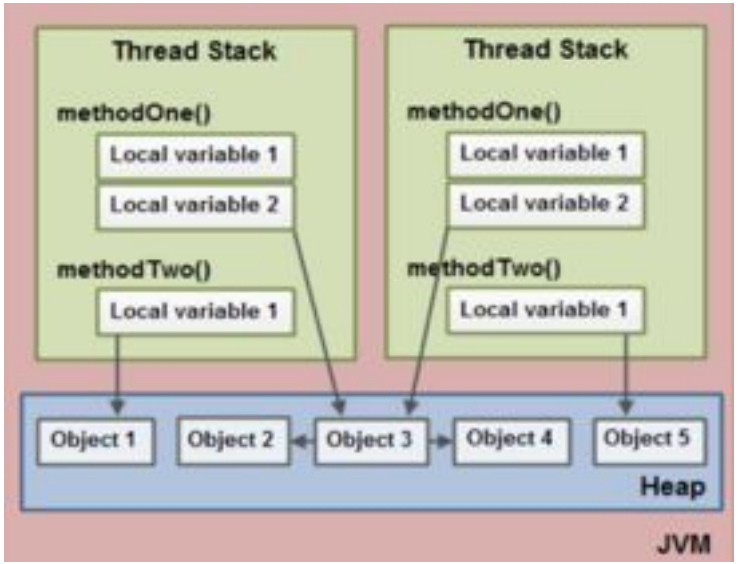


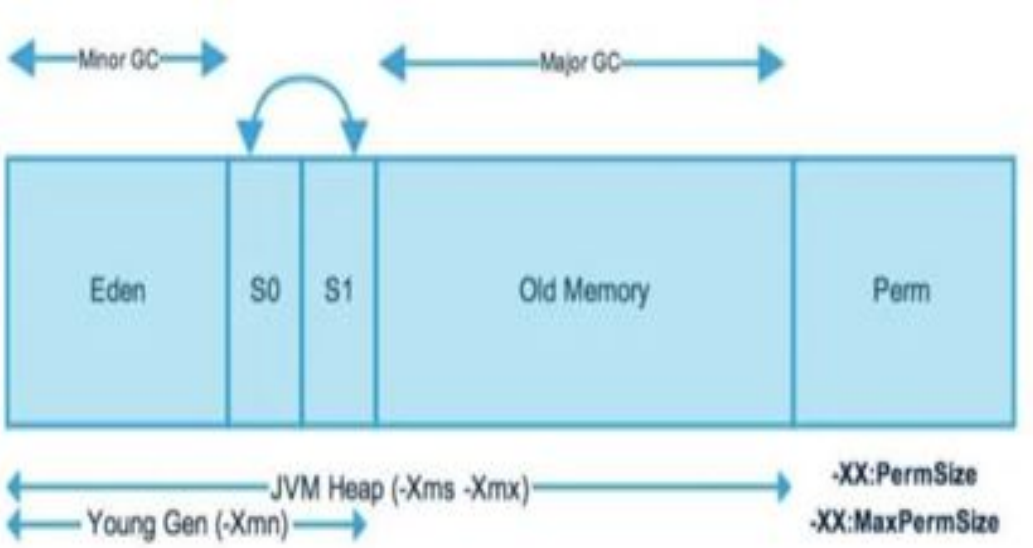


Java Virtual Machine Mimarisi



Java Hafıza Mimarisi





Java İlk Program

```
class MyFirstProgram {

    public static void main(String args[]) {
        System.out.println("Hello Java! | Merhaba Java!");
    }

}
```

Java Temel Kavramlar

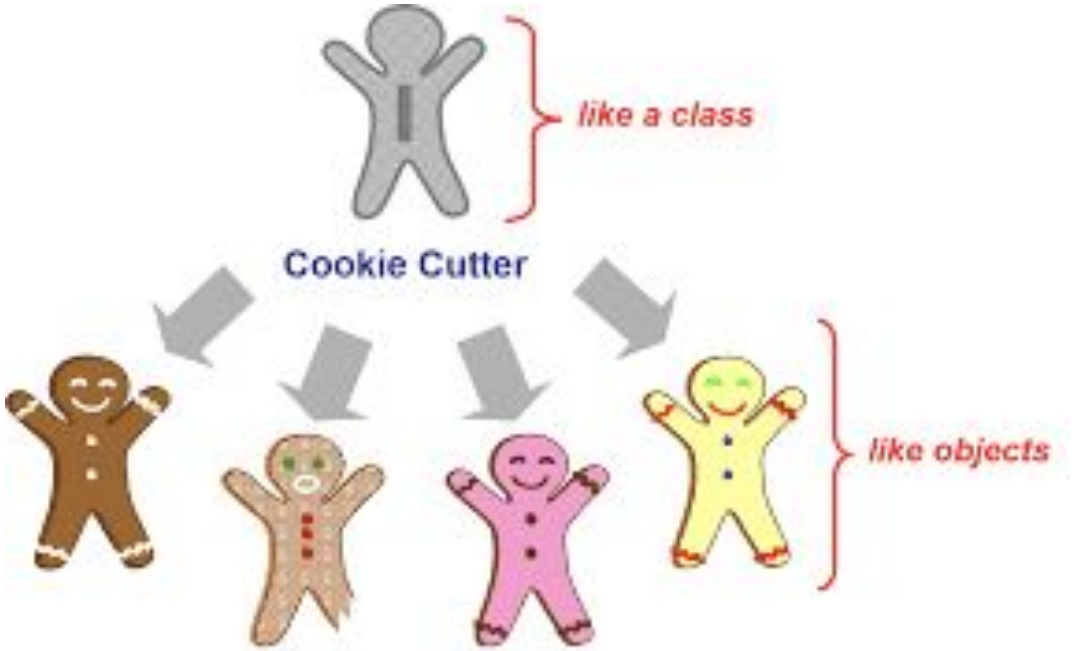
Sınıf: Gerçek hayatta yer alan bir nesnenin yazılım dünyasındaki model halidir.

