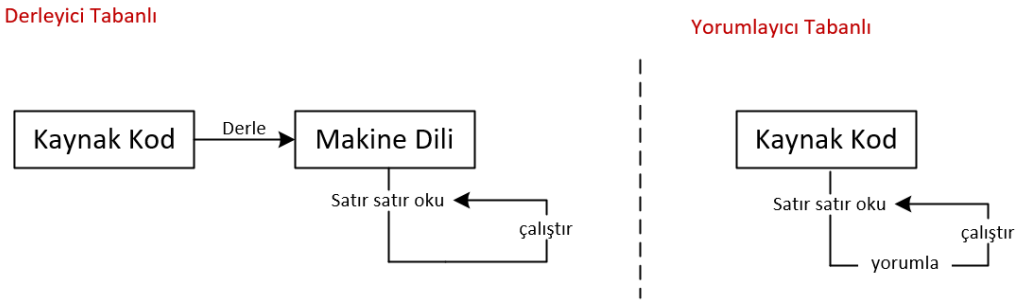
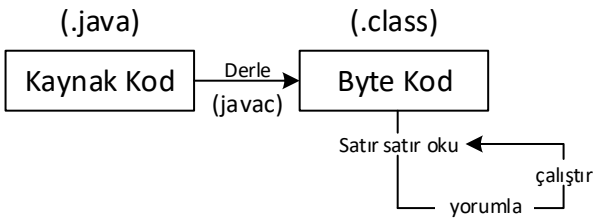
TEMEL KAVRAMLAR



- Kodda yapılan değişiklerin yansıması için derleme gerektirir
+ Doğrudan makinenin anlayacağı komutlara sahip olduğundan hızlı çalışır
- Farklı komut setlerine sahip işlemciler için farklı derleme sürümleri gerekebilir

+ Derleme işlemi olmadığından kodda güncelleme kolaydır
- Çalışma anında programlama dili – makine dili çevrimi yapıldığından nispeten yavaş çalışır
+ İşletim sistemi üzerindeki yorumlayıcı anlık olarak komut setine uygun kod dönüşümünü yaptığından aynı kod her bilgisayarda çalışabilir

Şekil 1. Derleyici ve yorumlayıcı tabanlı dillerin farkları



Şekil 2. Java çalışma yapısı (hem derlenir, hem yorumlanır). Bu sayede byte kod komut seti farketmeksizin JRM yüklü her bilgisayarda çalışabilir, diğer taraftan yorumlanabilir dillere göre daha optimize bir ara kod çıktısı (byte kod) ürettiğinden daha hızlı çalışır.

JAVA ÖZET BİLGİLER

- Nesnesi üretilmeden kullanılabilecek fonksiyonlar static olmaldır.
- public static void main(String [] args) özel bir fonksiyon olup sınıf çalıştırıldığında çalışacak olan fonksiyondur.
- if else yapısı aşağıdaki gibidir.
    - if (a==5) { … } else if(a==6) { … } else if(a==7) { … } else { … }
- Eşdeğer while ve for döngü örnekleri aşağıda verilmiştir.

```
int i = 0;
while(i < 5){
    System.out.println(i);
    i++;
}
```

≡

```
int i = 0;
while(i < 5)
    System.out.println(i++);
```

≡

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

- Do while örneği aşağıda verilmiştir.

Ekran çıktısı

10

```
int i= 10;
do {
    System.out.println(i);
}while(i<5);
```

≠

Ekran çıktısı

```
int i= 10;
while(i<5) {
    System.out.println(i);
}
```

- Tek satırda dizi elemanların eklenmesi için dizi ve matrisler aşağıdaki şekilde oluşturulabilir.
  - int [] dizi = {3,5,7,9};
  - int [][] matris = {{3,5},{7,9}};
- Dizideki elemanları gezen alternatif for örneği aşağıda verilmiştir (diğer dillerdeki foreach kavramına karşılık gelmekte).

```
int dizi [] = {1,2,3,4};
for (int a : dizi) {
    System.out.println(a);
}
```

≡

```
int dizi [] = {1,2,3,4};
for (int a : dizi)
    System.out.println(a);
```

- Döngü ve karşılaştırma ifadelerinde süslü parantez açılmadığında tek satır (komut) if bloğunun içerisindeymiş gibi işlem görür (yukarda verilen örnekteki gibi).
- Süslü parantez kullanılmadan if içerisinde if yazıldığında sonra gelen else en son yazılan if'e aitmiş gibi işlem görür (solda). Üstteki if'e ait else tanımlamak için süslü parantez kullanılması gerekir (sağda).

```
if(a>3)
    if(a==5)
        System.out.println("Beş");
    else
        System.out.println("3'ten büyük ama 5 değil");
```

```
if(a>3) {
    if(a==5)
        System.out.println("Beş");
} else
    System.out.println("3 veya 3'ten küçük");
```

- Noktalı virgül bulunan döngü örneği

```
public static void main(String[] args) {
    int i = 20;
    for (int j = 0; j < 10; i++, j++); {
        i = i + 5;
    }
    System.out.println(i);
}
```

Ekran çıktısı

35

≠

```
public static void main(String[] args) {
    int i = 20;
    for (int j = 0; j < 10; i++, j++) {
        i = i + 5;
    }
    System.out.println(i);
}
```

Ekran çıktısı

80

- j for içerisinde tanımlandığından dışarıda erişilemez. Hata verir!

```
public static void main(String[] args) {
    int i = 20;
    for (int j = 0; j < 10; i++, j++) {
        i = i + 5;
    }
    System.out.println(j);
}
```

- for içerisinde j değeri artmadığından sonsuz döngü meydana gelir.

```
public static void main(String[] args) {
    int i = 20;
    for (int j = 0; j < 10; i++); {
        i = i + 5;
    }
    System.out.println(i);
}
```

RANDOM SAYI ÜRETME

- **Math.Random()** -> [0-1) arası rastgele sayı üretir (1 dahil değil).
    - [0 50] arası bir tam sayı üretmek için (0 ve 50 dahil):
      (int) (Math.Random()*51)
    - [10 50] arası bir tam sayı üretmek için:
      10 + (int) (Math.Random()*41)
- **Random r = new Random()** -> Random sayı üretmek için nesne oluşturur.
    - **r.nextInt()** -> min integer ile max integer değer arası rastgele tamsayı üretir.
    - **r.nextInt(10)** -> [0, 10) arasında bir tam sayı üretir (10 dahil değil)
    - [10 50] arası bir tam sayı üretmek için:
      10 + r.nextInt(41)