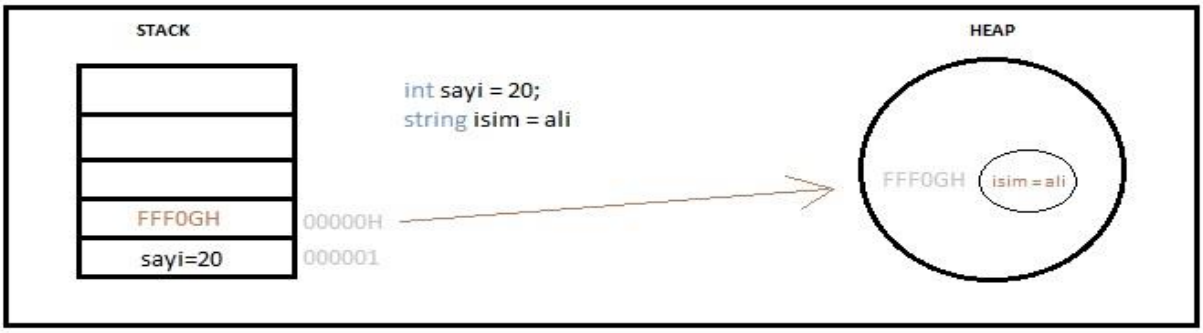
3. Java'nın platform bağımsızlığını nasıl sağladığını anlatınız.

Yazdığımın yazılımlar bilgisayarlar (makineler) üzerinde çalışır .Fakat biz bunları yüksek seviyeli günlük konuşma diline yakın bir şekilde yazılan diller (C# ,java) ile yazarız. Yani programı yazdığımızda makinenin bu kodu kodu anlayıp çalıştırması doğrudan mümkün değildir . Bunun için yazdığımız kodun makinenin anlayacağı düşük seviyeli makine diline çevrilmesi ve makineni çalıştırabileceği bir kod üretilmesi lazım. Bu iki türlü gerçekleşiyor derleme ve yorumlama şeklinde . Java hem derlenen hem yorumlanan bir dildir. Java da yazdığımız .java uzantılı dosyalar javac (java compiler) yardımıyla derlenerek .class uzantılı byte koda dönüşür (makine kodu ile yüksek seviyeli dil arasında bir kod).ve daha sonra bu class uzantılı dosya jvm tarafından yorumlanarak çalıştırılabilir bir kod parçası haline gelir.Jvm java kodlarının çalışması için gerekli düzenlemeleri sağlayan bir sanal platformdur. Jvm her makine için farklıdır örneğin Windows için farklı macos için farklı bir jvm vardır . Burada paltform bağımsızlığı sağlayan şey derlenerek .class uzantılı hale dönüşen dosyadır .Bu dosya jvm kurulu her makine jvm tarafından yorumlanıp çalışarak javanın platform bağımsız bir dil olmasını sağlar .

