8 ocak Innova Patika spring bootcamp ödevi

Öğrenci : Ozan Aydoğan
Öğretmen : Halit Mızrak

Compiler

Herhangi bir programlama dilinde belli bir amaç için yazılmış kodun, bytecode yapısına dönüştürülmesini sağlayan programdır.

java'da, java compiler, kaynak kodlarımızı yani .java uzantılı dosyalarımızı okur anlar ve soyut class yapıları oluşturur(.class). bu yapılara bytecode denir.

C, C++ gibi diller sadece compiler kullanır.

Interpreter

(yorumlayıcı) JVM bir interpreterdir. Java compiler'da gerçekleşen işlemler sonucunda elde edilen bytecode'u yorumlayıp işlemcinin anlayacağı talimatlar dizisine dönüştüren, yani makine koduna (binary code) dönüştüren yapıdır. Java, Python yazılım dilleri interpreter kullanır.

Interpreter’da kod hata alana kadar çalışır. hata aldığı anda durur bu yüzden hata ayıklamak daha kolaydır.

Compiler ve Interpreter

Java platform bağımsız bir dildir ve kendisi bir platformdur. Yapısında bulundurduğu Compiler ile herhangi bir platformda yazılan kaynak kodu derleyip bytecode yapısına dönüştürür. Linux, windows, MacOS işletim sistemleri için oluşturulan JVM yapılarıyla (interpreter) bytecode yorumlama işlemi gerçekleşir ve Bir kere yazılan kod platform farketmeden her yerde çalıştırılabilir.

Compiler ve Interpreter

- Compiler bir programı bütün olarak alır ve çevirir, Interpreter programı satır satır çevirir.
- Compiler, ara kod veya hedef kod oluşturur fakat Interpreter herhangi bir ara kod oluşturmaz.
- Compiler'da bir hata oluştuğunda çeviri işlemi durur ve hata giderildikten sonra bütün program yeniden çevirilir. Interpreter eğer bir hata meydana gelirse o anki çeviriyi engeller ve hata giderildiğinde kaldığı yerden devam eder. Bu yüzden Debug işlemi daha kolaydır.

JVM (Java visual machine)

Java compiler tarafından derlenen .java uzantılı dosyalar sonucunda oluşan .class uzantılı dosyaları (bytecode'ları), işlemcinin anlayacağı dil olan makine diline çevirir. JRE ve JDK'nın bir bileşenidir.

JRE (Java runtime environment)

Java programlama dili ile yazılmış olan uygulama ve appletlerin çalışmasını sağlayan bileşenler ile JVM'e kütüphaneler sağlar. JRE = JVM + Java Kütüphaneleri’dir

JDK (Java Development Kit)

Java'da geliştirme yapmak isteyen her developerin mutlaka indirmesi gereken bir bileşendir. yapısında JVM ve JRE yapılarını bulundururken ayrıca, debuggers, compile araçları, javac gibi yapılarıda barındırır. JDK = JRE + Compiler + Debugger

JIT (Just in time)

JVM yapısının altında olan bir bileşendir. Kaynak kodlar derlenerek .class uzantılı dosyalar elde ediliyordu. Bu dosyalar JVM yapısında makine kodlarına dönüştürülerek işlemler gerçekleştiriliyordu. 2 kere derleme işlemi gerçekleştiği için bu durum hız kaybına sebep oluyordu,

JIT sayesinde sadece kullanılacak olan bytecode kısmı çalıştırılır. diğer kısımlar yorumlanır. böylece tekrar tekrar derleme işlemi gerçekleşmediği için hız artışı sağlamış oluyoruz.

Stack ve Heap

Ram'in mantıksal bölümleridir. Stackte değer tipler, pointer ve adresler saklanırken, Heap kısmında referans Değerleri saklanır.

Stack ve Heap

Stack'e erişim Heap'e göre oldukça hızlıdır. Stack LIFO mantığıyla çalışmaktadır yani son gelen ilk çıkar. Heap yapısında ise değerler random olarak tutulur. Stack'te primitive tipler, değerlerini kendi üzerlerinde tutarak saklanır (int short boolean ..) Objelerin adresleride Stack'te saklanır. Heap kısmında ise objelerin kendileri saklanır.

Stack ve Heap

NOT:

String değişkeni Referans tiptir fakat bir istisna olarak değer tip gibi davranır. String tipindeki değişkenlerde tutulan veriler char array şeklinde tutulur. Array yapıları referans tiptir fakat istisna olarak string, değer tip gibi davranır.

Wrapper class

javada ilkel bir veri türünü nesneye, nesneyi ise ilkel bir veri türüne dönüştürmek için kullanılır. her primitive veri tipinin bir wrapper sınıfı vardır. örneğin int -> Integer, char -> Character gibi.

unboxing ve autoboxing

unboxing = Bir wrapper türündeki bir nesneye karşılık gelen ilkel değerine dönüştürmeye denir.

autoboxing = ilkel bir değeri onun wrapper sınıfına dönüştürmeye denir.

Wrapper class ve Primitive tiplerin farkı

ilkel veri tiplerine, stackte belli bir alan ayrıldığı için null değer alamazlar fakat wrapper nesneleri bir objedir ve null değeri alabilir.

ArrayList gibi yapılarda wrapper classlar kullanılabilir fakat primitive tipler kullanılamaz.

By pass value, By pass reference

Programlama dillerinde, metotlara parametreler aktarılırken 2 farklı yöntem kullanılır. Birisi değere göre geçirme (by pass value), diğeri referansa göre geçirme (by pass reference). Değere göre geçirme işleminde metota parametre olarak gelen değerin direk metot parametresine kopyalayarak geçirilmesidir. Hafızada bir kopya oluşur ve işlemler gerçekleştirilir. Bu işlemlerden, metoda parametre olarak gelen değişkenler etkilenmez. Java by pass value yaklaşımıyla metotlara değer aktarır.

By pass value, By pass referance

“By pass value”ye örnektir. Aşağıdaki kodda x değeri 5 olarak kalır,

```
void foo(int param) { param++; }

int main()
{
  int x = 5;
  foo(x);
  printf("%d\n",x); // x == 5
}
```

By pass value, By pass referance

By pass referance yaklaşımı ise, metotta bulunan parametrelere, metoda gelen parametrelerin bellek işaretçilerinin atanması olarak gerçekleşir. C ve C++ dillerinde pointer kullanımlarıyla by pass referance yaklaşımı gözle görülebilir.

Serialization

Serileştirme işlemini kullanarak bir nesnenin anlık olarak tuttuğu bilgiler bir yere kaydedilir ve istenildiği zaman aynı şekilde bu bilgiler elde edilir. Serileştirme yapılırken nesne bilgileri, byte olarak kaydedilir. Bu nesneye tekar ulaşılmak istenildiğinde bu byte veriler kullanılarak nesne tekrar elde edilir. Bu şekilde bir nesnenin byte olarak saklanmasına Serialization, bu byte veriden tekrar nesne oluşturulmasına da DeSerialization denir. Yani dosyadaki verilerinizi okuyabilmek için deserialization işlemi yapmanız gerekir.