

Programlama Dilleri Ödevi Java Dili

Dilin Tarihçesi

1991 yılında Sun Microsystems şirketi mühendislerinden James Gosling ve 12 arkadaşı Green Project(Yeşil Proje) isimli bir proje geliştirmeye başladılar ve takımın ismini de Green Team(Yeşil Takım) koydular.

Gömülü Sistemler üzerine çalışan James Gosling ve ekibi başlangıçta C ve C++ dillerini kullanılsalar da bu dillerin geliştirdikleri projeye uygun bir dil olmadığını ve geliştirdikleri proje için yetersiz olduğunu görüp yeni bir arayış içine girdiler.

İşte bu sırada “Oak” şimdiki adıyla Java dili doğmuş oldu. Ekibin geliştirmeye başladığı dilin ismini takım liderlerinden James Gosling koymuştu; fakat ilerleyen zamanlarda bu ismin ticari bir amaçla kullanılamayacağı ortaya çıktı. Yeni bir isim arayışına giren ekip bir gün bir kafede oturup kahvelerini yudumlarken yeni geliştirdikleri dilin adının içtikleri kahvenin adı olan Java olması kararını aldılar. O günden bugüne adı Java olan programlama dili bugün milyarlarca cihazda aktif şekilde çalışmaktadır.

Tasarım amacı

Java basittir. Java'yı tasarlayanlar kaynak programın kolay yazılabilmesini, kolay derlenmesini ve kolay düzeltilemesini (debug) amaçladılar.

Java nesne yönelimli bir programlama dilidir. Nesne yönelimli programlama paradigmاسının bütün avantajlarını taşır. Programcıya kalıtım, polimorfizm, modular programlama, hata ayıklama (debug) ve kodların yeniden kullanılabilmesi gibi önemli yetenekleri sunar.

Java dağıtık bir sistem olma niteliğine sahiptir. Bir ağ üzerindeki birden çok farklı bilgisayarın bütünlilik bir sistem olarak bir arada çalışmasını sağlar.

Java çoklu iş yapma (multithreaded) yeteneğine sahiptir. Çoklu iş yapma niteliği, bilgisayarın aynı anda birden çok işi yapabilmesi demektir. Başka dillerde sistemle ilgili prosedürlerin çağrılmalarıyla yaptırılan multithreaded özelliği java dilinin özünde vardır. Multithreaded yeteneği, özellikle görsel programlamada ve ağ programlamada önem kazanır.

Java platform bağımsızdır. Java programları harklı platformlar için ayrı ayrı değil, JVM için bir kez derlenir. Derleme sonunda ortaya çıkan java bytecode JVM tarafından yorumlanır. Bytecode, JVM yüklü her makinede çalışabilir. JVM sanal makinesi her makineye kolayca ve ücretsiz yüklenebilir. Dolayısıyla, java programları bir kez yazılır ve her yerde çalışır.

Java taşınabilir. İşletim sisteminden ve donanımdan bağımsız oluşu nedeniyle, Java Bytecode bir bilgisayar sisteminden farklı bir başkasına kolayca taşınır. Aynı java programının farklı sistemlerde sorunsuz çalışma yeteneği, programlama alanında geniş ufuklar açmıştır.

Java sağlamdır. Başka dillerin ancak koşturma anında belirleyebilecegi hataları, java derleme anında belirler. Güçlü hata ayıklama (debug) yeteneği vardır.

Java güvenlidir. Java dili, derleyicisi ve yorumlayıcısı güvenlik öncelikli olarak tasarlanmıştır. Tasarımında güvenliği öne çikaran ilk dildir.

Java Ağ dostudur. Java'da ağ programı yazmak, dosyalara veri gönderip veri almak kadar kolay bir iştir.

Dilin kullanım alanları ve Hedef kitlesi

Java platformdan bağımsız çalışan, nesne tabanlı, yüksek hızlı, basit tasarımlı, güvenli, dinamik bir programlama dilidir. Java Windows sürümleri, Mac OS, HP-Unix, Sun Solaris, Redhat Linux, Ubuntu, CentOS gibi platformlarda çalışabilir.

