1.HAFTA JAVA BOOTCAMP ÖDEVİ

3) Java'nın platform bağımsızlığını nasıl sağladığını anlatınız

Java'nın platformdan bağımsız olmasının sebebi, doğrudan makine diline derlenmemesidir. Diğer diller makinenin işletim sisteminde derlendiği için platform bağımlıdır. Bu durum Java'da tam tersidir. Java kodu **JVM** (Java Sanal Makinesi) 'nin yorumlayabileceği **Bytecode**'larına dönüşür. Bu bytecode'ları için çalıştıkları işlemci önemsizdir. Aynı program nerede derlenirse derlensin aynı bytecode'u üretir. Bu da Java'nın **"Bir kere yaz her yerde çalıştır"** prensibidir.

4) Java'da heap ve stack kavramlarını örneklerle açıklayın

Stack ve Heap Ram'de bulunan mantıksal bölümlerdir. **Stack**' de Primivite tip olan int, short, byte, long, decimal, double, float gibi tipler tutulur.

Heap' de ise referans değerleri tutulmaktadır. Diziler ,string, array, interface, class referans tiplerdir. Stack Heap'den daha hızlıdır.

Örnek verecek olursak; primitive tiplerde (==) kullanırsak bu değişkenlerin içindeki değerlerin eşit olup olmadığını kontrol ederiz. Referans tiplerde ise (==) nesnelerin adreslerinin eşit olup olmadığını kontrol ederiz.

Stack: int a=5 dersek 5 rakamı stack de yer alır.

Heap: String name = "Metin" dersek name stack' de tutulurken, "Metin" heap'de tutulur.

5) String class'ı nasıl immutable olmayı sağlamaktadır örnek ve çizimlerle açıklayınız

Immutable (değişmez), nesneler bir kez oluşturulduktan sonra içeriği değiştirilemeyen sınıflardır. String, Integer, Long, Double, Boolean immutable sınıflardır.

- Class içerisinde tanımladığımız değişkenler **private** olursa dışarıdan herhangi bir müdahale olmaz.
- Class içerisinde setter metodları olmamalı. Çünkü **setter** metodu ile kullanıcı değişkenler üzerinde değişiklik yapabilir.
- Değişkenleri final olarak tanımlayabiliriz. **Final** olarak belirtilen değişkenler sonradan değiştirilemezler.
- Classımızı final olarak oluşturursak **extend** alınmasını önlemiş oluruz.

```
public final class Product { // Sınıfı final tanımlarız.
private final String name;  //Değişkeni hem final hem private tanımladık.
    public Product(String name) {  //Constructor
        this.name = name;
```

```
    }
    public String getName() {  //Getter metodu
        return name;
    }
}
```

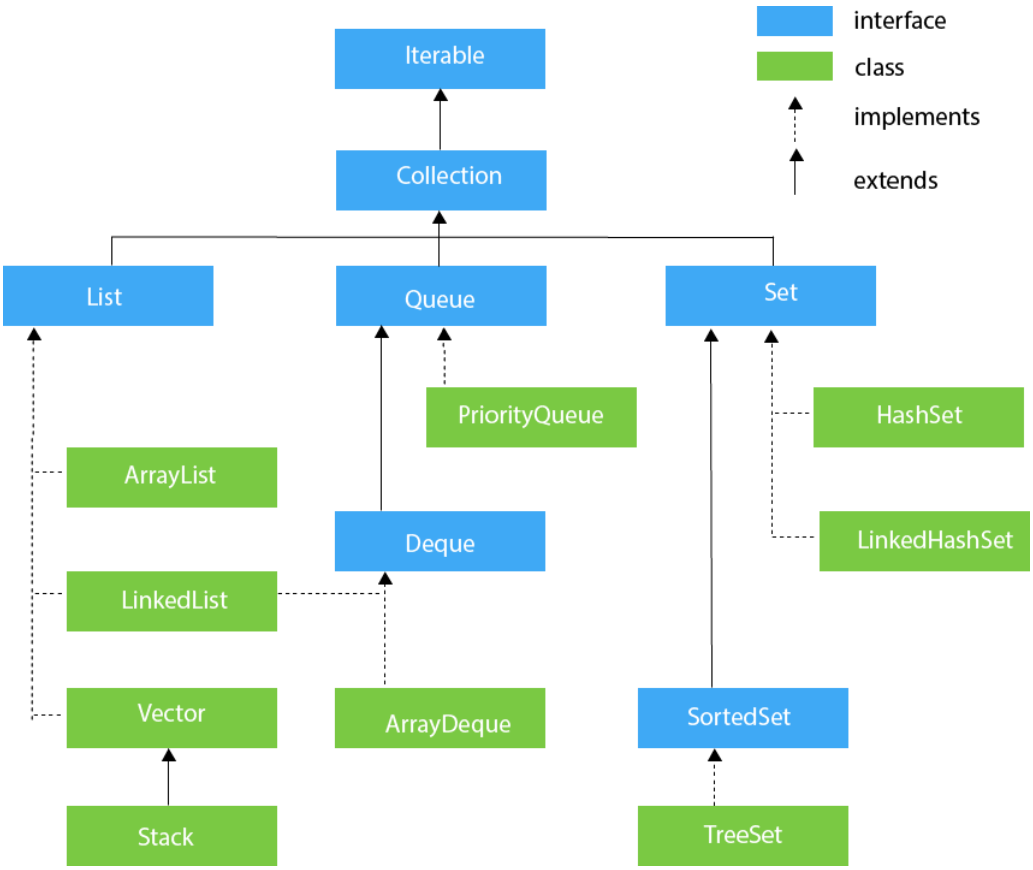
6) Java neden çoklu kalıtımı desteklemez açıklayın