Nesne: Tasarlanan bu modelden üretilenlere de nesne denilmektedir.

Metot (Fonksiyon): Fonksiyonlar eylemleri işaret eder. Örneğin: Printer isimli bir sınıfa ait “print” isimli eylem yazılım dünyasında bir metodu ifade eder. Metotlar belli parametreler alarak veya almadan belli bir üreten veya üretmeyen kod parçalarıdır.

Değişken: Değişkenler nesnenin niteliklerini işaret ederler. Örneğin: Printer isimli sınıfa ait “color” isimli nitelik bir değişkeni ifade eder. Eğer rengi beyazsa beyaz bir değer alır.

Java Temel Kavramlar



Java’da Değişkenler (Variables)

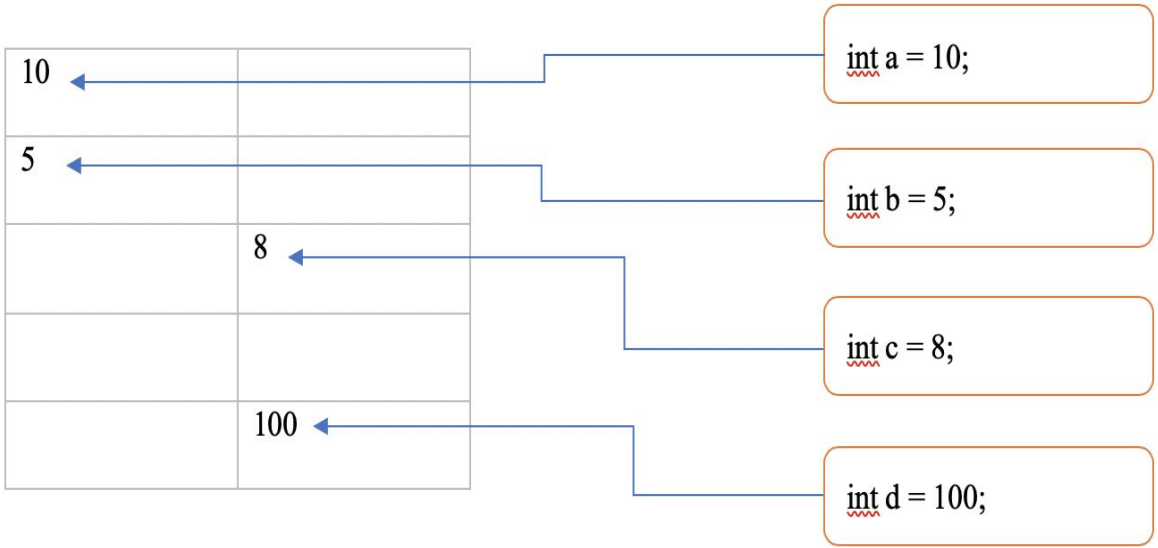
- Yerel Değişkenler
- Sınıf Değişkenleri (Statik olanlar)
- Nesne Değişkenleri (Statik olmayanlar)

Java Anahtar Sözcükler (Reserved Keywords)

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

Java Değişkenler

```
<veri tipi> <değişken ismi> = veri (değer)
```

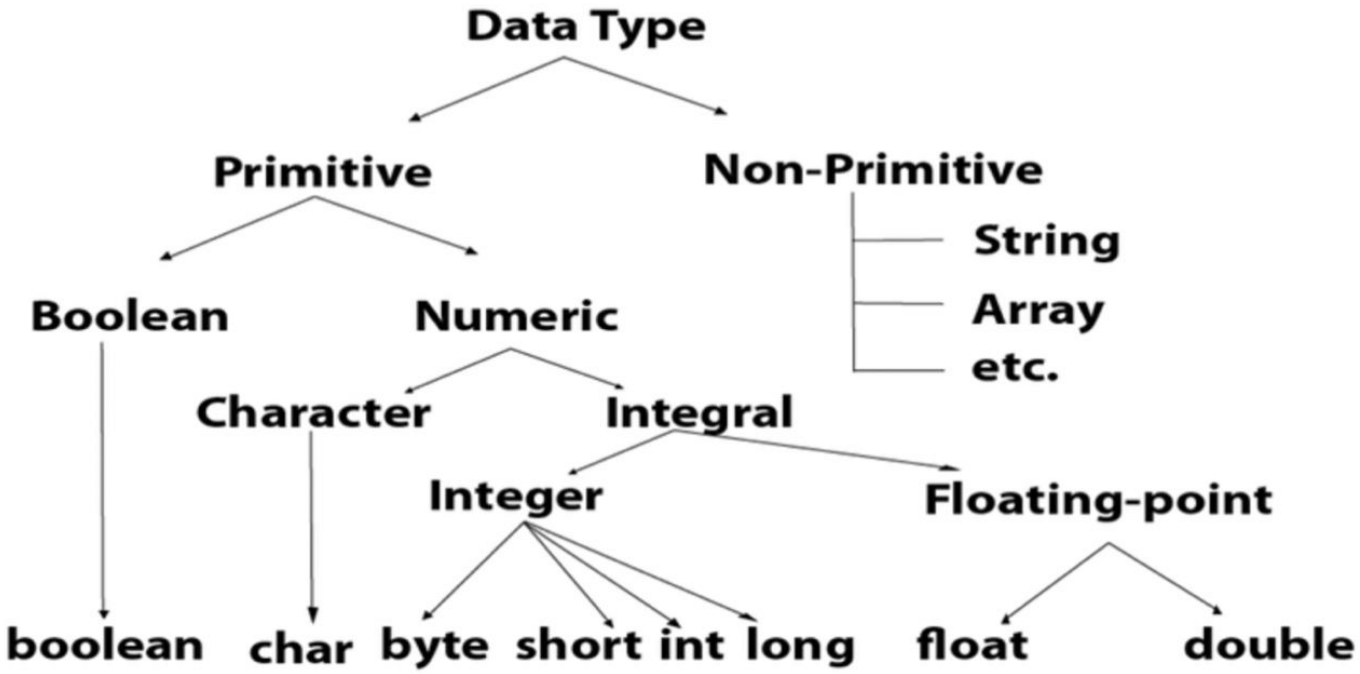


Java Değişkenler

```
<veri tipi> <değişken ismi> = veri (değer)
```

```
int a, b, c;          // 3 tane değişken virgüller ile ayrılarak tek satırda
tanımlanabilir.
int a = 10, b = 10;   // Birden fazla değişken aynı satırda ilk değerleri
atanarak tanımlanabilir.
byte b = 22;          // Tek değişkene ilk değer ataması yapılarak
double pi = 3.14159; // Tek değişkene ilk değer ataması yapılarak
char a = 'a';         // Tek değişkene ilk değer ataması yapılarak
```

