INTERVIEW PREP
MÜLAKATLARA HAZIRLIK

Batch 81

Güçlü yönlerinizden bahseder misiniz?

Senaryo 1 : Öğrenmeyi çok seviyorum. Günceli takip etmeyi, yaptığım işlerde güncel toolları kullanmayı tercih ederim. Değişikliklere çok kolay adapte olurum. Çalışma ortamında, sistemlerde güncelleme ya da değişiklikler olduğunda kolaylıkla adapte olabiliyorum.

Test Hiyerarşisi ya da Test Piramidi Hakkında Ne Biliyorsunuz?

Testlerin hangi sıraya göre yapılacağını ifade eder.

Test piramidi test hiyerarşisini gösteren yapıdır. Bir fonksiyonun testi yapılırken önce unit, sonra integration, E2E ve UI testleri yapılır. Regressionda da sorunsuz çalışmasının ardından UAT testler yapılarak ürün canlıya alınabilir. Yani ürünün oluşturulmasından canlıya alınmasına kadar geçen süreçteki işlemleri kapsar

Verilen taski zamanında bitiremediğinde ne yaparsın?

Öncelikle taskin neden zamanında bitirilemediği önemli. Benim dışında gelişen aksaklıklar varsa öncelikle bununla alakalı takım liderine bilgi veririm. Yaşanan aksaklıkların giderilmesi için bir eylem planı belirlerim ve takım liderini bu konuda bilgilendiririm. Bu aşamadaki koordinasyon sağlamam gereken kişilerle iletişime geçerim. Sorunlar çözüldükten sonra bu taski önceliklendirerek en kısa tamamlamaya çalışırım

Verilen taskin en iyi biçimde anlamak için ne yaparsınız?

Taskleri grooming toplantısında netleştiriyoruz. Yapılacak işin içeriğini tam olarak anlamaya çalışıyoruz. Task ile ilgili bir problem varsa bu toplantıda gideriliyor. Tüm dev ve testerların yanı sıra BA de toplantıda hazır bulunuyor. Gereken desteği bizlere BA sağlar.

Bir testin tamamlandığına ve yeterli olduğuna nasıl karar verirsiniz?

Testin tamamlanması süreci tek başıma karar verebileceğim bir şey değil. Öncelikle yazılan test caseler fonksiyonu test etmek için yeterli mi onu kontrol ederim. Bunun için requirementları, kabul keriterlerini incelerim. Test caselerin tüm fonksiyonu kapsayacak şekilde hazır olduğuna emin olduktan sonra testleri koşarım. Testlerin tamamlanmasında Definition Of Done(DoD) bitti tanımı bizim için önemlidir. Bu tanıma uyuyorsa test lead ve PO nun da onayı ile test işlemini sonlandırırım.

Java'nızı puanlar mısınız?

Ben bir puanlama yapmak yerine size neleri rahat bir şekilde kullanabildiğimi söyleyebilirim

1- OOP Concept
2-Data Structure
3-String Manupilation
4-Exceptions

Tabi ki bu cevabın yeni sorulara gebe olduğunu unutmamak gerekir. İfade ettiğimiz her şeyin açıklaması bizden istenebilir.

OOP Concept Nedir? Projede nasıl kullandınız

1- Inheritance(Miras)
2- Encapsulation(Data Hiding)
3- Abstraction (Soyutlama)
4- Polymorfizm(Çok biçimlilik)

Projede nerede kullandınız?

Data Structure Nedir?

Primitive(8 tane byte,short,int ,long,char,boolean,double,float)
Non-primitiveler(String, Wrapper Class(Byte,Integer,.....))
Collection (listler,array,set, queue,deque,linkedlist..)
Map+