Çoklu kalıtımı desteklememesinin sebebi ise karmaşıklığı azaltmaktır. İki veya daha fazla sınıftan kalıtım aldığımızı düşünelim. Bu sınıflar içerisindeki değerlerin farklı olduğu aynı metotlar bulunabilir. Daha sonra bunu kendi sınıfımızdan çağırırsak program bu metodun hangi sınıftan geldiğini, hangi metodun içerisindeki değerleri çalıştıracağını çözemez ve derleme hatası oluşur.

7) Build Tool nadir? Java ekosistemindeki build toolar neler açıklayın?

Build Tool yazılım derlemesinin oluşturulmasını sağlar. Java'daki build toollar **Maven, Gradle,Apache Ant dır.** Gradle Android uygulama geliştirilirken kullanılır.

8) Collection framework içerisindeki bütün yapıları önemli methodlarıyla örnekleyip açıklayınız.



List Interface:

ArrayList Sınıfı

Bu sınıfa ait dizilerin boyutları azaltılabilir ya da arttırılabilir.

```
ArrayList<String> isimler = new ArrayList<String>();
    isimler.add("Ahmet");
    isimler.add("Ayşe");
    isimler.add("Esra");
    System.out.print("ArrayList: ");
```

LinkedList Sınıfı

Bu listelerde elemanlar kendisinden sonra gelen elemanların bilgilerini tutarlar.

```
LinkedList<String> hayvanlar = new LinkedList<String>();
    hayvanlar.add("Köpek");
    hayvanlar.add("Kedi");
    hayvanlar.add("Sincap");
    hayvanlar.add(3, "Keçi");
    System.out.print("LinkedList: ");
```

Set Interface:

HashSet sınıfı: Verilerin yazıldığı sırayla geleceğini garanti etmez. Düzensiz bir şekilde çıktı gelebilir. Küme yapısına benzer. Aynı isme sahip birden fazla eleman eklenemez.

```
HashSet<String> filmler = new HashSet<String>();
    filmler .add("Umudunu Kaybetme");
    filmler .add("Esaretin Bedeli");
    filmler .add("Seven");
    filmler .add("Zindan Adası");
    filmler .add("Umudunu Kaybetme");
    filmler .add(Null);
    System.out.print("HashSet:");
```

Ekran Çıktısı: Esaretin Bedeli, Zindan Adası, Umudunu Kaybetme, Seven, Null

LinkedHashSet Sınıfı: HashSet’ten tek farkı, eklenen elementlerin ekleme sırasına göre tutulmasıdır.

```
LinkedHashSet<String> aylar = new LinkedHashSet<String>();
        aylar.add("Ocak");
        aylar.add("Şubat");
        aylar.add("Mart");
        aylar.add("Nisan");
        aylar.add("Mayıs");
        gunler.aylar("Ocak");
        System.out.print("LinkedHashSet:");
```

Ekran Çıktısı: Ocak, Şubat Mart, Nisan, Mayıs

TreeSet Sınıfı: TreeSet'lerde elementler artan sıralamayla tutulur.

```
TreeSet<Integer> sayilar = new TreeSet<Integer>();
        sayilar.add(3);
        sayilar.add(9);
        sayilar.add(1);
        sayilar.add(4);
        sayilar.add(3);
System.out.print("TreeSet:");
```

Ekran Çıktısı:1 3 4 9

Map Interface:

HashMap

HashMap'ler sıralamayı garanti etmez. null key ve null value kabul eder.

```
HashMap<String, String> sayi = new HashMap<String, String>();
    sayi.put(1, "bir");
    sayi.put(2, "iki");
    sayi.put(3, "üç");
    sayi.put(4, "dört");
    System.out.println(sayi);
```

Ekran Çıktısı: 1 bir , 2 iki, 3 üç, 4 dört

TreeMap

HashMap’lerden farkı, elementleri artan sıralama ile tutmasıdır.

```
TreeMap<Integer, String> günler = new TreeMap<>();
        günler.put(3, "Çarşamba");
        günler.put(4, "Perşembe");
        günler.put(2, "Salı");
        günler.put(1, "Pazartesi");
       System.out.print("TreeMap:");
```

Ekran Çıktısı: 1 Pazartesi, 2 Salı, 3 Çarşamba, 4 Perşembe

HashTable

HashMap null değer alırken HashTable almaz.

```
Hashtable<Integer, String> mevsim = new Hashtable<>();
            mevsim.put(1, "Kış");
            mevsim.put(2, "İlkbahar");
            mevsim.put(3, "Yaz");
            mevsim.put(4, "Sonbahar");
           System.out.print("HashTable:");
```