Java Veri Tipleri



Java İşlem Operatörleri

Java Temel Operatörler

- Aritmetiksel Operatörler
- İlişkisel ve Eşitlik Operatörler
- Bitsel (Bit Düzeyinde) Operatörler
- Mantıksal Operatörler
- Atama Operatörleri

Java Temel Operatörler

```
public static void main(String args[]) {
   int a = 10;
   int b = 20;
   int c = 25;
   int d = 25;

   System.out.println("a + b = " + (a + b) );
   System.out.println("a - b = " + (a - b) );
   System.out.println("a * b = " + (a * b) );
   System.out.println("b / a = " + (b / a) );
   System.out.println("b % a = " + (b % a) );
   System.out.println("c % a = " + (c % a) );
   System.out.println("a++   = " +  (a++) );
   System.out.println("b--   = " +  (a--) );

   System.out.println("d++   = " +  (d++) );
   System.out.println("++d   = " +  (++d) );
}
```

Aritmetik Operatörleri

```
public static void main(String args[]) {
   int a = 10;
   int b = 20;
   int c = 25;
   int d = 25;

   System.out.println("a + b = " + (a + b) );
   System.out.println("a - b = " + (a - b) );
   System.out.println("a * b = " + (a * b) );
   System.out.println("b / a = " + (b / a) );
   System.out.println("b % a = " + (b % a) );
   System.out.println("c % a = " + (c % a) );
   System.out.println("a++   = " +  (a++) );
   System.out.println("b--   = " +  (a--) );

   System.out.println("d++   = " +  (d++) );
   System.out.println("++d   = " +  (++d) );
}
```

İlişkisel ve Eşitlik Operatörleri

```
public static void main(String args[]) {
    int a = 10;
    int b = 20;

    System.out.println("a == b = " + (a == b) );
    System.out.println("a != b = " + (a != b) );
    System.out.println("a > b = " + (a > b) );
    System.out.println("a < b = " + (a < b) );
    System.out.println("b >= a = " + (b >= a) );
    System.out.println("b <= a = " + (b <= a) );
}
```

Bitsel Operatörler

```
public static void main(String args[]) {
   int a = 60;        /* 60 = 0011 1100 */
   int b = 13;        /* 13 = 0000 1101 */
   int c = 0;

   c = a & b;         /* 12 = 0000 1100 */
   System.out.println("a & b = " + c );

   c = a | b;         /* 61 = 0011 1101 */
   System.out.println("a | b = " + c );

   c = a ^ b;         /* 49 = 0011 0001 */
```

Bitsel Operatörler

```
        System.out.println("a ^ b = " + c );

        c = ~a;           /*-61 = 1100 0011 */
        System.out.println("~a = " + c );

        c = a << 2;       /* 240 = 1111 0000 */
        System.out.println("a << 2 = " + c );

        c = a >> 2;       /* 15 = 1111 */
        System.out.println("a >> 2  = " + c );

        c = a >>> 2;      /* 15 = 0000 1111 */
        System.out.println("a >>> 2 = " + c );
    }
```

Mantıksal Operatörler

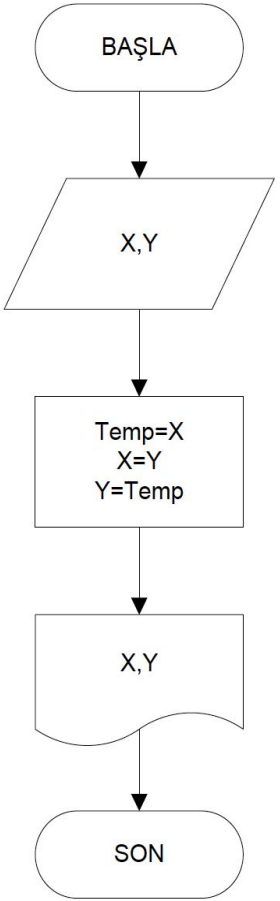
```
public class Test {

   public static void main(String args[]) {
      boolean a = true;
      boolean b = false;

      System.out.println("a && b = " + (a&&b));
      System.out.println("a || b = " + (a||b) );
      System.out.println("!(a && b) = " + !(a && b)); } }
```

“instanceof” Operatörü

```
public class Test {

   public static void main(String args[]) {

      String name = "James";

      boolean result = name instanceof String;
      System.out.println( result );
   }
}
```

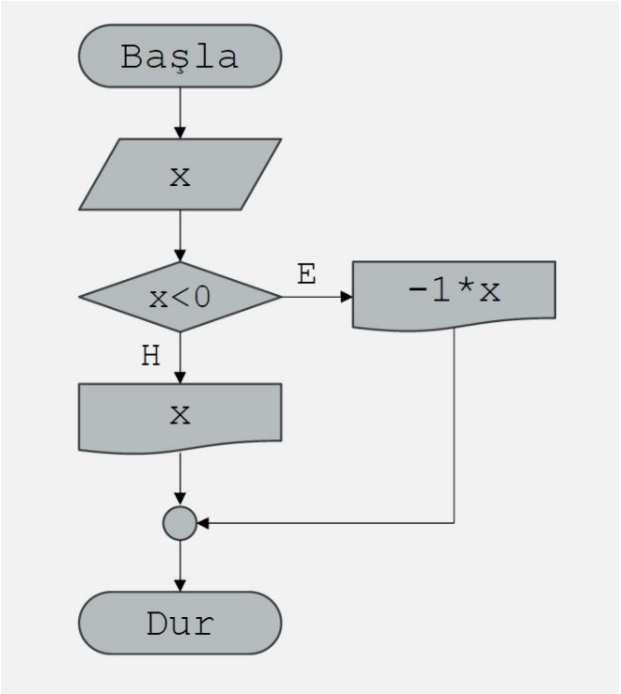
Java Kod Örnekleri

- Sağ tarafta yer alan akış diyagramını Java ile kodlayınız.