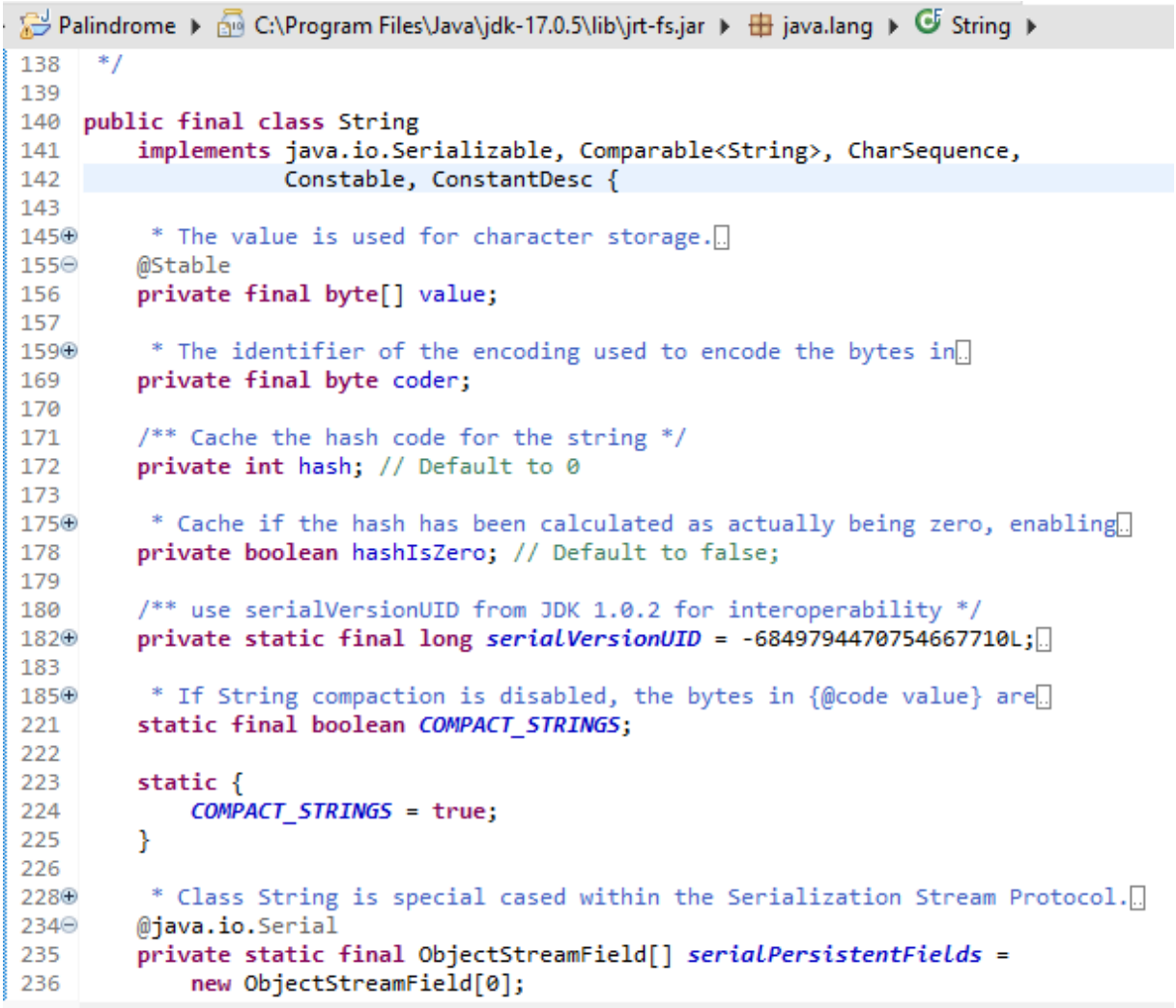
4. Java'da heap ve stack kavramlarını örneklerle açıklayın.

İkisi de ramde bulunan mantıksal alanlardır.Heap ve statken önce veri tiplerine bakmamız lazım. Bu veri tipleri primitif(değer tipli) ve referansa tipli olarak ikiye ayırabiliriz. Primitif tipler byte ,short ,int ,long, double,decimal,float tır. Referans tipler ise string, array ve sınıflar olarak yazabiliriz . Temel olarak primitif tipler doğrudan kendilerine atanan değeri tutarlar .Referans tipler ise kendilerine atanan değerin tutulduğu adresi tutarlar (işaret ederler).Primitif tipler stackte saklanırken referans tipli değişkenlerin kendileri stackte kendilerine atanan değerler ise heapte tutulur. heapte saklanır . Stack son girenin ilk çıktığı üst üste ve sıra ile kaydedilen bir yığındır .Heapte ise stackin aksine veriler dağınık olarak saklanır. Ayrıca heap ve stack arasındaki farklardan biri de tututaln verilerin silinmesiyle ilgili stackte tutulan değişkenler yaşam döngüleri bittiğinde(içerisinde tanımlandıkları süslü parantez kapandığında diyebiliriz sanırım) otomatik olarak silinirler,heapte saklanan verilerin ise kendisini işaret eden bir obje bulunduğu sürece silinmezler .Kendisini işaret eden bir obje kalmadığında ise çöp toplayıcı (garbage collector) tarafından temizlenirler. Aşağıdaki resimde görüldüğü gibi int türünde tanımlanan sayı değeri stackte tutulurken ,string türünde tanımlanan "ali" değeri heap te tutuluyor.



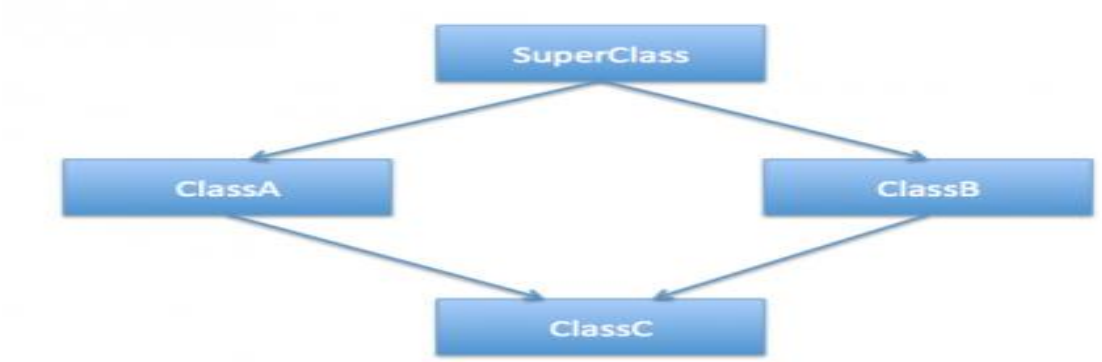
5. String class'ı nasıl immutable olmayı sağlamaktadır örnek ve çizimlerle açıklayınız.