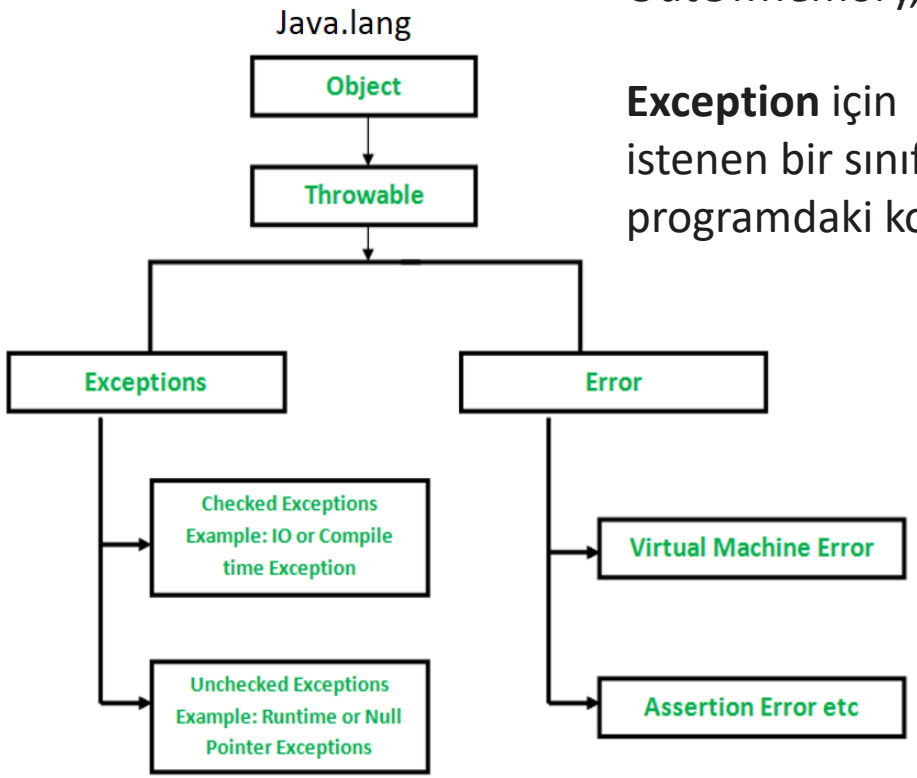
String Manupilation ?

String methodları

- Uppercase,lovercase
- İndexof, charAt, substring, length,
- Contains,equals,……

Exceptionlardan bahseder misiniz?

Error, sistem kaynaklarının eksikliği nedeniyle ortaya çıkan kritik koşullardır ve programın koduyla ele alınamaz. Hatalar hiçbir şekilde telafi edilemez, çünkü bunlar yaratılamaz, atılamaz, yakalanamaz veya cevaplanamaz. Örneğin OutOfMemory, StackOverFlow.



Exception için yanlış koddan kaynaklanan hataların olduğunu söyleyebiliriz. Örneğin, istenen bir sınıf bulunamazsa veya istenen bir yöntem bulunmazsa. Bu tür Exceptionlar programdaki koddan kaynaklanmaktadır; Bu türExceptionlardan sistem sorumlu değildir.

kaç tip exceptions biliyorsunuz?

- (checked exception)/Compile time exception
  Ör: Thread.sleep(ms); → Handle edilmez ise Java bunu compile edemez

- Unchecked Exception/Runtime exceptions
  Ex: ArithmeticException (int i=100/0;),
  NoSuchElementException
  Bu exceptions runtime da meydana gelir ve developer isterse bu sorunu çözer.

Exception ile mücadele edebilmek için neler yaparsın?

try...catch block.

Try catch ile birden fazla exceptionu nasıl handle edersin?

Bazen try içerisindeki kod bloğunda, birden çok hata oluşabilecek durum meydana gelebilir. Böyle durumları kotarmak için birden fazla catch cümleciği arka arkaya kullanılır. Bir hata meydana geldiği anda, her catch ifadesi sırasıyla denetlenir ve istisnanın tipiyle uyuşan catch cümleciği çalıştırılır.

```
publicstaticvoid main(String[] args) {
        try {
            Scanner sc=new Scanner(System.in);
            System.out.println("1.sayıyı girin:");
            doublebolunen=sc.nextDouble();
            System.out.println("2.sayıyı girin:");
            doublebolen=sc.nextDouble();
            doublesonuc=bolunen/bolen;
            System.out.println("Sonuç="+sonuc);
        } catch (InputMismatchException hata1) {
            System.out.println("Lütfen sadece sayı giriniz.");
        }
        catch(ArithmeticException hata2)
        {
            System.out.println("Bölen sayı 0 olamaz");
        }
        finally{
            System.out.println("İşlem sonlandı.");
        }
    }
```

What is finally keyword?