Java Kod Örnekleri

- Sağ tarafta yer alan akış diyagramını kodu Java ile kodlayınız.



Java Kod Örnekleri

- Aşağıda yer alan sözde kodu Java ile kodlayınız.

Çözüm;

1. Başla
2. a ve b değerlerini oku
3. m=0
4. a>=b olduğu sürece tekrarla

4.1 a=a-b

4.2 m = m + 1

5. kalan a ve bölüm m ‘yi yaz
6. Son

Java Kod Örnekleri

- Aşağıda yer alan sözde kodu Java ile kodlayınız.

Çözüm;

1. Başla
2. T=0, i=0
3. i<101 olduğu sürece tekrarla

3.1 m değerini oku

3.2 T = T + m

3.3 i = i + 1

4. T = T / 100
5. Ortalama T ‘yi yaz
6. Son

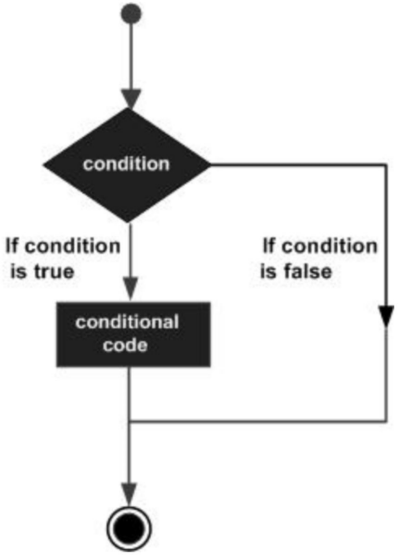
Java Karar Mekanizmaları

Karar Mekanizmaları (if-else)

- Java dilinde if-else yapıları ile program akışını dallandırabiliriz ve başka bir yönden ilerlemesini sağlayabiliriz. Rayları değiştiren makaslar gibi düşünebiliriz. Eğer koşulu sağlıyorsa bu komutları çalıştır, eğer sağlamıyorsa şu komutları çalıştır gibi gündelik ifadeyle izah edebiliriz.

```
if(age < 50) {
        // personel kayıtlarını getir
}
```

```
if(creditRatio > 0.7) {
        System.out.println("Kurumsal müşteri tipinde kredi");
}
else {
        System.out.println("Standart müşteri tipinde kredi");
}
```

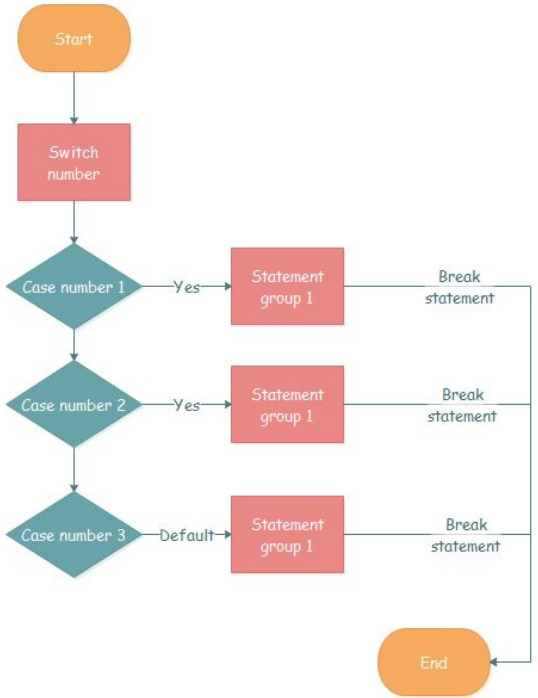


Karar Mekanizmaları (Soru işareti (?) Operatörü)

```
public class Test {

   public static void main(String args[]) {
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( "Value of b is : " +  b );

      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

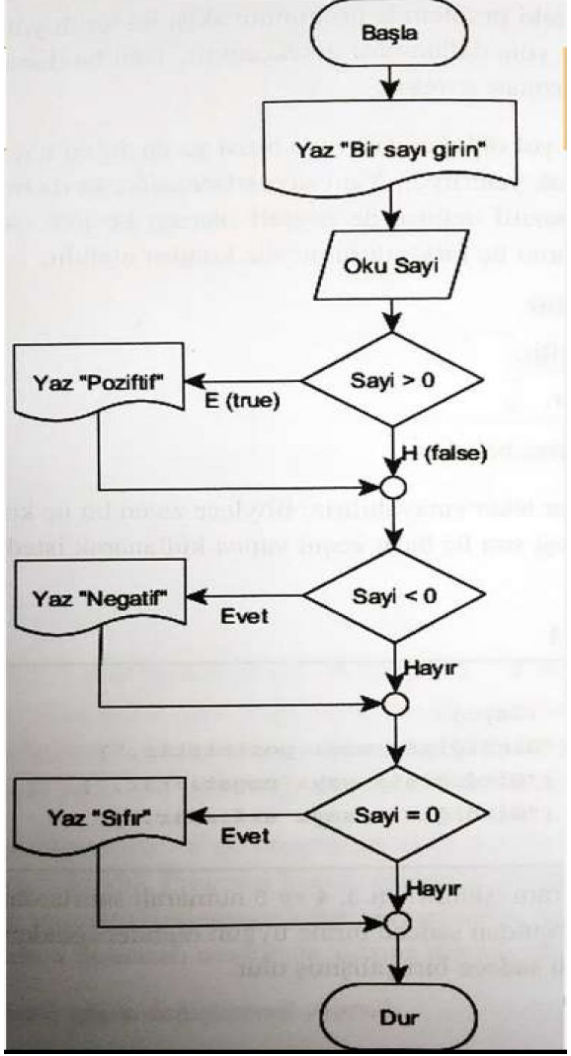
Karar Mekanizmaları (switch-case)