String diğer wrapper classlar gibi immutable yani değiştirilemez bir classdır. Yani sınıf içerisinde bulunan değişken değerlerini değiştirimez ,yazılan methodları ovveride edemez ve bu sınıftan kalıtım alamayız.Class içerisinde tanımlanan her alanın (değişken method gibi) private ve final tanımlandığı ve setter metodları yazılmadığı için sınıf dışından bunlara müdahale edemeyiz değiştiremeyiz.Sınıfı ise final anahtar kelimesi ile tanımlayarak bu sınıftan kalıtım alınmasının önüne geçiliyor. Aşağıdaki fotoğrafta görüldüğü gibi string sınıfı bahsettiğim bütün yöntemleri kullanarak immutuable bir sınıf haline gelmiştir



6. . Java neden çoklu kalıtımı desteklemez açıklayın?

Çoklu kalıtım bir sınıfın iki farklı base sınıftan kaltım almasına denir . Java çoklu kalıtımın diamond problem olarak bilinen bir çıkmaza sebep olmasından dolayı çoklu kalıtımı desteklemez. Bunu aşağıdaki şema yardımıyla daha net anlatabilirim.



Şekilde görüldüğü gibi A ve B sınıfları normal kalıtım yoluyla süper classtan türetilmiş C classı ise çoklu kalıtım yoluyla A ve B sınıflarının her ikisinden kalıtım alarak oluşturulmuş. Şimdi abstract olan süper classın bir hızlan adında abstract methoda sahip olduğunu düşünelim. Doğal olarak A ve B classlarında bu hızlan metodunun gövdesini doldurmamız gerekiyor ve iki sınıfta bu methodların gövdeleri farklı bir biçimde dolduruluyor . Sonrasında ise C classından base sınıfa ait olan ve kalıtım yoluyla geçen hızlan methodu çağrılmak istendiğinde derleyici hangi sınıftaki methodu çalıştıracağını bilemiyor ve tam olarak problem de bu. Java çoklu kalıtımı desteklemzken çoklu implementasyonu destekliyor.bunu ise şöyle açıklayabilirim.İnterfacler sadece method tanımları bulunur asıl implementasyon ise interfaci implemente eden sınıf içerisinde yapılır bu sebeple çoklu kalıtımda karşımıza çıkan problem çoklu implementasyonda karşımıza çıkmıyor .

7.Build toolar yazdığımız projeyi oluşturmak ve yönetmek için kullanılacak bir araçtır bu toollar sayesinde projemizi kolayca build edebiliriz, jar ve war gibi paket formatlarında derleyebilir ve projenin bağımlılıklarını kolayca yönetebiliriz .java için kullanılan başlıca build toolar şunlardır.