- **Finally block nedir?**
  - finally block java exception handling icin kullanilir. Finally block u sadece try ile ve ya try-catch ile birlikte kullanabiliriz.finally block icindeki code her zaman calisir.

- **finally block'un calismadigi zamanlar hangileridir ?**
- System.exit(0); bu kod java programini durdurur ve finally block calismaz
- JVM ===> heap hafizasi dolu oldugu zaman çalışmaz (outOfMemoryError)
- stack hafizasi dolu oldugu zaman çalışmaz (StackOverFlow error)

int ve Integer farkı nedir? Integer niye kullanırız?

int → primitive

Integer → Non-primitive(Wrapper class)

Sayılar üzerinde methodlar yardımı ile işlem yapmak için kullanılır.

- valueOf() methodu bir String'i Wrapper'a çevirir.
- shortValue() methodu bir Wrapper'ı primitive çevirir.
- byteValue() methodu bir Wrapper byte çevirir.
- parseInt() methodu Wrapper primitive inte çevirir.

Access modifier nedir?

Erişim Düzenleyiciler

- Public -->Herkes ,her yerden ulaşabilir
- Private --> Aynı class tan ulaşılabilir
- Default -->Aynı package içerisinden ulaşılabilir
- Protected -->Aynı package içerisinden ulaşılabilir yada child lar ulaşabiliyor.

method overloading - method overriding

- Overloading ve overriding arasındaki ilk ve en önemli farklılık
- Overloading de **method isimleri aynı ancak parametreler farklı** olmalıdır.
- Overriding de ise method adı ve parametreler aynı olmalıdır. Ancak body değişir.
- Overloading ve overriding arasındaki ikinci büyük farklılık ise
- Overloading aynı class içerisinde gerçekleştirilirken, overriding aralarında inheritance relationship olan iki farklı class arasında gerçekleştirilir
- static, private ve final methodlar override edilemezler. Ancak overload edilebilirler
- Overload methodlarda return type aynı yada farklı olabilirler ancak override da return type aynı olmalıdır.

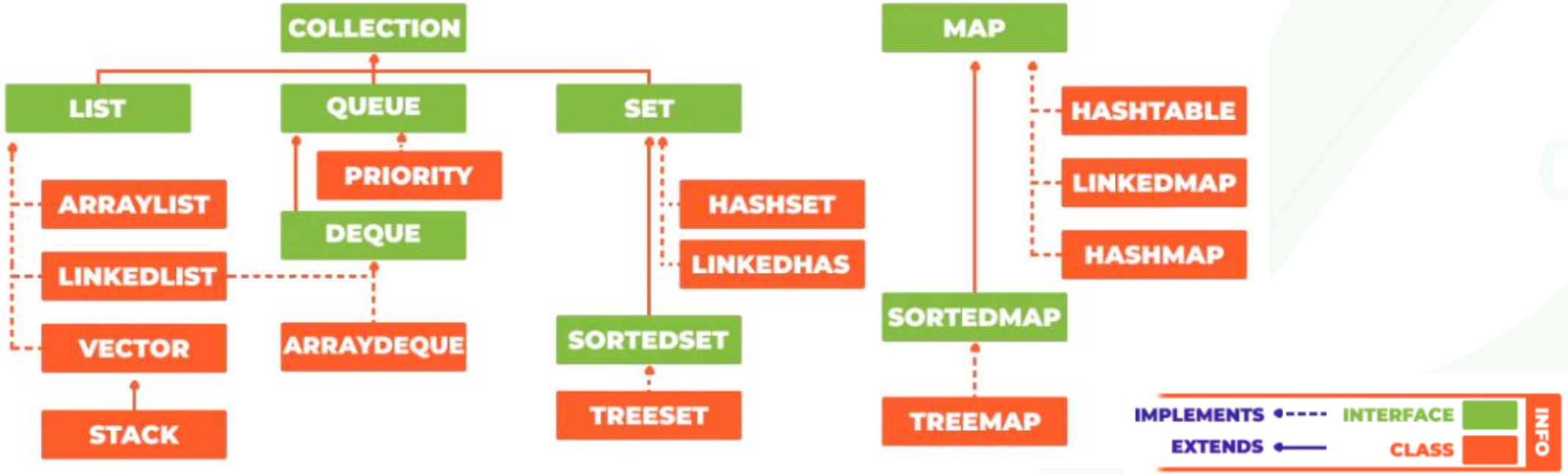
Constructor overloading ne demek?