```
switch (operationChoice) {
        case 0:
                System.out.println("Diğer işlemler menüsü");
                break;
        case 1:
                System.out.println("Vize işlemleri");
                break;
        case 2:
                System.out.println("Kredi kartı işlemleri");
                break;
        case 3:
                System.out.println("Ev kredisi işlemleri");
                break;
        case 4:
                System.out.println("Müşteri temsilcisi işlemleri");
                break;
        default:
                System.out.println("Lütfen geçerli bir işlem tipi
seçiniz");
}
```



Karar Mekanizmaları Örnek

- Sağ tarafta yer alan akış diyagramını Java ile kodlayınız.



Java Döngü Mekanizmaları

Döngü Mekanizmaları

- Döngüler 3 ana bileşenden oluşur:

  1- Döngünün iterasyon sayısını tutacak değişkene **ilk değer** atanır.

  2- Döngünün sonlandırılması veya devam etmesi için bir **koşul cümlesi** belirtilir.

  3- Döngünün her iterasyonda **ne kadar artıp ne kadar azalacağı** belirtilir.

Döngü: Tekrarlamalı işlemler döngü olarak adlandırılır. Şeklin içine döngünün başlangıç, bitiş ve artış (adım) değerleri yazılır.

Döngü Mekanizmaları (“for” döngüsü)

- -100 ile 1000 arasında döngü içinde ilerleyip negatif veya pozitif tek sayıları bulan algoritmanın Java kodu aşağıdaki gibidir:

```
for( int i=-100; i < 1000; i++) {
        // tek sayıları bul
        if(i % 2 == -1 || i % 2 == 1) {
                System.out.println("Tek sayı: " + i);
        }
}
```

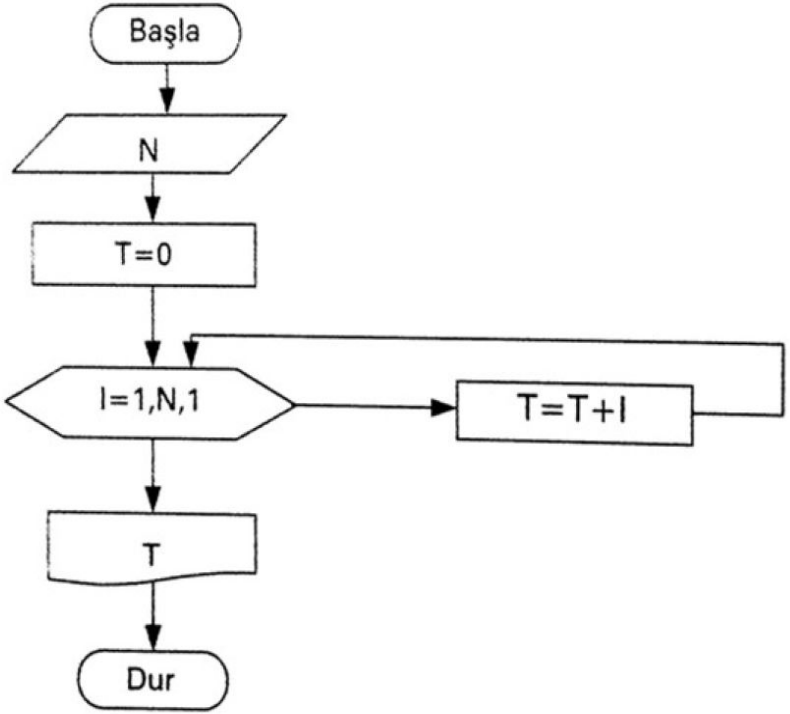
Döngü Mekanizmaları (“while” döngüsü)

- Klavyeden girilen değer ile şifreyi kıyaslayıp kullanıcının sisteme girişini sağlayan döngü algoritmasıdır.

```
Scanner scanner = new Scanner(System.in);
String customerPassword = "12345";
boolean passwordSuccessfull = false;