1 Apache Maven

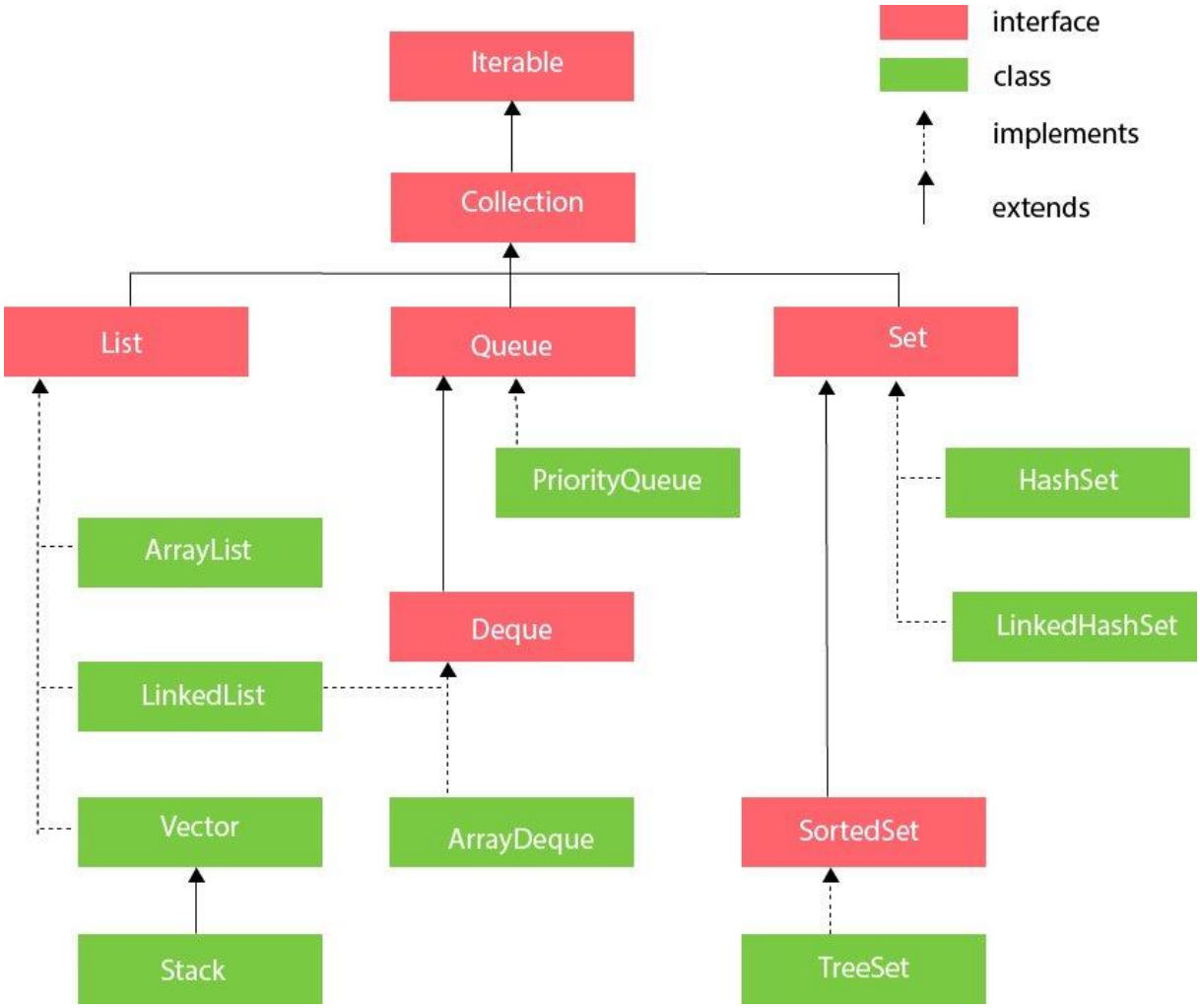
2 Ant with ivy.

3 Gradle.

4 SBT.

8.Collection framework içerisindeki bütün yapıları önemli methodlarıyla örnekleyip açıklayınız

Collection aynı türdeki nesneleri bir arada tutuabileceğimiz bu nesneler üzerinden berlirli işlemleri kolaylıkla yapmamızı sağlayan yapıdır.Ve aşağıdaki resimde görünen yapıda bir hiyerarşiye sahiptir.



Colllection yukardıda bahsettiğim gibi nesneler ile çalışmamızı sağlar .

Arraylist: Arrayler ile benzer özellikler gösterir verileri sıralı bir biçimde (giriş sırası) indexleyerek tutar.Arraylerin aksine boyutu dinamik olarak artar.tekrar eden veriller içerebilir.

Linkedlist: Burad ki elemanlar kendisinden sonra gelen eleaman ait bilgi ve ya adreslerini tutarlar .

Set: List ten farkı tekrar eden elemanlara izin vermemesidir bu özelliği ile matematikteki kümelere benzetebiliriz.

Hashset:öğeleri sırasız bir düzenden tutar .Elemanları hash tablosunda tutar bu elemanlara erişim amacyla elemanlara anahtar vasıtasıyla ulaşır .

Linkedset: Hashsetten farkı elemanlar eklanme sırasına göre tutulur.

SortedSet: ögeleri artan düzende tutar. Sıralamadan yararlanmak için birkaç ek işlem sağlar.

Queue:İlk giren ilk çıkar (fifo) mantığıyla çalışan bir yapıya sahiptir.

Stack: Burada ise queue nin aksine son giren ilk çıkar mantığı ile çalışır.

Sık kullanılan methodlar :

add(),clear(), size(), isempty() (isminden belli)

contains(object a) verilen objeyi içerip içermediğini kontrol edere boolean döndürür

remove (object a) verilen objeyi koleksiyondan kaldırı