3. 2 tip dil çeşidi var bunlardan biri compiled diğeri interpreted , compile bir kaynak kodun bir başka forma çevrilmesidir genelliklede makine koduna çevirilir bu makine kodu da platform spesifik olur. Interpreted dillerde de compile işlemi vardır just in time compile işlemi vardır. Kod anında çevrilip çalıştırılır, yorumlanır ve ayrı bir yere kaydedilmez, runtimeda üretilen kod da platform bağımsızlığı yaratır. Üretilen bu kod çalışmak için bir interpreter ister. Java virtual machine bu kodu çalıştırmaya yarar. Java hem compiled hemde interpreted bir dildir.

4. Stack boyutları belli, sabit değerlerin (method-specific) saklandığı alan iken Heap ise objects ve jre classların, değişken değerlerin saklandığı alanlar diyebiliriz. Heap her yaratılan objede bir yer açarken stack method bitince silinir, sadece bir kullanımlıktır. Bir obje heapde tutulurken referanası stackde tutulur.

5. Bir string değişkeni ürettiğimizde değişkenin ismi memoryde üretilen değerin referansı oluyor. Memorydeki ise string objesi oluyor. Örneğin bir isim değişkeni yarattığımızda “Selim” string object olurken isim referansı oluyor string objectler bir havuzda tutuluyor. İsim değişkenine başka bir değer verdiğimizde Java memorydeki değeri değiştirmek yerine havuzda yeni bir değer oluşturuyor ve bu isim değişkenini ona yönlendiriyor.

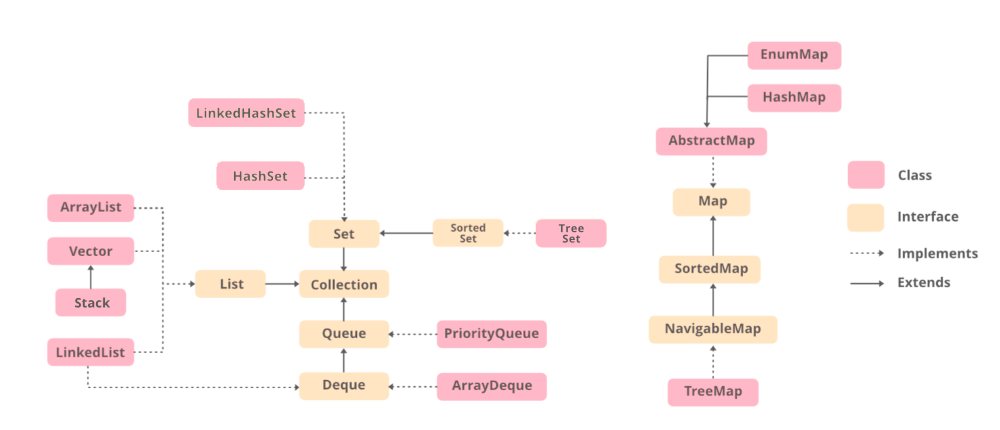
metin, anten içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yani immutable dediğimizde hafızadaki değer immutable oluyor referansı değil.

6. Çoklu kalıtım dediğimiz şey bazı dillerde var oran bir subclassın birden fazla superclassları olmasıdır. Java birden çok subclass olmasını desteklerken birden fazla superclass olmasını desteklemez. Çoklu kalıtım yapılırsa oluşabilecek probremlere “diamond problem” deniyor. Örneğin bir hayvan classımız var ve iki tane subclass kedi ve köpek var bunlarında ses çıkarma methodları var. Kedi ve köpeğin superclass olduğunu ve altında bir evcil hayvan subclassı yaptığımızı düşünelim. Bu subclass ses çıkarma methodunu kullanmak istediğinde hangisinden kalıtım alacağını bilemez. Bunu engellemek için Java çoklu kalıtıma izin vermez.

7. Build toollar kaynak kodu çalışabilir bir uygulamaya çevirme sürecini otamatik bir şekilde yürüten programlardır. Compiling, linking ve packaging gibi tekrar eden işlerin yapılma sürecidir. Apache Ant, Maven, Gradle, TeamCity, MSBuild gibi java build toolları vardır.

8.Collection Frameworks;

Yukarıda collection frameworklerin hiyerarşik yapısını görebiliyoruz. İncelemek gerekirse;

Collection framework yapısı hiyerarşik bir yapıya sahiptir. Bu hiyerarşinin ilk noktası Iterable<T> interface i ile başlar. Bu interface’de kendi içinde Collection <E> interface’ini ortaya çıkartır.