while(!passwordSuccessfull) {

        System.out.println("Hesap şifrenizi giriniz:");
        String typedPassword = scanner.next();
        if(customerPassword.contentEquals(typedPassword)) {
                passwordSuccessfull = true;
                System.out.println("Sisteme başarılı giriş yaptınız!");
        }
}
```

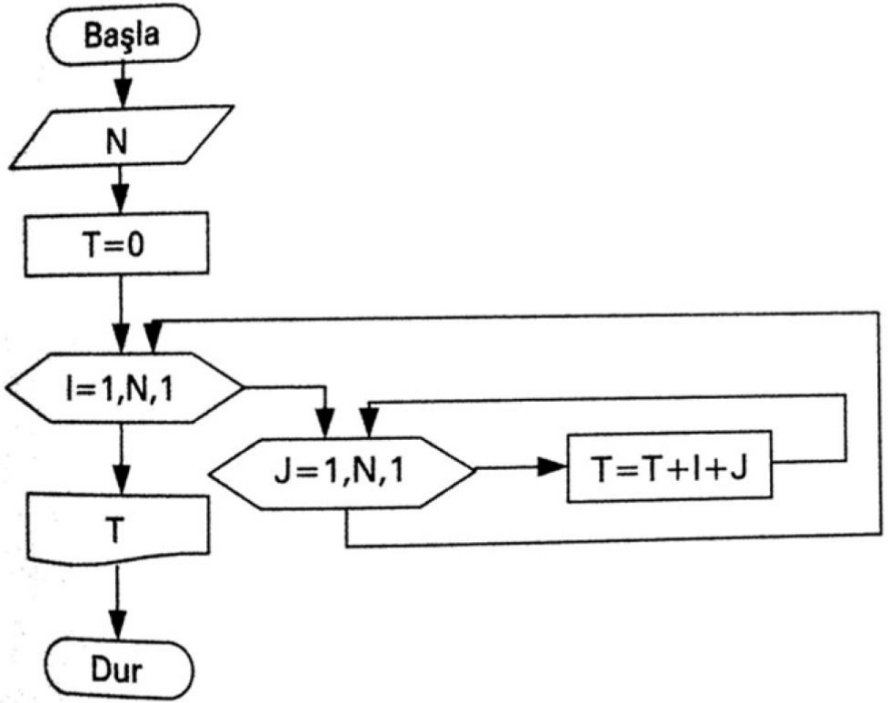
Döngü Örnek

- Sağ tarafta yer alan akış diyagramını Java ile kodlayınız.



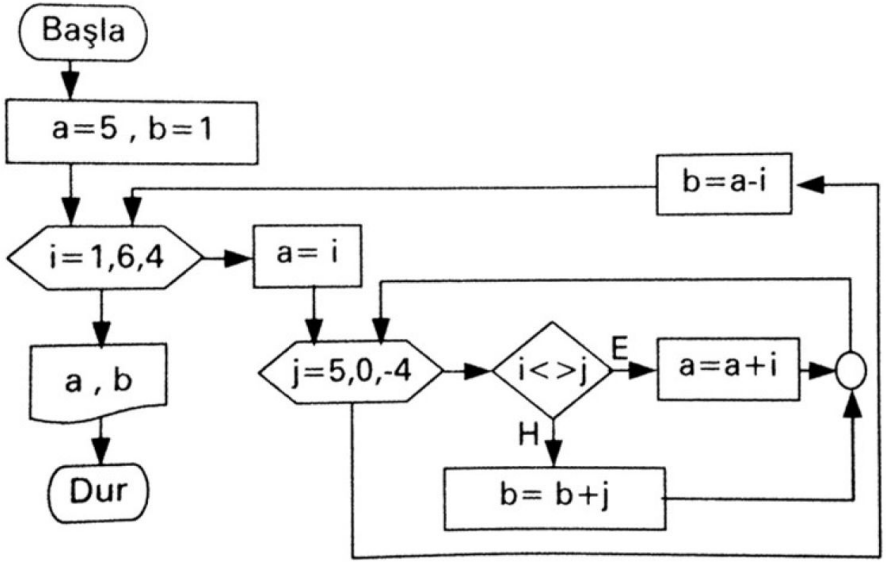
Döngü Örnek

- Sağ tarafta yer alan akış diyagramını Java ile kodlayınız.



Döngü Örnek

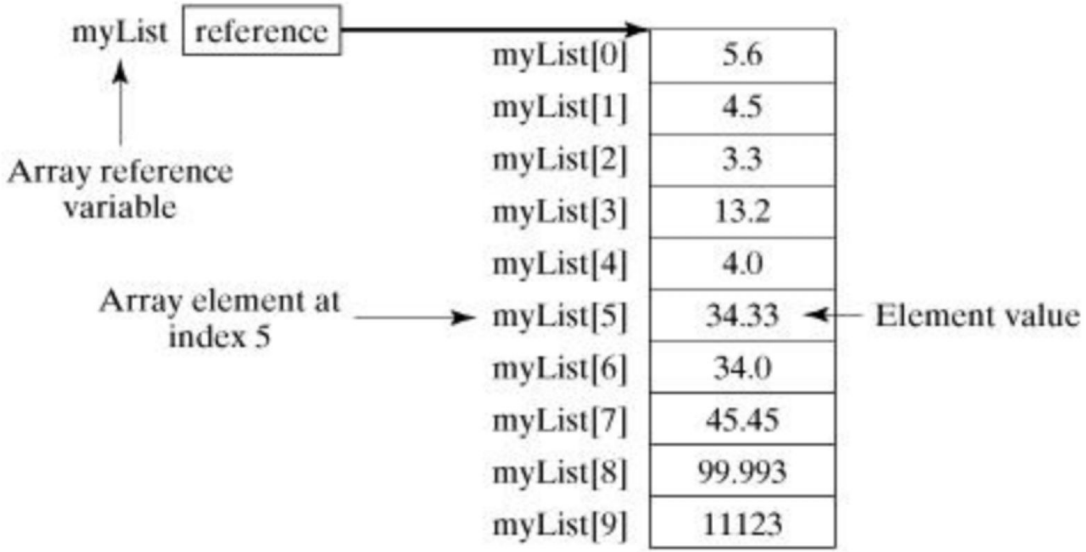
- Sağ tarafta yer alan akış diyagramını Java ile kodlayınız.



Java Diziler (Arrays)

Diziler

- Dizi (Array) kavramı programlama dillerinde bir veri tipini ifade eder.
- Bu veri tipi liste halindeki ardışık verileri bir arada tutan yapıya denilir.
- Bu ardışık yapıya ait elemanlara indeks yoluyla erişim sağlanabilir. Diziler sabit boyutludur.



Diziler

```java
// Java'da diziye ilk değerler süslü parantezler arasında verilir.
double[] myList = { 1.9, 2.9, 3.4, 3.5 };

// tüm dizi elemanlarını arada boşuk bırakarak sırayla ekrana yazdırır.
for (int i = 0; i < myList.length; i++)
{

        System.out.println(myList[i] + " ");

}
```

Diziler Örnek

- **Örnek:** Rastgele tam sayılardan oluşan “dizi1” ve “dizi2” isminde iki tane dizi tanımlayınız. Ardından, bu iki diziyi toplayıp “diziSonuc” isminde tek bir dizi haline getiren algoritmayı yazınız.

Java Diziler (Matrisler)

Matrisler

- Java’da matrisler varsayılan bir veri tipi olarak bulunmazlar. Dizilerin 2 boyutlu halleri şeklinde tanımlanırlar.
- Matris satır ve sütun şeklinde tablo verisi formatındaki verileri tutmak için kullanılır.

