1) Pass by value, pass by reference kavramları nedir ? Java' ile ilişkili olarak açıklayınız
Function çağırdığımızda parametre olarak yolladığımız değerleri value yada referans olarak yollayabiliriz. Value olarak yolladığımızda bir nevi değerin kopyasini yolluyoruz ve function tarafından yapılan değişiklikler ana variableimiza etki etmiyor. Basitçe şöyle bir örnek verebiliriz:

```
public static int test (int n){
   return n+1;
}

public static void main(String[] args) {
    int passByValue = 1;
    System.out.println(test(passByValue));
    System.out.println(passByValue);
}
```

output:

```
2
1
```

Pass by referansda yollanılan variable'ın memory refereansı yollanıyor ve yapılan değişiklikler doğrudan ana variablea etki ediyor.
Fakat pass by referans kısmında Java'da birkaç durum haricinde daima pass by value kullanılıyor.

```
public static void test (int n[]){
   n[0]++;
}

public static void main(String[] args) {
   int passByReference[] = {0};
   System.out.println(passByReference[0]);
   test(passByReference);
   System.out.println(passByReference[0]);

}
```

output:

```
0
1
```

2) Immutability nedir, neden önemlidir ? Bir Java sınıfı nasıl immutable yapılır?
Immutable değiştirilemez nesneler oluşturulduktan sonra değiştirilemeyen sınıflar diyebiliriz. Değişmeyen bir nesneyi değiştirmek istersek onu klonlamamız gerekir. Java 8'den sonra mutable Date sınıfı yerine immutable LocalDate sınıfı gelmiştir.
3) Framework ve library arasındaki fark nedir ?
Libraryler fonksiyon kümelerinden oluşur ve fonksiyonlar birtakım işler yaparak sonucu alıcıya return eder. Fakat Frameworkler design kalıpları vardır kontrol akışı bellidir ve o düzen içerisinde ilerlemen gerekir. Kısaca
You call library, Framework calls you.

4) Java'da Garbage Collector' un görevi nedir ?
   Javada bellek yönetimi için arka planda garbage collector vardır JVM içinde çalışır. Memory management kısmını halleder. Referans edilmeyen objeleri silerek memory de yer açılmasını sağlar.

5) Memory leak nedir? Java'da memory leak oluşması mümkün müdür?

   Memory harcanırken yarattığımız objeler veya variablelar memory harcar ve daha sonrasında bunları deconstruct etmemek memoryi doldurabilir. Memory'de yer kalmaması ve kaynaklara erişilememesi memory leak durumuna yol acar. Javada referans edilen objelere garbage collector toplamayacağı için memory leak olusmasi mümkündür. Örnek olarak Statik objeler yaratıp unutmamız memory leak oluşmasına neden olabilir.

6) Java sürümleri ne sıklıkla çıkmaktadır
   Major updateler genellikle 6 ayda bir cikmaktadir. March ve September aylari icerisinde.

7) Stack ve Heap nedir Java'da hangi yapılar stack ile, hangi yapılar heap ile ilişkilidir?
   Stack ve heap genelde memory de kullanilan mantıksal yapilardir. Stack yapisinda LIFO(Last in first out) mantigi ile dizilir ve sirasi gelmeden ara değer kullanılmaz.
   Class referanslari stackde içindeki variablelar ise heapde saklanır.
   Heapde yapı karışık olarak tutulur ayrıca garbage collector de heap mantığı ile çalışır.

8) OpenJDK ve OracleJDK arasındaki farklar nelerdir?

   OpenJDK ve OracleJDK arasındaki en büyük fark lisanslarıdır. OpenJDK tamamen open source bir projedir. OracleJDK commercial lisans gerektirir. Kurumsal support almak için OracleJDK tercih edilebilir.

9) @FunctionalInterface anotasyonu nerelerde kullanılabilir, neleri sağlar?
   FunctionalInterface annotasyonu içerisinde sadece bir tane abstract metod olan interfacedir. Functional interfaceler, lambda expressionların kullanılabilmesi için tanımlanır.
   @FunctionalInterface anotasyonunu eklemek zorunlu derğildir fakat doğrulanması için önemlidir. Aksi taktirde birden fazla abstract metod bulunan interfacelerde compile error hatası verecektir.

10) Java'da hangi functional interface'ler yer almaktadır? Yaptığınız araştırmada en popüler/göze çarpanlar hangileridir?

    Runnable, ActionListener, Comparable, Predicate, Operations, Consumer, Supplier gibi functional interfaceler vardır. En çok kullanılan Runnable olabilir çünkü, Threading de kullanılıyor.

Collection Pipeline

- First encounters
  Collection pipeline yazılımda en çok kullanılan design kalıplarından biridir. Unixde grep xargs sort gibi komutları pipelinelayıp istediğimiz sonucu elde edebiliyoruz. Aynı

kalıplar OOP dillerde de buluuyor yazıda Ruby ve smalltalks örneği verilmiş ve articlelar içerisinden pipeline yapılarak istenilen articlea ulaşılmaya çalışılmış.

- Defining Collection Pipeline
  Collection pipeline yazılımı modüle edilmesi üzerine oluşturulan bir sistemdir. Unixde koleksiyonlar her satır boşluklarla ayrılmış değerler içerir. Nesne yönelimli programlamada ise koleksiyonlar nesnelerdir. Kendileri kolekisyon olabilir yada daha fazla koleksiyon içerebilir. Functional dillerde oop pipeline yerine hashmap kullanılır.

- Exploring more pipelines and operations
  Pipeline operationsları arasında toplam kelime sayısını bulma veya basic matematik operationsları yapma örnekleri verilmiş Ruby ile.
  Bir sonraki örnekte ise pipeline ile kaç tane article olduğunu bulmak için bir örnek vermiş ve group_by kullanmış.
  Bir sonraki örnekte her bir tagden kaç tane olduğunu bulmak için bir örnek yapılmış, önce tagları mapleyip gruplamış, daha sonra sonucu tekrar mapleyip Word countlarını toplamışve ekrana basmış.

- Alternatives
  Collection pipelineların döngülerle nasıl yazılacağıyla alakalı örnekler verilmiş.
- Nested Operator Expressions
  Collectionların set operationlarla manipüle edilmesiyle alakalı ruby ve clojure örnekleri verilmiş. Union, intersection gibi operationlarla collectionlardan istediğimizi elde edebiliriz fakat nested operatör expressionları tercih ettiğini belirtmiş.

- Laziness
  Büyük yavaş kompleks metodlarda performansı arttırmak amacıyla kullanılan bir metod olarak tanımyabilirim. Örneğin büyük verilerde sadece ihtiyacımız olan kadarını almak gibi (Emin değilim tam anlayamadım)

- Parallerism
  Birçok pipeline operationu kullandığımızda bir element diğer bir elemente bağlı değilse veya etkilemiyorsa paralel olarak birçok çekirdek kullanarak hızlandırabiliriz.

- İmmutability
  Collection pipelines immutable veri yapılarıdır. Bir pipeline oluşturduğunuzda her operasyon bir yeni collection üretir ve sonraki pipelinea input olarak yollar.

- Debugging
  Modern IDE ler debugging için birçok tool sağlıyor fakat başka debugging yolları olarak kritik noktaları ekrana yazdırarak yada mapleyerek debug edebiliriz.