- **Constructor**, Oluşturulan classların object olarak tanımlanması durumunda proje dosyasının alt yapısını hazırlayan, kurucu rol üstlenerek çeşitli ilk işlemleri gerçekleştiren, kullanılan sınıf yapısı ile aynı isme sahip olan, geriye değer döndürmeyen özel metot yapılarıdır.
- Constructor overloading ise, constructorın aynı class içerisinde birden fazla şekilde tanımlanmasıdır.

```
public toplam(){
}
public toplam(int i,int j){
        this.i=i
        this.j=j
}
```

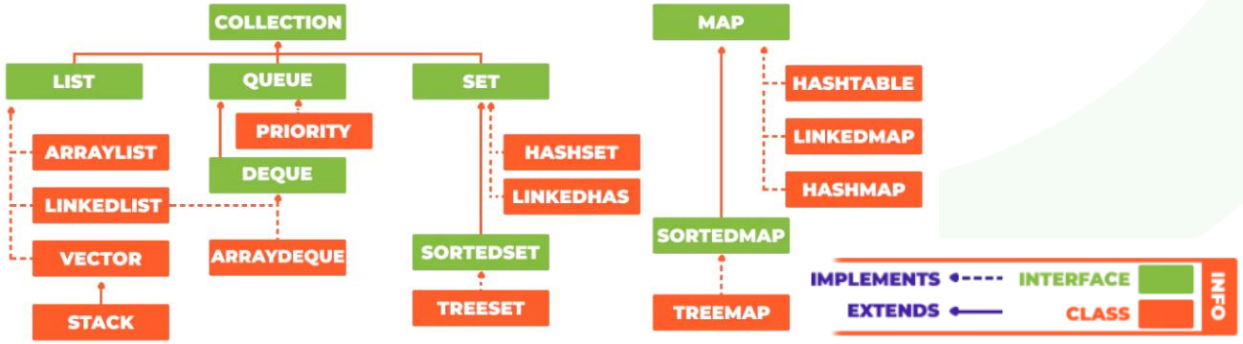
Collectionlardan bahseder misin?

Collectionlar içerisinde veri yığınları tutabildiğimiz yapılardır. Bunlar data structure kategorisinde ele alınırlar. Bu oluşumların en temelinde **List**, **Set** ve **Map** olmak üzere üç tip yer almaktadır. Bu kavramlardan Set ve List, **Java Collection** interface'ini kullanırken Map kavramı ise ayrı bir tür olarak bu yapıdan ayrılır. Bu yapıların bütününü ele aldığımızda ortaya Java Collection Framework çıkmaktadır.



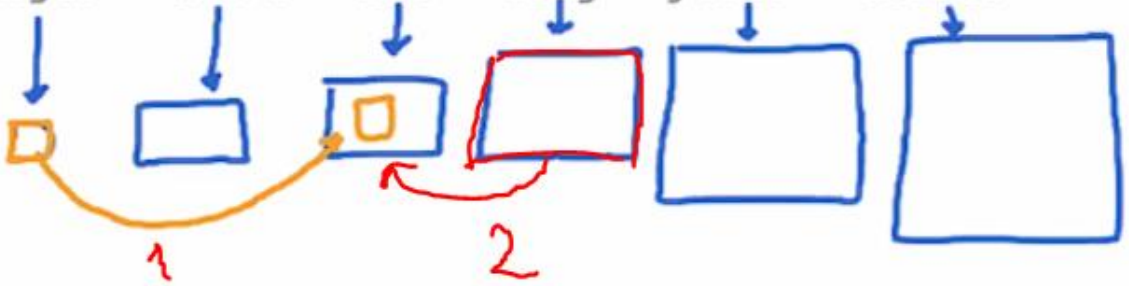
Collectionlardan bahseder misin?