```
double[][] B = {
  new double[]{1d, 2d, 3d, 7d},
  new double[]{5d, 2d, 8d, 1d}
};
```

2x4 ‘lük başka bir ise aşağıdaki gibi olacaktır.

$$B = \begin{bmatrix} 1 & 2 & 3 & 7 \\ 5 & 2 & 8 & 1 \end{bmatrix}$$

3x2 ‘lik bir matris olduğunu düşünelim.

$$A = \begin{bmatrix} 1 & 5 \\ 2 & 3 \\ 1 & 7 \end{bmatrix}$$

Matris Örnek

- **Örnek-1:** Rastgele sayılardan oluşan 3x5'lik bir matris oluşturunuz. Ardından, matrisi formatlı bir şekilde ekrana yazdırınız.

- **Örnek-2:** Rastgele sayılardan oluşan 3x5'lik "matris1" ve "matris2" isminde iki matris oluşturunuz. Bu iki matrisi toplayıp "matrisSonuc" ismindeki 3x5'lik matrisi oluşturup ve ekrana yazdırınız.

Java Fonksiyonlar

Java Fonksiyonlar

- Fonksiyonlar sayesinde programlar **küçük parçalar halinde yazılabilir** ve daha sonra bu parçalar birleştirilip bir bütün elde edilebilir.
- Fonksiyonlar ile ilgili teknik detaylara gelecek olursak **her fonksiyonun mutlaka bir tipi olmak zorundadır.** Program hangi tipte ise o tipte bir değeri return etmelidir. Yani fonksiyon çalışıp görevini yerine getirdikten sonra bir geri dönüş gerçekleştirmelidir.
- Eğer bir geri dönüş beklenmiyorsa fonksiyonun tipi **void** olmalıdır.
- Her fonksiyonun bir adı olmalı ve adından sonra **( )** parantezler gelmeli. Bu **parantezler içine eğer varsa parametreler girilir.** Parametre olmak zorunda değildir. Ve **fonksiyonun ne yapacağı ise { }** süslü parantezler içinde belirtilmelidir.

Fonksiyonun Morfolojik Yapısı

```
<erişim belirteci> <dönüş tipi>  <fonksiyon ismi>  ( <eğer gerekliyse parametreler> )
    public              void        ekranaYazdir        (long sayi1, long sayi2)


// Değer döndürmeyen fonksiyon
public void kullaniciSiparisleriYazdir()
{
   // fonksiyonun yaptığı işlemlere ait Java kodları ...
}

// Değer döndüren fonksiyon
public int topla(int sayi1, int sayi2)
{
    // return ifadesiyle değer döndürdüğü belirtiliyor
    return sayi1 + sayi2;
}
```

Fonksiyon Örnek

- **Örnek-1**: Kullanıcıdan int tipinde iki sayıyı ekrandan alınız. Ardından, kullanıcının seçebileceği 4 işlem tipini ekrana yazdırınız. Kullanıcının 4 işlemden birini seçmesini sağlayınız. Kullanıcı çıkış tuşuna (“Q”) basmadığı sürece matematiksel işlem yapabilmesine izin veriniz. Bu 4 matematiksel işlemi (toplama, çıkarma, çarpma ve bölme) Java’da fonksiyonları kullanarak yazınız.

Java Sınıf ve Nesne

Java’da Sınıflar (Classes)

```
public class Dog {

    String breed;
    int age;
    String color;

    void barking() {
    }

    void hungry() {
    }

    void sleeping() {
    }
}
```

Java’da Nesneler (Objects)

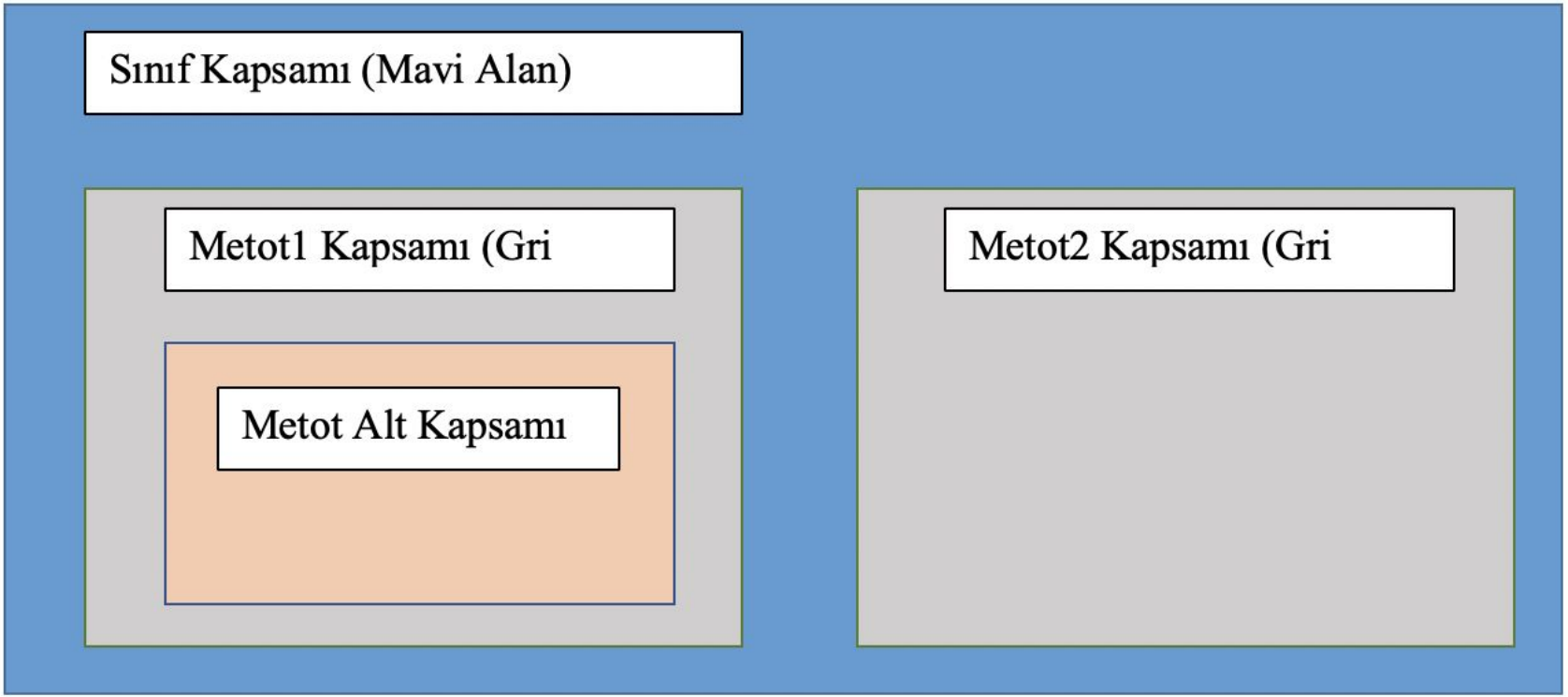
Gerçek hayata döndüğümüzde etrafımızda yüzlerce nesne görürüz. Aslında, her nesnenin var olan bir durumu ve davranışı vardır.

Örneğin: bir köpeği ele aldığımızda rengi, ismi, cinsi köpeğe ait durumu ifade eder. Havlaması, koşmak, acıkması ise onun davranışlarını ifade eder.

Java’da Kod Kapsam Blokları