Collection arayüzünün içinde kendi metodları bulunur. Bu metodlar sırasıyla şöyle sıralanabilir;

* boolean add(Object element) : Object eklemek için kullanılır.
* boolean remove (Object element) : Object’ i kaldırmak için kullanılır.
* boolean addAll(Colletions<E>c) : verilen listeyi toplu olarak ekler.
* boolean removeAll(Colletions<E>c) : verilen listeyi toplu olarak çıkartır.
* Boolean isEmpty(): Collectionın doluluğunu kontrol eder ve boşsa true değeri döndürür.
* Boolean contain(Object element) : verilen elemanın bulunup bulunmadığını kontrol eder.
* Boolean containsAll(Collections<E> c ) : Verilen obje listesinin bulunup bulunmadığını kontrol eder.
* Boolean retainAll(Collections <E> c) : Verilen listede olmayan elemanları collectiondan çıkartmak için kullanılır.
* Void clear(): listedeki elemanları çıkartır.
* Object [] toArray () : List’i array’e dönüştürür.
* İnt size () : Eleman sayısını verir.
* <T> T [] toArray( T[] a ) : Array ‘e dönüştürülen liste yapısının içeriği değiştirilebilir.

Bu metodlar dışında List , Set ve Queue arayüzleri sayesinde Collection framework genişletilmiştir.

1. SET

Set arayüzünün temel özelliklerinden biri içerisinde duplike değer tutmamasıdır. SET arayüzü üç ayrı sınıfa ayrılır.

* 1. HashSet : Kapasitesi genişletilebilirdir ve sıralı bir koleksiyon türü değildir.
  2. LinkedHashSet: HashSet’ den farklı olarak elemanları eklenme sırasına göre tutmaktadır.
  3. TreeSet: Elementler artan sıraya göre tutulmaktadır.

Metod örneği:

(Tree set)

**public** **static** **void** main(String[] args) {

TreeSet<String> set= **new** TreeSet<String>();

set.add("Sıralama");

set.add("Artan");

Iterator<String> selim = set.iterator();

**while** (selim.hasNext()) {

System.***out***.println(selim.next());

}

}

}

ÇIKTI :

Artan

Sıralama

(HashSet)

**public** **static** **void** main(String[] args) {

HashSet<String> hs = **new** HashSet<String>();

hs.add("Kodluyoruz");

hs.add("Java");

hs.add("programlama");

hs.add("selim");

Iterator<String> selim = hs.iterator();

**while** (selim.hasNext()) {

System.***out***.println(selim.next());

}

}

}

ÇIKTI :

Java

Kodluyoruz

selim

programlama

1. LIST

Set’ den farklı olarak duplike değer tutabilirler. İçerisinde yeni öğe atandıkça uzunlukları da artar. İçerisindeki elemanları ekleme sırasına göre tutar ve index yoluyla ulaşma fırsatı verir. 4 sınıfa ayrılmıştır.

* 1. ArrayList : Dinamik olarak şekillenebilir ve tek boyutlu dizi sınıfıdır.
  2. LinkedList :Her eleman kendinden sonra gelecek olan elemanın adresini de bilir. Ekleme çıkarma işleri yoğun olduğunda daha çok tercih edilir.
  3. Vector : Boyutu dinamik olarak değiştirilebilir.
     1. Stack : En son kaydedilen elemanı sorguda ilk eleman olarak döndürür. Alt metotları şöyle sıralanabilir;
        + Boş olup olmadığının kontrolü için empty() metodu
        + Sona eleman eklemek için push() metodu
        + Elemanın yerini öğrenmek için search() metodu
        + Son eklenen elemanı görmek için peek() metodu
        + Son elemanı listeden çıkartmak için pop() metodu

Metod örneği:

(ArrayList)

**public** **static** **void** main(String[] args) {

ArrayList<Integer> selim = **new** ArrayList<Integer>();

selim.add(5);

selim.add(1);

System.***out***.println(selim);

selim.remove(0);

selim.add(3);

selim.add(4);

System.***out***.println(selim);

selim.remove(0);

System.***out***.println(selim);

}

}

ÇIKTI :

[5, 1]

[1, 3, 4]

[3, 4]

1. QUEUE

Bu arayüzde ilk kaydedilen eleman ilk olarak dönmektedir. Alt metodlar şöyle sıralanabilir;

* Verilen elemanı ekledikten sonra işlemin başarısız olması durumunda hata alabilmek için add() metodu
* Verilen elemanı ekledikten sonra işlemin başarısız olması durumunda null dönmesi için offer() metodu
* Sıradaki elemanı ulaşabilmek için peek () metodu
* Baştaki elemanı çıkartmak için poll() metodu

Collection hiyerarişisine dahil olmayan ama Collection Framework yapısının bir parçası olan diğer arayüz ise MAP olarak karşımıza çıkmaktadır.

1. MAP

Map arayüzünde depolanmak istenen her döge için bir anahtar oluşmaktadır. Depodaki her öğeye anahtarı aracılığı ile erişilir. Tekrar eden anahtarlar bulunmaz.

* 1. HashMap : Senkronize bir yapısı olmadığı için verilerin sırasına dair bir garanti verilmez.
  2. TreeMap : Elemanları artan sıraya göre tutmaktadır.

Metod örneği:

(HashMap)

**public** **static** **void** main(String[] args) {

Map<Integer, String> map = **new** HashMap<>();

map.put(0, "JAVA");

map.put(1, "KODLUYORUZ");

map.put(2, "SELIM");

System.***out***.println(map);

map.remove(1);

System.***out***.println(map);

}

}

ÇIKTI :

{0=JAVA, 1=KODLUYORUZ, 2=SELIM}

{0=JAVA, 2=SELIM}