•**List Bölümü:** Tanımlanan değerleri hafızasında sıralı bir şekilde tutar ve tekrarlanan değer atamaları yapılabilir.
•**Set Bölümü:** List bölümünün aksine kopya ya da tekrarlanan herhangi bir atama barındırmaz.
•**Queue Bölümü:** Kuyruk anlamına gelmektedir, ilk atanan değer her zaman ilk olarak çıkar. Bu yapısı ile birlikte **FIFO**(first in-first out) kavramı ile birlikte anılır.
Dikkat ettiyseniz tablo üzerinde olduğu gibi **Collection** ve **Map** kısmını ayırarak ele aldım. Temel olarak aynı mantıkta çalışsalar bile yapı olarak farklılık gösteriyorlar. Bu farklılığı anlayabilmek adına map kavramını kısa bir madde ile tanımlayabilirim.
•**Map Bölümü:** Tanımlanan değerlerin kendisine özel bir anahtar ile eşleştiren interface bölümüdür. Tanımlanan değerler **key** ve **value** olarak hafızada tutulur. Set bölümünde olduğu gibi tekrarlanabilir, kopya bir anahtar tanımlanması yapılmaz.



Type Casting Nedir?

```
//Type Casting: Bir [redacted] data type'ini diger [redacted] data type'ina cevirmek demektir.
//              byte < short < int < long < float < double
```




```
//Note 1: Kucuk data type'ini buyuk data type'ine cevirmeye "Auto Widening" denir
//Note 2: Buyuk data type'ini kucuk data type'ine cevirmeye "Explicit Narrowing" denir
```

1

2

```
byte age = 23;
int newAge=age;

long population = 1234;
int newPopulation = (int)population;
```

Sadece primitivelerde değil, tüm data typelarda casting işlemi yapılabilir

Static keywordu tanımlar mısınız?

Değişken → bir class içinde aynı değişkenin her yerde kullanılabilmesi için static tanımlaması yapmamız gerekir

```
public class Araba {
    int model_yil; //Nesneye ait değişken
    String plaka;  //Nesneye ait değişken
    String marka;  //Nesneye ait değişken
    static int araba_sayisi=0;   //Sınıfa ait değişken
    Araba (int model_yil, String plaka, String marka){
        this.model_yil=model_yil;
        this.plaka=plaka;
        this.marka = marka;
        araba_sayisi++; //Her yarattığımız nesne için araba sayımız 1 artsın.
    }
}
```

```
public static int getAraba_sayisi() {
    return araba_sayisi;
}
```

```
public static void main(String args[]){
    System.out.println("Başlangıçtaki araba sayımız "+Araba.getAraba_sayisi());

    //İki tane araba nesnesi oluşturalım.
    Araba a1 = new Araba(2015,"34 UG 6845","Hyundai i20");
    Araba a2 = new Araba(2014,"01 AK 3981","Nissan Juke");

    int araba_sayimiz;

    araba_sayimiz = Araba.getAraba_sayisi();
    System.out.println("Araba sayısı : "+araba_sayimiz);


}
```

Method → Normalde bir sınıftaki bir metodu çağırmak istiyorsak o sınıfın nesnesi üzerinden çağırırız. Nesnelerden bağımsız değişkenler yaratabildiğimiz gibi bağımsız metodlar da yaratabiliriz. Bunun için static anahtar kelimesini kullanırız. Bu metodları çağırabilmek için nesne oluşturmamız gerekmez ve direk sınıf üzerinden çağırabiliriz.

Static keywordu tanımlar mısınız?

Static Block

- Static block static variable'lara deger ataması yapmak icin kullanılır
- Static block, class ilk çalıştırılmaya başlandığında çalışır ve variable'lara değer ataması yapar.

```
class Test {
    static int i;
    int j;
    static {
        i = 10;
        System.out.println("static block called ");
    }
    Test(){
        System.out.println("Constructor called");
    }
}

class Main {
    public static void main(String args[]) {

       // Although we have two objects, static block is executed only once.
       Test t1 = new Test();
       Test t2 = new Test();
    }
}
```

- Static block'lar constructorlardan, tum methodlardan ve main methoddan once çalışır.
- Eğer 1'den fazla static block varsa üstteki once çalışır