```
public class ConnectionPool
{ // Sınıf kapsamının başlangıcı

        int connectionMaximumLimit = 50; // Nesne değişkenidir.

        static int currentActiveConnectionCount = 10; //static değişkendir.
Sınıf değişkenidir.

        public void acquireConnection()
        { // metot (fonksiyon) kapsamının başlangıcı

                int processId = 90; // Yerel değişkendir.

                // Diğer kodlar ...

        } // metot (fonksiyon) kapsamının sonu

} // Sınıf kapsamının sonu
```

Java’da Kod Kapsam Blokları



Java’da Sınıf Kurucuları (Constructors)

```java
public class Puppy {

   public Puppy() {
     // parametresiz bir kurucu metot.
   }

   public Puppy(String name) {
      // name isminde bir değişkenle tanımlanmış bir kurucu metot.
   }
}
```

Java’da Sınıf Kurucuları (Constructors)

```
Puppy p = new Puppy();
```

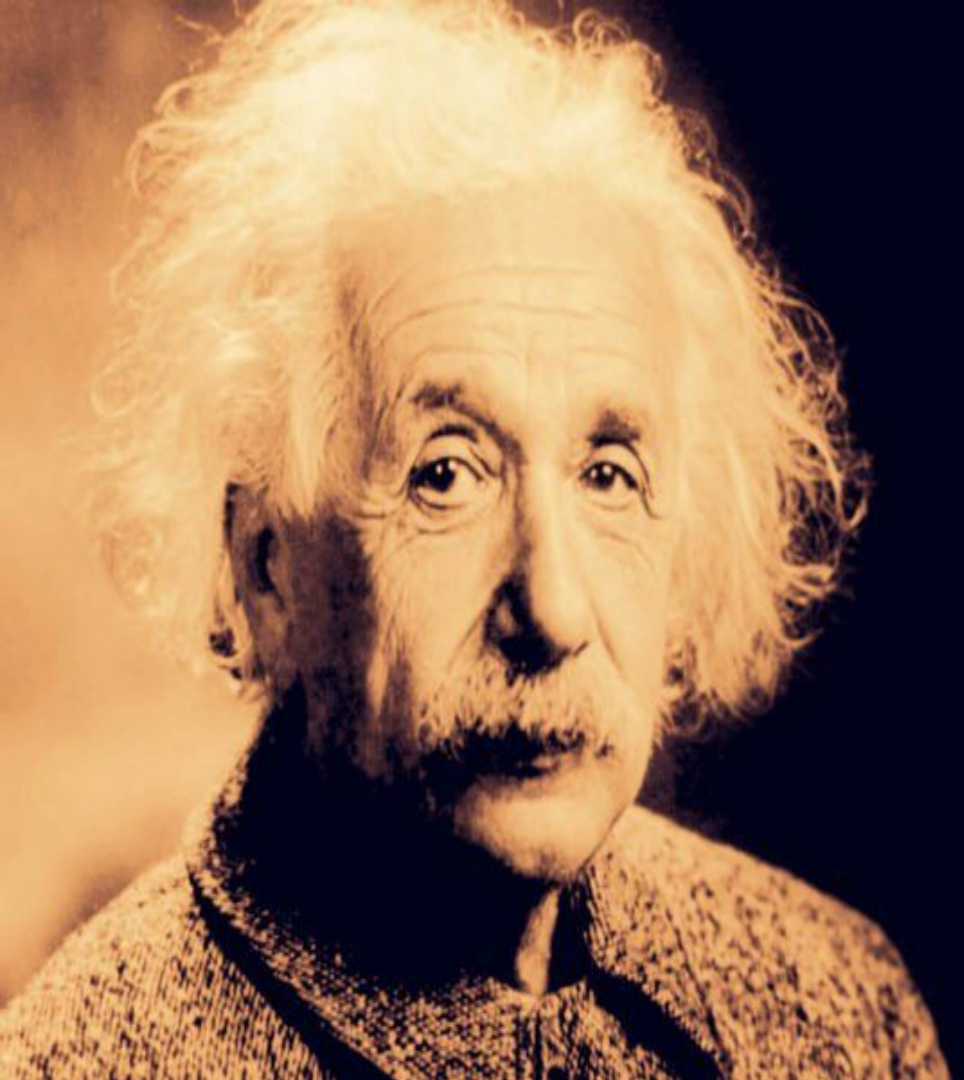
```
Puppy p = new Puppy("Pamuk");
```

Java’da Paketler (Packages)

Java’da sınıfları (classes) ve arayüzleri (interfaces) bir araya toplamak için paket kavramı vardır.

Birbiriyle mantıksal ilişkiye sahip sınıflar bir paket altına hiyerarşik bir şekilde toplanır.

```
import java.io.*;
```



“We now accept the fact that learning is a lifelong process of keeping abreast of change.”

Albert Einstein

Teşekkürler!

/in/batuhanduzgun

@batux

batuhan.duzgun@windowslive.com
www.batuhanduzgun.com