Java'nın kullanım alanlarını şöyle tarif edebiliriz:

- Herhangi bir platformda yazılan yazılımı diğer bir sanal platformda çalışırmak
- Web tarayıcısı ve erişilebilir Web hizmetleriyle çalışacak programlar oluşturma
- Çevrimiçi forumlar, mağazalar, anketler, HTML formlarını işleme ve daha fazlası için sunucu tarafı uygulamaları geliştirme
- Üst seviyede özelleştirilmiş uygulamalar ve hizmetler yaratmak için Java dili kullanarak uygulamaları ve hizmetleri birleştirme
- Cep telefonları, uzak işlemciler, mikro denetçiler, kablosuz modüller, sensörler, ağ geçitleri, tüketici ürünleri ve neredeyse tüm elektronik aygıtlar için güçlü ve verimli uygulamalar yazmaya müsaittir.

7

Desteklediği Paradigmalar: Object oriented, imperative, fonksiyonel programlama (FP) tartışmalarında nesne yönelimli (OOP) diller ile fonksiyonel programlama yapılamaz gibi söylemler olur. Bu kesinlikle yanlıştır.

Aritmetik işlem notasyonu: Infix

Bellek Yönetimi: Garbage Collector (Çöp Toplayıcı) java uygulamasının çalışma süresince çıkan ve kullanılmayan nesneleri temizler. Böylece Memory Management (Bellek Yönetimi) yükü JVM tarafından kontrol edilir. Java'nın C++ diline göre çok daha basit olmasının temel nedeni, otomatik bellek tahsisi yapması ve işi biten nesneleri bellekten yok etmesidir.

Değişken kapsamları: Değişkenlerin görülebildiği ya da erişilebildiği alana, kapsam (scope) denir. Java'da değişkenlerin kapsamları derleme zamanında belirlenir ve çalışma zamanında değişmez. Bu yüzden Java, statik kapsamlı (lexically scoped) dilleridir.

Tip sistemi ve Tip kontrolü : Tip sistemi güçlü(strong) tip kontrolü ise dynamic type .

KODLAR

Hello world

```
package helloworld;
```

```
public class Helloworld {  
  
    public static void main(String[] args)  
    {  
        System.out.println("Hello, World");  
    }  
  
}
```

Asal Sayı

```
package asalsayi;  
import java.util.Scanner;  
  
public class AsalSayi {  
  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        System.out.println("Üst Değer Limiti Giriniz : ");  
        int deger = scan.nextInt();  
        int kontrol = 0;  
  
        if(deger<2){  
            System.out.println("En küçük asal sayı 2 dir.");  
        }  
  
        for(int i=2; i<=deger;i++){  
  
            for(int j=2;j<i;j++){  
  
                if(i%j == 0){  
                    kontrol = 1;  
                }  
            }  
  
            if(kontrol == 0){  
                System.out.println(i);  
            }else{  
                kontrol = 0;  
            }  
        }  
    }  
}
```

Fibonacci

```
package fibonacci;

public class Fibonacci {

    public static void main(String[] args) {
        int n = 9;
        System.out.println(fib(n));
    }
    static int fib(int n)
    {
        if (n <= 1)
            return n;
        return fib(n-1) + fib(n-2);
    }
}
```

Mükemmel Sayı

```
package mukemmelsayi;
import java.util.Scanner;
public class MukemmelSayi {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int sayac=0;

        System.out.println("Sayı giriniz: ");

        int sayi = scan.nextInt();

        for (int i=1;i<=sayi/2;i++){
            if(sayi%i==0){

                sayac+=i;

            }
        }

        if(sayac==sayi) System.out.println(sayi+" sayısı mükemmel sayıdır.");
        else System.out.println(sayi+" sayısı mükemmel sayı değildir.");

    }
}
```

Insertion Sort

```
package insertionsort;
public class InsertionSort {

    public static void main(String[] args) {

        int array[] = { 12, 11, 13, 5, 6 };

        InsertionSort ob = new InsertionSort();
        ob.sort(array);

        printArray(array);
    } void sort(int array[])
    {
        int n = array.length;
        for (int i = 1; i < n; ++i) {
            int key = array[i];
            int j = i - 1;

            while (j >= 0 && array[j] > key) {
                array[j + 1] = array[j];
                j = j - 1;
            }
            array[j + 1] = key;
        }
    }

    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }
}
```